Introducción a Linux y al Software Libre.

Ibán Martínez Gómez

Antes de empezar ...

Este manual pretende servir de pistoletazo de salida para todas aquellas personas cuyas inquietudes en el campo de la informática, van más allá de lo puramente lúdico o profesional. Es algo evidente que la informática ha cambiado nuestras vidas, no siempre para bien, pero en este caso, el software libre sí que puede ser un buen cambio, ya que dicho movimiento va en favor de la gente con inquietudes, espíritu emprendedor y con ganas de hacer su vida un poco más interesante mediante el conocimiento.

Introducción a Linux y al software libre, no es un remedio milagroso, en él no está el saber absoluto de Linux, no, simplemente es el comienzo de la aventura, nada más. El verdadero milagro está dentro de nosotros, en las ganas de aprender, de crecer como ser humano, esa sí que es la verdadera fuente del saber, por eso, desde estas líneas invito a no desaprovechar jamás ese talento.

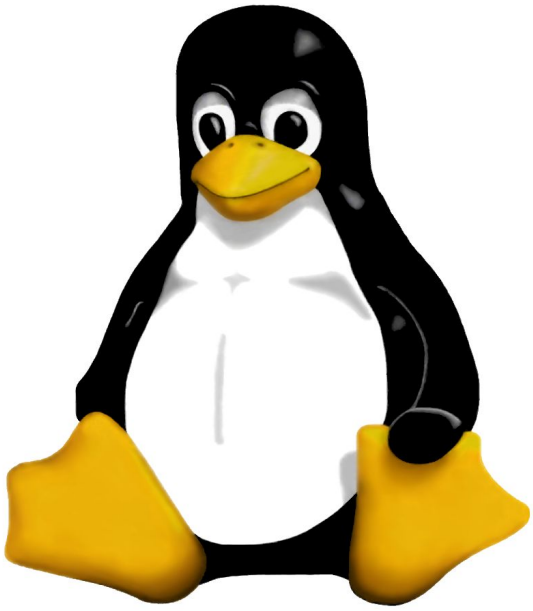
Ibán Martínez Gómez

nnset@spymac.com

2004

Índice

Tema 1.
El Software Libre y Linux.



Quién o qué, sería el más adecuado para definir lo que es el software libre, o simplemente un proyecto abierto, pues bien, lo ideal sería que existiese una enciclopedia mundial, abierta, gratuita y escrita en cooperación con gente de todo el planeta, pero claro, eso es imposible ... o no. Gracias al movimiento del software libre, se inició un proyecto llamado wikipedia, ¿qué es eso exactamente? Bueno, no es más que una enciclopedia online escrita de forma cooperativa entre usuarios de la red de redes. Así que, bajo los términos legales en los que se basa wikipedia, le cedemos el turno a ella para que nos aclare qué son Linux y el software libre.

De Wikipedia, la enciclopedia libre. *http://es.wikipedia.org/wiki/Software_libre*

Software libre es el nombre por el que se le conoce a cierto tipo de software, y al movimiento que lo promueve, que se caracteriza por promover las siguientes libertades y obligaciones al usuario final y a los desarrolladores:

- Libertad de ser **utilizado** por cualquier persona para **cualquier** propósito.
- Libertad de ser copiado y **distribuido libremente** (se le puede pasar al vecino sin ningún problema). En este punto hay una diferencia importante entre las licencias tipo BSD y tipo GPL.
  - **BSD**: La libertad de distribución es total. Esto incluye cambiar la licencia, "cerrando" el código.
  - **GPL**: La redistribución de versiones modificadas ha de hacerse bajo la misma la licencia. De este modo se impide que de programas licenciados bajo GPL puedan derivar en programas propietarios.
- Libertad de estudiarlo (para esto es indispensable contar con el código fuente).
- Libertad de modificarlo. Si se redistribuyen los cambios en el caso de GPL es obligatorio hacerlo bajo la misma licencia (por supuesto, el código fuente también es prerrequisito de este enunciado). Si se modifica para uso interno y no se redistribuye, no es necesario publicar los cambios. En este punto también hay matices, por ejemplo entre la GPL y la LGPL:
  - **GPL**: Cualquier código que utilice código GPL (a través del aporte de código, enlazamiento de librerías etc), debe ser necesariamente liberado bajo la GPL.
  - **LGPL**: Se pueden enlazar dinámicamente librerías LGPL con programas con licencias distintas, incluso propietarios.

El *software libre* no debe confundirse con el software de dominio público, el cual no tiene restricción de distribución alguna. De ello se desprende que el software libre es un software protegido, cuya licencia de distribución es un poco peculiar, pero que debe ser respetada como cualquier otra licencia de distribución .

Mitos sobre el software libre.

Los detractores del software libre utilizan muchos argumentos para desestimar su calidad o sus beneficios. Entre los argumentos más utilizados se puede mencionar:

- El coste de mantenimiento y entrenamiento de personal para trabajar con software libre supera el coste de compra, mantenimiento, actualización y entrenamiento para un software propietario.

  La adopción de software libre en empresas bancarias en América Latina ha demostrado lo contrario.

- El software libre *contamina* debilitando los derechos de autor sobre otras obras.

  El exigir que el software derivado cumpla con ciertas reglas de distribución es mucho menos restrictivo que prohibir la creación y distribución de productos derivados de la mayoría de las otras licencias.

- El software libre es más propenso a ser atacado por infracciones de derechos de autor.

  Se pretende que un distribuidor comercial tiene mayor control sobre sus desarrolladores por lo que cometerán menos infracciones a las leyes de derechos de autor, sin embargo han existido hasta el momento muy pocas decisiones judiciales sobre el tema a pesar del tiempo transcurrido desde que existen las licencias de software libre.

- A diferencia del software propietario, el software libre no ofrece garantías ni protección contra demandas por derechos de autor.

  Es cierto que los distribuidores de software ofrecen en general muy pocas garantías, pero este punto no es específico a los vendedores de software libre. Como ejemplo de ello, las garantías ofrecidas por Windows 98 son muy limitadas.

- Con la licencia GPL se corre mucho riesgo dado que nunca ha sido defendida en un juicio.

  Cierto, pero ¿es más segura una licencia que ha sido disputada frecuentemente o una que nadie ha disputado?

- Los métodos de desarrollo por software libre cortan la innovación.

  Este argumento no tiene sustento alguno. El servidor HTTP Apache, el distribuidor de correo sendmail, el servicio de nombres Bind son ejemplos de software libre que han sido líderes en su área de aplicación.

Una vez wikipedia nos ha facilitado una buena y sencilla definición de lo que es el software libre, nos falta ver cómo encaja en esa defínción el sistema operativo Linux, damos paso a wikipedia.

De Wikipedia, la enciclopedia libre. *http://es.wikipedia.org/wiki/GNU/Linux*

GNU/Linux es la denominación defendida por **Richard Stallman** y otros para el sistema operativo que utiliza el **kernel Linux** en conjunto con las aplicaciones de sistema creadas por el proyecto GNU. Comunmente, este sistema operativo es denominado simplemente Linux.

Desde 1984, Richard Stallman y voluntarios están intentando crear un sistema operativo *libre* con un funcionamiento similar al UNIX, recreando todos los componentes necesarios para tener un sistema operativo funcional que se convertiría en el sistema operativo GNU.

En el comienzo de los años 90, después de seis años, GNU tenía muchas herramientas importantes listas, como compiladores, depuradores, intérpretes de comando etc, excepto por el componente central: el núcleo. Con el surgimiento del kernel Linux, esta laguna fue llenada y surgió el sistema operativo con el kernel Linux en conjunto con las herramientas GNU. De esta manera, Stallman juzga que este sistema operativo es una "versión modificada" del sistema GNU y por lo tanto debe tener la denominación GNU/Linux. Esta denominación resolvería la confusión entre el nucleo y el sistema operativo completo a que puede llevar, y de hecho ha llevado, la denominación Linux, en solitario. Stallman también espera que con el aporte del nombre GNU, se dé al proyecto GNU que él encabeza el reconocimiento que merece por haber creado las aplicaciones de sistema imprescindibles para ser un sistema operativo compatible con UNIX. Es conocida la cita de Richard Stallman: *«It's GNU/Linux, dammit!»* (*«¡Es GNU/Linux, maldita sea!»*).

Richard Stallman ha reconocido que desde que existe Linux el desarrollo de un nucleo específico del proyecto GNU (el Hurd) ya no es prioritario. Esto explica que después de dos décadas desde el anuncio del proyecto GNU, un sistema unicamente GNU no esté acabado.

Algunas distribuciones, apoyan esta denominación, e incluyen *GNU/Linux* en sus nombres, tal es el caso de *Debian GNU/Linux* o GNU/LinEx. En el proyecto Debian también existen Debian GNU/Hurd y Debian GNU/BSD que combinan las aplicaciones de sistema de GNU con esos nucleos.

En ocasiones, el proyecto KDE ha utilizado una tercera denominación: GNU/Linux/X para enfatizar los tres proyectos sobre los que se apoya su entorno de escritorio.

Algunos sectores de la comunidad de usuarios del sistema operativo han rechazado la denominación GNU/Linux por varias razones, entre ellas que ya se había empezado a denominar Linux al sistema operativo antes de que Richard Stallman promocionase esta denominación. Otras personas se oponen a la postura ideológica de Stallman radicalmente en contra del software propietario y por ello son contrarios al uso de este nombre para evitar la promoción de las ideas del fundador del proyecto GNU. Otros sectores de la comunidad han reconocido la conveniencia de este nombre.

Hay que señalar que, al igual que es una simplificación denominar al sistema que usa el usuario final *Linux*, obviando las aplicaciones GNU que completan el sistema operativo, el conjunto linux+GNU representa solamente una parte (aunque importante) del software encontrado en una distribución Linux. Existe una gran cantidad de software original del sistema operativo BSD o producido independientemente de los proyectos GNU y Linux por otras personas u organizaciones, como por ejemplo Apache, el X Window System, Samba, KDE, OpenOffice.org y miles de otros.

Tema 2.
Distribuciones Linux.

Poco a poco nos vamos introduciendo en ese mundo idílico del software libre, ahora vayamos un paso más allá, como sabemos qué es Linux, veámos de que manera se *distribuye* por el mundo. De nuevo nos serviremos de wikipedia.

De Wikipedia, la enciclopedia libre. *http://es.wikipedia.org/wiki/Distribuci%C3%B3n_Linux*

Una **distribución Linux**, o distribución GNU/Linux es un sistema operativo completo, basado mayoritariamente o totalmente en software libre con posibilidad de ofrecer software propietario o comercial, pero usado como núcleo Linux.

Algunos ejemplos de distribuciones de Linux son:

- Debian.
  - Knoppix.
  - LinEx.
  - Guadalinex.
- Gentoo.
- Linspire (antes Lindows).
- Mandrake.
- Red Hat.
  - Fedora Core.
- SuSE.
- Slackware.
- BestLinux.
- ¡Y cientos más!

Estándares interdistribuciones.

Linux Standard Base (Fundación de estándares Linux) es una organización consagrada a desarrollar una cooperación estrecha entre diferentes distribuciones. El Filesystem Hierarchy Standard (Estándar jerárquico de sistema de ficheros) es una importante herramienta de la organización para lograr una cierta normalización oficial.

Alien es un programa para poder convertir entre múltiples formatos de distribución como deb, rpm o tgz. Te permite adaptar un programa "empaquetado" para una distribución y compatibilizarlo en el formato de tu distribución.

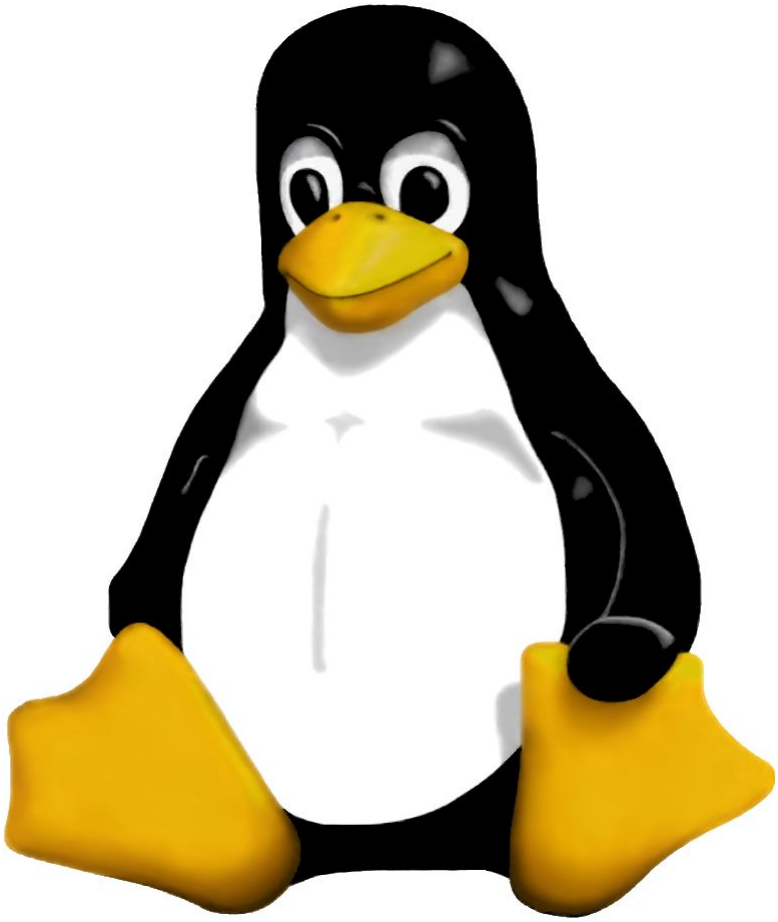
Llegados a este punto sería bueno plantearse la siguiente pregunta, ¿Qué distribución es la más adecuada para mi? Bueno, eso es la eterna pregunta, depende de las ganas, la ambición y la paciencia de cada uno. Para dar un poco de luz a esa oscura decisión, es recomendable visitar esta web, donde compara las características de cada distribución **http://www.distrowatch.com/** .

Como consejo personal, voy a exponer unas cuantas distribuciones ordenadas por mi criterio de dificultat, ese criterio es muy general y contempla desde instalación hasta gestión de software, por lo que no es un criterio demasiado fino. La lista va de más “fácil” a más “difícil”.

Nombre.	***Web.***
Knoppix.	http://www.knopper.net/knoppix/index-en.html
Mandrake.	http://www.mandrakelinux.com/
Red Hat.	http://www.redhat.com/
Suse.	http://www.suse.com/

NOTA: TODAS las distribuciones de Linux NO comerciales se pueden descargar en : **http://www.linuxiso.org/**

Curiosidad:
"But what ... is it good for?" / "Pero ... ¿Para que nos puede servir esto?"
Ingeniero en la Advanced Computing Systems Division de IBM, 1968, opinando sobre el microchip.

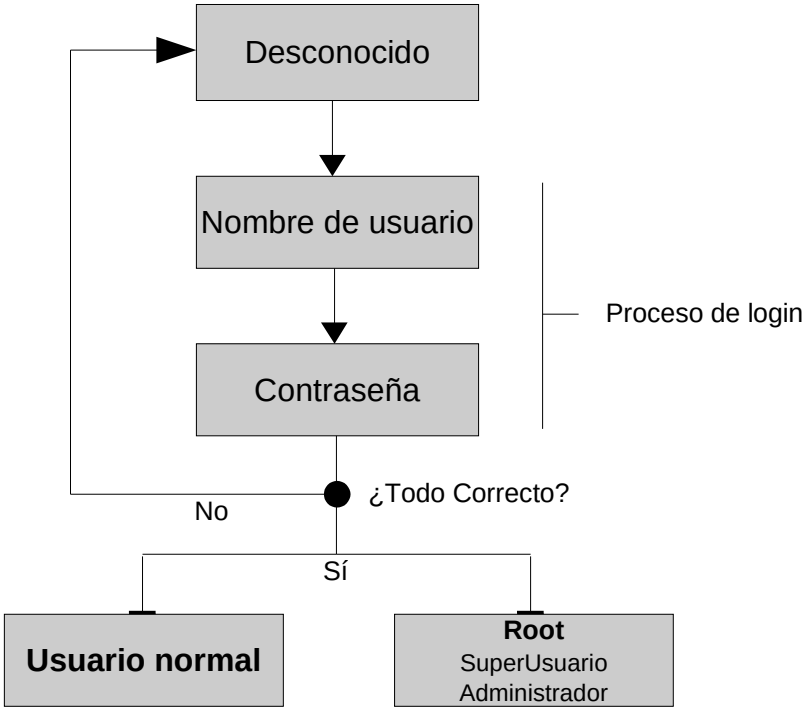


Tema 3.
El Uso del Sistema.

http://tira.escomposlinux.org/

3.1 ¿Quiénes somos para Linux?

Linux como sistema operativo, que pretende dar seguridad, no se fía de nadie, por lo que si no demostramos que somos "amigos", no nos deja acceder a él. Una vez **identificados** (*nombre de usuario*) y **autentificados** (*código de acceso*) Linux nos permite acceder a ciertos recursos[1] en función de quién seamos, es decir, si nos hemos identificado como usuario sin proviliegios (usuario normal), no podremos retocar la configuración "importante" del sistema, pero si nos hemos identificado como *root* o superusuario, linux nos deja acceder a todos los recursos sin límite.



- Usuario normal, sin privilegios.

La razón de ser de este tipo de usuario es la siguiente, él ha de hacer su tarea, sin preocuparse de si daña al sistema o a los demás usuarios, los privilegios se los otorga el *root*, en función de la tarea desenpeña.

- Root.

El *superusuario* o *root*, tiene plenos poderes dentro del sistema operativo, se supone que es él el administrador y último responsable del estado del sistema, por lo que obviamente, ha de tener acceso a todo, ya sea para bien o para mal.

- ¿Cómo se si puedo usar este recurso?

En líneas generales basta con ver qué *permisos* tiene ese recurso, este concepto se ve más adelante, o si el mismo programa nos da alguna advertencia del tipo: *faltan privilegios* o *no puedo acceder a este recurso.*

1 Recurso : Todo aquello que se puede usar dentro del sistema. Por ejemplo : ficheros, dispositivos, etc...

- Algunas operaciones del root.

El root puede instalar software dentro del sistema, puede desinstalarlo, modificarlo, actualizarlo, en definitiva no tiene límite dentro del campo de la gestión del sistema. Respecto a los usuarios normales, a éstos les puede abrir cuentas, permitir que accedan a recursos determinados, o impedirlo, borrar cuentas de usuario...

El root es el único usuario que puede apagar, reiniciar o suspender el equipo.

- ¿Cómo le decimos a linux que somos el root?

Si ya estamos identificados (proceso de login) como usuario normal, hemos de teclear en la consola , "su" :

```
Usuario@MiPC/$ su
Nos preguntará el password de root
root@MiPC/$
```

NOTA : Si esto no se entiende, no hay problema, la aclaración de qué es la consola o de cómo interactuamos con ella, se trata en las próximas páginas.

Inciso de vocabulario.

Palabra	***Definición***
Fichero	Se refiere a cualquier documento creado por el usuario, también engloba lo que serían aplicaciones en otros sistemas operativos.
Directorio	También llamado "*Carpeta*" por otros sistemas operativos, se utilizada para ordenar de forma inteligible para el usuario los ficheros del sistema.

3.2 Métodos de interacción con el sistema.

En Linux hay dos maneras de interactuar con el sistema, la primera es mediante la consola o shell, con órdenes escritas, este método permite tener el control absoluto sobre el sistema, ya que la shell nos da una colección de órdenes muy amplia para gestionarlo. La segunda manera es mediante un entorno gráfico, donde vamos de ventana en ventana modificando las opciones que deseamos. Como Linux inicialmente no contaba con entorno gráfico, a día de hoy, puede que hayan ciertas operaciones que no se puedan realizar desde dicho entorno, por lo que hace necesario dominar la shell.

3.2.1 Shell.

Veámos qué es la shell :

```
Usuario@MiPC:/$
```

Pues ciertamente no es gran cosa, pero nos permite hacer de todo. Ahora veámos qué infromación hay oculta en esa línea. Para verlo más claro, cambiaremos un poco la situación.

```
Usuario@MiPC:/usr/share/openoffice$
```

Usuario: Quién soy. Nos dice con que nombre de usuario estamos indentificados dentro del sistema.

MiPC: Dónde estoy. Nos dice en que grupo de trabajo estamos, este nombre se define en la instalación del sistema operativo.

/usr/share/openoffice$ Dentro del entorno de trabajo MiPC, el directorio en el cual estamos.

Ahora que ya tenemos más clara la idea de lo que es la shell, expliquémos algunos “trucos” para su mejor uso.

- En Linux **HAY** distinción entre mayúsculas y minúsculas, es decir no es lo mismo llamar a la orden ls que a la orden LS, de hecho LS no existe, lo mismo pasa con los nombres de los ficheros, directorios,etc...
- Los nombres de ficheros no tienen extensión como en otros sistemas operativos, esto se verá con más detalle en el tema de *SISTEMA DE FICHEROS.*
- La Shell nos ayuda a no escribir demasiado mediante la tecla TAB:

por ejemplo, veámos este directorio:

```
Usuario@MiPC:/home/prueba$ ls -la
total 12
drwxr-sr-x    2 usuario  staff       4096 2004-07-18 18:33 .
drwxrwsr-x   10 root     staff       4096 2004-07-18 18:33 ..
-rw-r--r--    1 usuario  staff         32 2004-07-18 18:33 FicheroConNombreLargoQueDaPerezaEscribir
Usuario@MiPC:/home/prueba$
```

Supongamos que queremos ver que hay dentro del fichero “FicheroConNombreLargoQueDaPerezaEscribir”, para hacer esto usaremos la orden “*cat*”, comentada más adelante, pues la orden sería del estilo **cat “*nombre del fichero*”**, claro que deberíamos escribir ese nombre de muchos carácteres, pero la shell permite hacer eso con pocas pulsaciones de tecla,

escribiendo lo siguiente:

```
Usuario@MiPC:/home/prueba$ cat F<Pulsamos TAB>
```

y automáticamente obtenemos :

```
Usuario@MiPC:/home/prueba$ cat FicheroConNombreLargoQueDaPerezaEscribir
```

Lo que ha pasado es que si pulsamos *TAB* en la shell, ésta intenta rellenar la línea de comandos con el nombre del fichero que empiece con las letras que ya hemos introducido. En este caso, como en el directorio prueba sólo hay un fichero que empiece con F ya es suficente información. Si pulsamos intro obtenemos:

```
Usuario@MiPC:/home/prueba$ cat FicheroConNombreLargoQueDaPerezaEscribir

¡¡¡ Felicidades has podido acceder al fichero usando la tecla TAB !!!

Usuario@MiPC:/home/prueba$
```

Curiosidad:
"640K de RAM deberían ser suficientes para todo el mundo."
Bill Gates, 1981.

3.2.1.1 Comandos Básicos en la Shell.

Una vez ya nos sabemos orientar en la consola, es hora de aprender a hacer cosas realmente útiles, la forma de “hacer cosas” con la shell es mediante órdenes escritas que nos devuelven el resultado por la misma consola, en definitiva la idea es ordenar y ver el resultado.

Las órdenes que vamos a ver se pueden agrupar de la siguiente forma:

- ✔ Navegación por el sistema de ficheros.
- ✔ Operaciones sobre ficheros.
- ✔ Varios.

La metodología utilizada para documentar cada órden es la siguiente : Comentario de la órden y de sus opciones y al lado de ésta, un extracto de la ayuda (*man*) de Linux sobre ese comando.

La ayuda *man*, se comenta en el capítulo de Ayuda propia de Linux, pero como adelanto decir que es un comando parecido al *help* de otros sistemas operativos, *man,* nos saca por pantalla la información relativa a una órden shell en concreto. Decir que *man* es una herramienta fundamental para todo usuario de Linux.

Navegación por el sistema de ficheros.

En este apartado veremos cómo podemos listar el contenido de un directorio, acceder a directorios, borrar la pantalla, copiar, mover ficheros y crear directorios.

ls (**L**i**S**t)

Descripción:

La orden ls se utiliza para listar el contenido de un directorio.

La sintaxi es:

ls -*opciones* [Directorio_a_Listar].

Donde lo contenido en [ ] es OPCIONAL, de hecho las opciones no son obligatorias pero son de un uso extremandamente frecuente.

La forma más común de uso de la órden ls es la siguiente:

ls -la

Proporciona el listado de todos los ficheros del directorio actual, incluso los ocultos y propios del sistema, además aportando información relevante sobre ellos como los permisos,propietario,etc... En el capítulo sobre sistema de ficheros se estudiará con más detalle la información relativa a los permisos.

LS(1) LS(1)

NOMBRE

ls, dir, vdir - listan los contenidos de directorios

SINOPSIS

ls [opciones] [fichero...]
dir [fichero...]
vdir [fichero...]
Opciones de POSIX: [-CFRacdilqrtu1]
Opciones de GNU (en la forma más corta): [-1abcdfghiklmnopqrstuvwxABCD-FGHLNQRSUX] [-w cols] [-T cols] [-I patrón] [--full-time] [--show-con-trol-chars] [--block-size=tamaño] [--format={long,verbose,com-mas,across,vertical,single-column}] [--sort={none,time,size,extension}] [--time={atime,access,use,ctime,status}] [--color[={none,auto,always}]] [--help] [--version] [--]

DESCRIPCIÓN

{OMITIDA para verla usar : *man ls* en la consola}

OPCIONES DE POSIX

-F Añade tras cada nombre de directorio un `/', tras cada nombre de FIFO un `|', y tras cada nombre de un ejecutable un `*'.

-R Lista recursivamente los subdirectorios encontrados.

-a Incluye en el listado ficheros cuyos nombres empiecen por `.'.

-d Lista nombres de directorios como otros ficheros, en vez de listar sus contenidos.

-l Escribe (en formato de una sola columna) los permisos del fichero, el número de enlaces que tiene, el nombre del propietario, el del grupo al que pertenece, el tamaño (en bytes), una marca de tiempo, y el nombre del fichero. De forma predeterminada, la marca de tiempo que se muestra es la de la última modificación; las opciones -c y -u seleccionan las otras dos que hay. Para ficheros especiales de dispositivo el campo de tamaño se reemplaza comúnmente por los números de dispositivo mayor y menor.

-r Invierte el orden de la clasificación.

-1 Para la salida en una sola columna.

Esto es un extracto del man, hay mucha más documentación disponible.

Ejemplos de comando ls:

```
Usuario@MiPC $:/home/prueba# ls
coches  doom  FicheroConNombreLargoQueDaPerezaEscribir  leeme.txt  quake
Usuario@MiPC $:/home/prueba#
```

Se puede ver que *ls* a secas, simplemente nos lista qué hay en el directorio sin más información.

```
Usuario@MiPC $:/home/prueba# ls -la
total 28
d rwx r-sr-x    3 Usuario    staff     4096 2004-07-19 18:34  .
d rwx rwsr-x   10 root       staff     4096 2004-07-18 18:33  ..
d rwx r-sr-x    2 root       staff     4096 2004-07-19 18:34  coches
-r rx rwx       1 root       staff        8  2004-07-19 18:30  doom
-rw-r--r--      1 Usuario    staff       32  2004-07-18 18:33  FicheroConNombreLargoQueDaPerezaEscribir
-rw-r--r--      1 root       staff       32  2004-07-19 18:34  leeme.txt
-rwxr-xr-x      1 root       staff        8  2004-07-19 18:34  quake
Usuario@MiPC $:/home/prueba#
```

Con las opciones -*la* ya podemos ver con más detalle lo que hay en el directorio /home/prueba. De nuevo la explicación de toda esta información, se verá en el capítulo de sistema de ficheros.

cd (**C**hange **D**irectory)

Esta órden no viene provista de ayuda en el man de linux.

Descripción:

cd permite cambiar de directorio.

Sintaxi:

cd *directorio_destino*

Ejemplos:

cd /home/Usuario

cd .. nos permite acceder al directorio padre o anterior al directorio actual.

cd prueba

Ejemplos :

```
Usuario@MiPC $:/ pwd
/
Usuario@MiPC $:/
```

```
Usuario@MiPC $: pwd
/usr/share/openoffice/bin
Usuario@MiPC $:
```

pwd (**P**rint **W**orking **D**irectory)

Descripción:

Nos dice en qué directorio estamos actualmente

Sintaxi:

pwd

```
PWD(1)                  User Commands                  PWD(1)
NAME
       pwd - print name of current/working directory
SYNOPSIS
       pwd [OPTION]
DESCRIPTION
       NOTE:  your  shell may have its own version of pwd which will supercede the
version described here. Please refer to your shell's  documentation for details
about the options it supports.
       Print the full filename of the current working directory.
       --help display this help and exit
       --version
              output version information and exit
 AUTHOR
       Written by Jim Meyering.
 SEE ALSO
       The  full  documentation for pwd is maintained as a Texinfo manual.  If  the i
nfo and pwd programs are properly installed at your site, the com mand:
              info pwd
       should give you access to the complete manual.
pwd 5.0.91                October 2003                 PWD(1)
```

clear

Descripción:

Borra la pantalla de la shell dejándola libre de respuestas anteriores.

Sintaxi:

clear

```
clear(1)                                          clear(1)
NAME
       clear - clear the terminal screen
SYNOPSIS
       clear
DESCRIPTION
       clear clears your screen if this is possible.  It looks in the environment for the
terminal type and then in the terminfo database to  figure out how to clear the
screen.
 SEE ALSO
       tput(1), terminfo(5)
                                                  clear(1)
```

Operaciones sobre ficheros.

cp (**C**o**P**y)

Descripción:

cp sirve para copiar un fichero desde su ubicación actual a otra, o incluso duplicarlo en el mismo directorio cambiándole el nombre.

Sintaxi:

cp -opciones *origen destino*

Donde:

origen es el nombre del fichero a copiar, que puede incluir la ruta de acceso a él o no, y *destino* es el nombre que se le dará al fichero “copia de origen”.

Ejemplos:

```
cp fichA.txt  fichB.txt
```

Esto duplica fichA.txt en el mismo directorio donde está, pero a la copia se le llama fichB.txt.

```
cp /home/Usuario/fichA.txt  fichB.txt
```

Esto copia el fichero /home/Usuario/fichA.txt, al directorio actual , a la copia se le llama fichB.txt. **Atención**, los dos ficheros están en directorios diferentes por lo que a la copia, llamada fichB.txt, se le podría llamar perfectamente fichA.txt.

CP(1) CP(1)

NOMBRE

cp - copia ficheros y directorios

SINOPSIS

cp [opciones] fichero camino
cp [opciones] fichero... directorio
Opciones de POSIX: [-fipRr]
Opciones de GNU (en la forma más corta): [-abdfilprsuvxPR] [-S SUFIJO]
[-V {numbered,existing,simple}] [--sparse=CUANDO] [--help] [--version]
[--]

DESCRIPCIÓN

cp copia ficheros (o, opcionalmente, directorios). Uno puede bien copiar un fichero a un destino dado, o copiar arbitrariamente varios ficheros a un directorio destino.

Si el último argumento se refiere a un directorio existente, cp copia cada fichero fuente a ese directorio (manteniendo el mismo nombre). En otro caso, si sólo se dan dos ficheros, copia el primero sobre el segundo. Es un error que el último argumento no sea un directorio y se den varios argumentos no opciones. (Así por ejemplo, `cp -r /a /b' copiará /a a /b/a y /a/x a /b/a/x en caso de que /b ya exista, pero copiará /a a /b y /a/x a /b/x si no existía /b con anterioridad.)

Los permisos de los ficheros y directorios creados serán los mismos que los de los ficheros originales, aplicándoles la operación de bits Y sobre 0777, y modificados por la umask del usuario (a menos que se haya especificado la opción -p). (Pero durante la copia recursiva de directorios, a los permisos finales de los directorios recién creados se les aplicará la operación de bits O con S_IRWXU (0777), de forma que se permita al proceso leer, escribir y pasar por el directorio recién creado.)

OPCIONES DE POSIX

-f Borrar ficheros destino existentes si se requiere. (Vea más arriba.)

-i Pregunta si sobreescribir ficheros regulares destino existentes. (Escribe una pregunta en stderr, y lee la respuesta desde stdin. Sólo copia tras una respuesta afirmativa.)

-p Preserva los permisos, el propietario y el grupo (incluyendo los bits SUID y SGID) de los ficheros originales, más el tiempo de última modificación y el de último acceso. En caso de que la duplicación del propietario o grupo falle, se limpian los bits setuid y setgid. (Observe que después de todo el fuente y la copia pueden muy bien tener tiempos de último acceso diferentes, puesto que la operación de copia es un acceso al fichero fuente.)

-R Copia directorios recursivamente, y hace lo correcto cuando se encuentran objetos distintos de ficheros ordinarios o directo rios. (Así, la copia de un FIFO un un fichero especial es un FIFO o un fichero especial.)

-r Copia directorios recursivamente, y hace algo sin especificar con objetos distintos de ficheros ordinarios o directorios.

(Así, está permitido, de hecho recomendado, que la opción -r sea un sinónimo de -R. Sin embargo, un comportamiento tonto, como el de la presente versión de GNU de cp (vea más abajo) no está prohibido.)

OPCIONES DE GNU

-i, --interactive
Pregunta si sobreescribir ficheros de destino regulares existentes.

-l, --link
En vez hacer copias de ficheros que no son directorios, hace enlaces duros.

-p, --preserve
Preserva los permisos, el propietario, el grupo y los tiempos de los ficheros originales.

OBSERVACIONES

Esta página describe cp según se encuentra en el paquete fileutils-4.0; otras versiones pueden diferir un poco. Envíe por correo electrónico
correcciones y adiciones a la dirección aeb@cwi.nl. Informe de fallos en el programa a fileutils-bugs@gnu.ai.mit.edu.

GNU fileutils 4.0 Noviembre 1998 CP(1)

```
Usuario@MiPC:/home/prueba2$ cp -r /home/prueba /home/prueba2/
```

Esto copia el directorio /home/prueba, incluyendo subdirectorios, en el directorio /home/prueba2 .

Nota : dentro de prueba2 nos encontramos con lo siguiente después de hacer cp.

```
Usuario@MiPC:/home/prueba2$ ls -la
total 12
drwxr-sr-x     3 Usuario   staff       4096 2004-07-21 20:14 .
drwxrwsr-x    11 root      staff       4096 2004-07-21 20:13 ..
drwxr-sr-x     3 Usuario   staff       4096 2004-07-21 20:14 prueba
```

es decir, ha creado el directorio origen (prueba) y dentro está lo que contenía /home/prueba.

mv (**Mo**v**e**)

Descripción:

mv sirve para cambiar de directorio un fichero.

Sintaxi:

mv -opciones *origen destino*

Donde:

origen es el nombre del fichero a mover, que puede incluir la ruta de acceso a él o no, y *destino* es el nombre que se le dará al fichero, en este caso ese nombre puede incluir o no el directorio destino.

mv también puede ser usado para renombrar ficheros, es decir para hacer un *rename* , órden que se usa en otros sistemas operativos.

MV(1) MV(1)
NOMBRE
mv - mueve (renombra) ficheros
SINOPSIS
mv [opción...] origen destino
mv [opción...] origen... destino
Opciones de POSIX: [-fi]
Opciones de GNU (en la forma más corta): [-bfiuv] [-S sufijo] [-V {numbered,existing,simple}] [--help] [--version]
[--]

DESCRIPCIÓN
mv mueve o renombra ficheros o directorios.
Si el último argumento nombra a un directorio existente, mv mueve cada uno de los otros ficheros a un fichero con el mismo nombre en ese directorio. Si no, si sólo se dan dos ficheros, renombra el primero al segundo. Es un error que el último argumento no sea un directorio y se den más de dos ficheros.
Así, `mv /a/x/y /b' renombraría el fichero /a/x/y a /b/y si /b fuera un directorio existente, y a /b si no lo fuera.
Llamemos destino al fichero al cual se va a mover un fichero dado. Si destino existe, y o bien se ha dado la opción -i o bien destino no es modificable, y la entrada estándar es una terminal, y no se ha dado la opción -f, mv pregunta al usuario si quiere reemplazar el fichero, escribiendo una pregunta en la salida estándar de errores (stderr) y leyendo una respuesta desde la entrada estándar (stdin). Si la respuesta no es afirmativa, se salta ese fichero.
Cuando tanto origen como destino están en el mismo sistema de ficheros, son el mismo fichero (sólo el nombre se cambia; el propietario, permisos y marcas de tiempo permanecen intactos). Cuando están en sistemas de ficheros diferentes, el fichero origen se copia con el nuevo nombre y luego se borra. mv copiará el tiempo de modifi cación, el tiempo de acceso, el identificador del propietario y del grupo, y los permisos, si puede. Cuando la copia del ID del propietario o del grupo falle, los bits setuid y setgid se limpian en la copia.
OPCIONES DE POSIX
-f No pide confirmación.
-i Pide confirmación cuando destino existe. (En caso de que se den -f y -i, la última opción dada es la que tiene efecto.)
DETALLES DE GNU
La implementación de GNU falla (en fileutils-3.16) en el sentido de que mv sólo puede mover ficheros regulares entre sistemas de ficheros distintos.
OPCIONES DE GNU
-f, --force
Borra los ficheros de destino existentes sin preguntar nunca al usuario.
-i, --interactive
Pregunta si se desean sobreescribir ficheros de destino regulares existentes. Si la respuesta no es afirmativa, se pasa al siguiente fichero sin efectuar la operación.
-u, --update
No mueve un fichero no directorio que tenga un destino existente con el mismo tiempo de modificación o más reciente.
-v, --verbose
Muestra el nombre de cada fichero antes de moverlo.

Ejemplos:

```
mv fichA.txt  /home/Textos/fichA.txt
```

Movemos el fichero fichA.txt al directorio /home/Textos.

```
mv /home/Usuario/fichA.txt  fichA.txt
```

Esto mueve el fichero /home/Usuario/fichA.txt, al directorio actual.

Cambiar el nombre del fichero.

```
mv fichA.txt  fichB.txt
```

A partir de ahora fichA.txt se llama fichB.txt.

```
mv /home/Usuario/fichA.txt  fichB.txt
```

Esto mueve el fichero /home/Usuario/fichA.txt, al directorio actual y se le cambia el nombre, ahora el fichA.txt se llamará fichB.txt

```
mv /home/Usuario/fichA.txt  /home/Usuario/Algo/fichA.txt
```

Esto mueve el fichero /home/Usuario/fichA.txt, al directorio /home/Usuario/Algo/.

Curiosidad:
El Primer virus informático se creó el 3 de noviembre de 1983, por un estudiante de la universidad de Carolina del Sud y fue presentado como un experimento para un seminario sobre seguridad informática. Tardó en crearlo 8 horas y funcionaba bajo un VAX 11/750 con UNIX.

rm (**Re**M**ove**)

Descripción:

rm sirve para eliminar ficheros o directorios.

NOTA IMPORTANTE:

Usar esta órden siendo *root*, es altamente peligroso, ya que sin querer podemos borrar ficheros del sistema, inutilizándolo.

Consejo: **Borrar siempre como *usuario normal*** y cuando sea necesario y se pueda, borrar como *root* los ficheros de uno en uno.

Sintaxi: rm -opciones *objetivo*

Donde:

objetivo es el fichero o directorio que queremos eliminar.

```
RM(1)                                                    RM(1)
 NOMBRE
      rm - borra ficheros o directorios
 SINOPSIS
      rm [opciones] fichero...
      Opciones de POSIX: [-fiRr]
      Opciones de GNU (en la forma más corta): [-dfirvR] [--help] [--version] [--]
 DESCRIPCIÓN
       rm  borra  cada fichero dado.  Por lo normal, no borra directorios.  Pero cuando se da la
opción -r o -R, se borra el árbol de directorios entero a partir del directorio especificado (y sin
limitaciones en cuanto a la profundidad  de los árboles de directorio que pueden borrarse con `rm
-r').  Es un error que el último componente del camino de fichero sea . o ..  (para evitar así
sorpresas desagradables con `rm -r .*' o así).
      Si se da la opción -i, o si un fichero no es modificable, y la entrada estándar es una terminal, y
la opción -f no se  ha  dado,  rm pregunta al usuario si quiere borrar realmente el fichero,
escribiendo una pregunta en la salida estándar de errores y leyendo una respuesta desde la
entrada estándar.  Si  la  respuesta  no  es  afirmativa,  el fichero no se borra y se pasa al
siguiente.
 OPCIONES DE POSIX
      -f     No  pide  confirmación.  No escribe mensajes de diagnóstico. No produce un estado de
salida de error si los  únicos errores han sido ficheros que no existen.
      -i     Pide confirmación.  (En el caso de que se den tanto -f como -i, el último que se escriba es
el  que  tiene  efecto.)
      -r or -R
             Borra recursivamente árboles de directorio.
 OBSERVACIONES
       Esta página describe rm según se encuentra en el paquete fileutils-4.0; otras versiones
pueden  diferir  un  poco.
      Envíe por correo electrónico correcciones y adiciones a la dirección aeb@cwi.nl.  Informe de
fallos en el programa
      a fileutils-bugs@gnu.ai.mit.edu.

                GNU fileutils 4.0                Noviembre de 1998
```

Ejemplos:

rm fichA.txt
Borramos el fichero fichA.txt del directorio actual.

rm /home/Usuario/fichA.txt
Borramos el fichero fichA.txt del directorio /home/Usuario/

Ejemplo de eliminación de un directorio con todo lo que contenga, incluídos subdirectorios.

Sintaxi : rm -r *objetivo*

```
Usuario@MiPC:/home/prueba$ tree                    #Nos dibuja el "árbol de directorios", para ver que hay dentro
                                                   # del directorio prueba. Similar a la órden ls pero más gráfica.
|-- ayuda.txt                                      # Para más información sobre tree, man tree.
|-- IconoSoft.ico
`-- Catalogo                                       # Subdirectorio que contiene el catálogo de coches.
   |-- FicheroConNombreLargoQueDaPerezaEscribir
   |-- coches                                      # Subdirectorio donde se guardan datos de las marcas.
   |   |-- BMW.car
   |   |-- Delorean.car
   |   `-- Ferrari.car
   |-- readme.txt
   `-- Micatalog
2 directories, 8 files

Usuario@MiPC:/home/prueba$ rm -r Catalogo/         # Procedemos a borrar el catálogo con todo su contenido.
Usuario@MiPC:/home/prueba$ tree
.
`-- ayuda.txt                                      # Ya no aparece el catálogo, sí aparecen los ficheros que NO
`-- IconoSoft.ico                                  # estaban dentro de Catálogo/
0 directories, 2 files

Usuario@MiPC:/home/prueba$ ls
ayuda.txt  IconoSoft.ico
Usuario@MiPC:/home/prueba$
```

mkdir (**Ma**K**e** **DIR**ectory)

Descripción:

mkdir permite crear directorios.

Sintaxi:

mkdir -opciones *directorio.*

Donde:

directorio es el nombre que le daremos al directorio que queremos crear.

Ejemplos:

```
mkdir prueba
```

Creamos el directorio prueba dentro del directorio actual.

```
mkdir /home/Usuario/Doc/Dibujos
```

Creamos el directorio Dibujos dentro de /home/Usuario/Doc

```
MKDIR(1)                                MKDIR(1)
NOMBRE
    mkdir - crea directorios
SINOPSIS
    mkdir [opciones] directorio...
    Opciones de POSIX: [-p] [-m modo]
    Opciones de GNU (en la forma más corta): [-p] [-m modo] [--verbose]
    [--help] [--version] [--]
DESCRIPCIÓN
    mkdir crea directorios con los nombres especificados.
    De forma predeterminada, los permisos de los directorios creados son
    0777 (`a+rwx') menos los bits puestos a 1 en la umask.
OPCIONES
    -m modo, --mode=modo
        Establece los permisos de los directorios creados a modo, que puede ser simbólico
como en chmod(1) y entonces emplea el modo predeterminado como el punto de partida.
    -p, --parents
        Crea los directorios padre que falten para cada argumento directorio. Los permisos para
los directorios padre se ponen a la umask modificada por `u+rwx'. No hace caso de
rgumentos que correspondan a directorios existentes. (Así, si existe un directorio /a, entonces
`mkdir /a' es un error, pero `mkdir -p /a' no lo es.)
    --verbose
        Muestra un mensaje para cada directorio creado. Esto es más útil con –parents.
OBSERVACIONES
    Esta página describe mkdir según se encuentra en el paquete file tils-4.0; otras versiones
pueden diferir un poco. Envíe por correo electrónico correcciones y adiciones a la dirección
aeb@cwi.nl. Informe de fallos en el programa a fileutils-bugs@gnu.ai.mit.edu.
        GNU fileutils 4.0        Noviembre de 1998        MKDIR(1)
```

rmdir (**ReMove** **DIR**ectory)

Descripción:

rmdir permite borrar un directorio que ha sido préviamente vaciado.

Sintaxi: rmdir -opciones *directorio.*

Donde:

directorio es el nombre del directorio que deseamos suprimir.

Ejemplos:

```
rmdir prueba
```

Borramos el directorio prueba dentro del directorio actual.

```
rmdir /home/Usuario/Doc/Dibujos
```

Borramos el directorio Dibujos dentro de /home/Usuario/Doc

```
RMDIR(1)                                RMDIR(1)
NOMBRE
    rmdir - borra directorios vacíos
SINOPSIS
    rmdir [opciones] directorio...
    Opciones de POSIX: [-p]
    Opciones de GNU (en la forma más corta): [-p] [--ignore-fail-on-non-empty] [--help] [--version]
[--]
DESCRIPCIÓN
    rmdir borra directorios vacíos.
    Si un argumento directorio no se refiere a un directorio existente y vacío, es un error.
OPCIONES DE POSIX
    -p  Si directorio incluye más de un componente en el camino, lo borra, luego quita el último
componente y borra el directorio resultante, etc., hasta que todos los componentes hayan sido
eliminados. Así, `rmdir -p a/b/c' es equivalente a `rmdir a/b/c; rmdir a/b; rmdir a'.
OPCIONES DE GNU
    --ignore-fail-on-non-empty
        Normalmente, rmdir rehusará elminar un directorio que no está vacío. Esta opción hace
que rmdir ignore el
        fallo para eliminar el directorio, si ese fallo se debe a que el directorio no está vacío.
(Nueva en fileutils-4.0.)
    -p, --parents
        Como se acaba de explicar arriba.
CONFORME A
    POSIX 1003.2.
EJEMPLO DE UTILIZACIÓN
    La orden `rmdir fuu' borrará el directorio fuu si está vacío. Para borrar un directorio no vacío,
junto con todo lo que tenga debajo, emplee `rm -r fuu'.
OBSERVACIONES
    Esta página describe rmdir según se encuentra en el paquete fileutils-4.0; otras versiones
pueden diferir un poco. Envíe por correo electrónico correcciones y adiciones a la dirección
aeb@cwi.nl. Informe de fallos en el programa a fileutils-bugs@gnu.ai.mit.edu.
    GNU fileutils 4.0        Noviembre de 1998        RMDIR(1)
```

Error común.

```
Usuario@MiPC:/home/prueba# ls                    #Miramos el contenido del directorio.
leeme.txt
Usuario@MiPC:/home/prueba# cd ..                 # Retrocedemos.
Usuario@MiPC:/home# rmdir prueba                 # Borramos
rmdir: `prueba': El directorio no está vacío     # ¡¡¡ ERROR !!!
Usuario@MiPC:/home#                              # Hemos de borrar el directorio con la órden rm antes de usar rmdir.
```

Comandos varios.

more.

Descripción:

more nos permite ver el contenido de un fichero, ese fichero debería ser un fichero con contenido ascii para poder verlo.

Sintaxi:

more -opciones *nombre.*

Donde:

nombre es el nombre del fichero a visualizar.

Particularidades:

more tiene varios modos de avance, uno es mediante la tecla espacio, avanza de página en página, la tecla intro permite ir línea a línea. Notar que more sólo permite avanzar en un fichero, no permite ver líneas ya visualizadas.

```
MORE(1)          BSD General Commands Manual          MORE(1)
NAME
    more - file perusal filter for crt viewing
SYNOPSIS
    more [-dlfpcsu] [-num] [+/ pattern] [+ linenum] [file ...]
DESCRIPTION
    More is a filter for paging through text one screenful at a time.  This version is especially
primitve.  Users should realize that less(1) pro vides more(1) emulation and extensive
enhancements.
COMMANDS
    Interactive commands for more are based on vi(1).  Some commands may be preceded by a
decimal number, called k in the descriptions below.  In the following descriptions, ^X means
control-X.
    h or ?      Help: display a summary of these commands.  If you forget all the other commands,
remember this one.
    SPACE       Display next k lines of text.  Defaults to current screen   size.
    z           Display next k lines of text.  Defaults to current screen  size.  Argument becomes new
default.
    RETURN      Display next k lines of text.  Defaults to 1.  Argument  becomes new default.
    d or ^D     Scroll k lines.  Default is current scroll size, initially  11.  Argument becomes new
default.
    q or Q or INTERRUPT    Exit.
    s           Skip forward k lines of text.  Defaults to 1.
    f           Skip forward k screenfuls of text.  Defaults to 1.
    b or ^B     Skip backwards k screenfuls of text.  Defaults to 1.  Only   works with files, not pipes.
    '           Go to place where previous search started.
    =           Display current line number.
    /pattern    Search for kth occurrence of regular expression.
    n           Search for kth occurrence of last r.e.  Defaults to 1
SEE ALSO
    vi(1) less(1)
```

Cuando more esté funcionando si escribimos dos puntos (:) entramos en el modo inserción de comandos, es decir, ahora more nos presta atención para que le digamos qué hacer, sobre la visualización del fichero.

Algunos comandos son :

- q , cerrará more.
- /cadenaDeBúsqueda , more buscará por el texto de pantalla la “cadenaDeBúsqueda.”

Ejemplos:

```
more prueba
more /home/Usuario/Doc/Dibujos/leeme.txt
```

Curiosidad:
"Todo lo que se pueda inventar, ya ha sido inventado”
Charles H. Duell, Comisionado de la Oficina de patentes de EEUU. 1899.

less.

Descripción:

less es un programa parecido a more, pero con muchas más opciones y funcionalidades, pero en esencia, se usa para visualizar ficheros que con more no son visibles, por ejemplo, ficheros que no tienen contenido ascii , o ficheros muy grandes.

Sintaxi:

less -opciones *nombre.*

Donde:

nombre es el nombre del fichero a visualizar.

Particularidades:

Su uso básico es idéntico al de more, es decir, intro avanza línea a línea y espacio página a página, los dos puntos entran en modo inserción de comandos.

Comandos más usados:

LESS(1) LESS(1)

NAME

less - opposite of more

SYNOPSIS

less -?
less --help
less -V
less --version
less [-[+]aBcCdeEfFgGiIJLmMnNqQrRsSuUVwWX~]
[-b space] [-h lines] [-j line] [-k keyfile]
[-{oO} logfile] [-p pattern] [-P prompt] [-t tag]
[-T tagsfile] [-x tab,...] [-y lines] [-[z] lines]
[-# shift] [+[+]cmd] [--] [filename]...
(See the OPTIONS section for alternate option syntax with long option names.)

DESCRIPTION

Less is a program similar to more (1), but which allows backward movement in the file as well as forward movement.

Also, less does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like vi (1). Less uses termcap (or terminfo on some systems), so it can run on a variety of terminals. There is even limited support for hardcopy terminals. (On a hardcopy terminal, lines which should be printed at the top of the screen are prefixed with a caret.)

Commands

Commands are based on both more and vi. Commands may be preceded by a decimal number, called N in the descriptions below. The number is used by some commands, as indicated.

In the following descriptions, ^X means control-X. ESC stands for the ESCAPE key; for example ESC-v means the two character sequence "ESCAPE", then "v".

h or H Help: display a summary of these commands. If you forget all the other commands, remember this one.

SPACE or ^V or f or ^F

Scroll forward N lines, default one window (see option -z below). If N is more than the screen size, only the final screenful is displayed. Warning: some systems use ^V as a special literalization character.

z Like SPACE, but if N is specified, it becomes the new window size.

ESC-SPACE

Like SPACE, but scrolls a full screenful, even if it reaches end-of-file in the process.

RETURN or ^N or e or ^E or j or ^J

Scroll forward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size.

COPYRIGHT

Copyright (C) 2002 Mark Nudelman

AUTHOR

Mark Nudelman <markn@greenwoodsoftware.com>
Send bug reports or comments to the above address or to bug-less@gnu.org.
For more information, see the less homepage at http://www.greenwoodsoftware.com/less.

Entrada

Salida

q , cierra less.

%*número* , nos enseña la zona que corresponde a ese porcentage del documento, por ejemplo si ponemos **%90**, nos llevará a la línea del fichero que equivale al 90% del fichero, o a la línea a partir de la cual sólo queda un 10% de documento.

Ejemplos:

```
less prueba
less /home/Usuario/Doc/Dibujos/leeme.txt
```

cat.

Descripción:

cat vuelve a ser parecido a more, pero éste permite concatenar ficheros y sacarlos por pantalla.

Sintaxi:

cat -opciones *nombre1 nombre2 ... nombreN*

Donde:

nombreX es el nombre del fichero a visualizar y concatenar con el resto.

Particularidades:

La idea de cat es hacer un more de varios ficheros a la vez, de forma que se muestran según el órden impuesto en la llamada. Veámos unos ejemplos para aclarar esto.

Ejemplos:

```
cat prueba
```

Nos enseñará el contenido del fichero prueba.

```
cat /home/Usuario/Doc/Dibujos/leeme.txt
```

Nos enseñará el contenido del fichero leeme.txt.

```
CAT(1)                          User Commands
NAME
       cat - concatenate files and print on the standard output
SYNOPSIS
       cat [OPTION] [FILE]...
DESCRIPTION
       Concatenate FILE(s), or standard input, to standard output.
       -A, --show-all
              equivalent to -vET
       -b, --number-nonblank
              number nonblank output lines
       -e     equivalent to -vE
       -E, --show-ends
              display $ at end of each line
       -n, --number
              number all output lines
       -r, --reversible
              use \ to make the output reversible, implies -v
       -s, --squeeze-blank
              never more than one single blank line
       -t     equivalent to -vT
       -T, --show-tabs
              display TAB characters as ^I
       -u     (ignored)
       -v, --show-nonprinting
              use ^ and M- notation, except for LFD and TAB
       --help display this help and exit

       --version
              output version information and exit
       With no FILE, or when FILE is -, read standard input.
AUTHOR
       Written by Torbjorn Granlund and Richard M. Stallman.
REPORTING BUGS
       Report bugs to <bug-coreutils@gnu.org>.
COPYRIGHT
       Copyright (C) 2003 Free Software Foundation, Inc.
       This  is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY
       or FITNESS FOR A PARTICULAR PURPOSE.
cat (coreutils) 5.0.91                  October 2003
```

```
Usuario@MiPC:/home/prueba# ls                          #Vemos qué hay dentro de prueba
diario  leeme.txt                                      # Dos ficheros:  leeme.txt y diario
Usuario@MiPC:/home/prueba# cat leeme.txt  diario       #Intentamos visualizar los dos ficheros uno detrás de otro.
--------------------------------                       # Comienzo de leeme.txt
ESTE ES EL CONTENIDO DEL FICHERO
LEEME.TXT
--------------------------------
ABCDEFGHIJKLLLMNÑOPQRST
abcdefghijkllmnñopqrst
--final del fichero leeme.txt--                        #Fin de leeme.txt
--------------------------------                       #Comienzo de diario
ESTE ES EL CONTENIDO DEL FICHERO
diario
--------------------------------
DIA 12:
      Quiero ir.
DIA 16:
      Al final sí que fuí.
--final del fichero diario--                           #Fin de diario
Usuario@MiPC:/home/prueba#
```

date.

Descripción:

date nos muestra por pantalla la fecha actual y nos permite modficarla.

NOTA : Para poder modificar la hora del sistema, hemos de ser necesariamente *root*.

Sintaxi: date -opciones

para ver la fecha no se usa ninguna opción, para modificarla hay que hacer:

date -s *datos_de_la_nueva_fecha*

Ejemplos:

```
usuario@MiPC:/date
```

Esto nos muestra por pantalla la fecha actual del sistema.

```
usuario@MiPC:/home/# date -s 7/24/04
sáb jul 24 00:00:00 GMT 2004
```

Establece la fecha del sistema a 24 de Julio de 2004.

```
usuario@MiPC:/home/# date -s 12:00:45
sáb jul 24 12:00:45 GMT 2004
```

Establece la hora del sistema a las 12 horas 0 minutos y 45 segundos.

```
DATE(1)                User Commands                DATE(1)
NAME
       date - print or set the system date and time
SYNOPSIS
       date [OPTION]... [+FORMAT]
       date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
DESCRIPTION
       Display the current time in the given FORMAT, or set the system date.
       -d, --date=STRING
              display time described by STRING, not `now'
       -s, --set=STRING
              set time described by STRING
       FORMAT controls the output.  The only valid option for the second  form specifies
Coordinated Universal Time.  Interpreted sequences are:
       %%     a literal %
       %a     locale's abbreviated weekday name (Sun..Sat)
       %A     locale's full weekday name, variable length (Sunday..Saturday)
       %b     locale's abbreviated month name (Jan..Dec)
       %B     locale's full month name, variable length (January..December)
       %c     locale's date and time (Sat Nov 04 12:02:33 EST 1989)
       %d     day of month (01..31)
       %D     date (mm/dd/yy)
       %F     same as %Y-%m-%d
       %g     the 2-digit year corresponding to the %V week number
       %G     the 4-digit year corresponding to the %V week number
       %h     same as %b
       %H     hour (00..23)
       %I     hour ( 1..12)
       %m     month (01..12)
       %r     time, 12-hour (hh:mm:ss [AP]M)
       %R     time, 24-hour (hh:mm)
       %T     time, 24-hour (hh:mm:ss)
AUTHOR
       Written by David MacKenzie.
REPORTING BUGS
       Report bugs to <bug-coreutils@gnu.org>.
COPYRIGHT
       Copyright (C) 2003 Free Software Foundation, Inc.
       This is free software; see the source for copying conditions.  There is
              NO warranty; not even for MERCHANTABILITY or FITNESS FOR  A
PARTICULAR PURPOSE.
date (coreutils) 5.0.91        October 2003                DATE(1)
```

---

cal.

Descripción:

Nos muestra el calendario del año solicitado.

Sintaxi:

cal *mes año*

Ejemplos:

cal 2004 : nos enseña el calendario completo del año 2004.

cal 10 1701 : nos muestra Octubre de 1701.

```
CAL(1)                BSD General Commands Manual
     CAL(1)
NAME
     cal, ncal - displays a calendar and the date of easter
SYNOPSIS
     cal [-3jmy] [[month] year]
     ncal [-jJpwy] [-s country_code] [[month] year]
     ncal [-Jeo] [year]
DESCRIPTION
     The cal utility displays a simple calendar in traditional format and ncal offers an
alternative layout, more options  and the date of easter.  .  If arguments are not specified,
the current month is displayed.
     The options are as follows:
     -J      Display Julian Calendar, if combined with the -e option, display date of easter
according to the Julian Calendar.
     -e      Display date of easter (for western churches).
     -m      Print a calendar where Monday is the first day of the week, as opposed to
Sunday.
     -j      Display Julian days (days one-based, numbered from January 1).
     -o      Display date of orthodox easter (Greek and Russian Orthodox Churches).
     -p      Print the country codes and switching days from Julian to Gregorian Calendar as
they are assumed by ncal. The country code as determined from the local environment is
marked with an asterisk.
```

locate.

Descripción:

locate nos permite buscar un fichero/directorio dentro del sistema de ficheros.

Sintaxi:

locate -opciones *nombre*

Donde:

nombre es el nombre del fichero/directorio buscar.

Particularidades:

Cada vez que se crea un directorio o fichero, el conjunto de ficheros del sistema cambia, por lo que locate se ha de “enterar” de esos cambios, para ello existe el comando : *updatedb* que actualiza la base de datos del sistema de ficheros.

LOCATE(1L) LOCATE(1L)

NAME

locate - list files in databases that match a pattern

SYNOPSIS

locate [-d path | --database=path] [-e | --existing] [-i | --ignore-case] [--version] [--help] pattern...

DESCRIPTION

This manual page documents the GNU version of locate. For each given pattern, locate searches one or more databases of file names and displays the file names that contain the pattern. Patterns can contain shell-style metacharacters: `*', `?', and `[ ]'. The metacharacters do not treat `/' or `.' specially Therefore, a pattern `foo*bar' can match a file name that contains `foo3/bar', and a pattern `*duck*' can match a file name that contains `lake/.ducky'. Patterns that contain metacharacters should be quoted to protect them from expansion by the shell.

If a pattern is a plain string -- it contains no metacharacters -- locate displays all file names in the database that contain that string anywhere. If a pattern does contain metacharacters, locate only displays file names that match the pattern exactly. As a result, patterns that contain metacharacters should usually begin with a `*', and will most often end with one as well. The exceptions are patterns that are intended to explicitly match the beginning or end of a file name.

The file name databases contain lists of files that were on the system when the databases were last updated. The system administrator can choose the file name of the default database, the frequency with which the databases are updated, and the directories for which they contain entries; see updatedb(1L).

OPTIONS

-d path, --database=path

Instead of searching the default file name database, search the file name databases in path, which is a colon-separated list of database file names. You can also use the environment variable LOCATE_PATH to set the list of database files to search. The option overrides the environment variable if both are used.

The file name database format changed starting with GNU find and locate version 4.0 to allow machines with diffent byte orderings to share the databases. This version of locate can automatically recognize and read databases produced for older versions of GNU locate or Unix versions of locate or find.

-e, --existing

Only print out such names that currently exist (instead of such names that existed when the database was created). Note that this may slow down the program a lot, if there are many matches in the database.

-i, --ignore-case

Ignore case distinctions in both the pattern and the file names.

SEE ALSO

find(1L), locatedb(5L), updatedb(1L), xargs(1L) Finding Files (on-line in Info, or printed)

Caso Común que se da con locate.

```
Usuario@MiPC:/home/prueba$ ls > nombreNuevo.txt          # Creamos el fichero nombreNuevo.txt
Usuario@MiPC:/home/prueba$ locate nombreNuevo.txt        # Lo Buscamos

locate: atención: la base de datos `/var/cache/locate/locatedb' tiene una antigüedad de más de 8 días

Usuario@MiPC:/home/prueba$ updatedb                      # locate sugiere que actualicemos la Base de datos.

/usr/bin/find: /lost+found: Permiso denegado             # updatedb nos avisa que no podemos acceder a
/usr/bin/find: /root/.kde: Permiso denegado              # ciertos ficheros ya que NO somos root.
/usr/bin/find: /root/.gnome: Permiso denegado            #
/usr/bin/find: /root/.gnome_private: Permiso denegado    #
...
Usuario@MiPC:/home/prueba$ su                            # Nos identificamos como root
Password:
Root@MiPC:/home/prueba$ updatedb                         # Volvemos a actualizar la base de datos
Root@MiPC:/home/prueba$ exit                             # Dejamos de ser root
exit
Usuario@MiPC:/home/prueba$ locate nombreNuevo.txt        # Buscamos.
/home/prueba/nombreNuevo.txt                             #Nos indica dónde esta el fichero
Usuario@MiPC:/home/prueba$
```

reboot.

Descripción:

reboot se encarga de reiniciar el sistema.

Sintaxi:

reboot -opciones

Particularidades:

Para poder utilizar reboot hemos de estar identificados como root.

poweroff.

Descripción:

poweroff se encarga de apagar el sistema.

Sintaxi:

poweroff -opciones

Particularidades:

Para poder utilizar poweroff hemos de estar identificados como root.

HALT(8) Linux System Administrator's Manual HALT(8)

NAME

halt, reboot, poweroff - stop the system.

SYNOPSIS

/sbin/halt [-n] [-w] [-d] [-f] [-i] [-p] [-h]
/sbin/reboot [-n] [-w] [-d] [-f] [-i]
/sbin/poweroff [-n] [-w] [-d] [-f] [-i] [-h]

DESCRIPTION

Halt notes that the system is being brought down in the file /var/log/wtmp, and then either tells the kernel to halt, reboot or poweroff the system.

If halt or reboot is called when the system is not in runlevel 0 or 6, in other words when it's running normally, shutdown will be invoked instead (with the -h or -r flag). For more info see the shutdown(8) manpage.

The rest of this manpage describes the behaviour in runlevels 0 and 6, that is when the systems shutdown scripts are being run.

OPTIONS

-n Don't sync before reboot or halt.
-w Don't actually reboot or halt but only write the wtmp record (in the /var/log/wtmp file).
-d Don't write the wtmp record. The -n flag implies -d.
-f Force halt or reboot, don't call shutdown(8).
-i Shut down all network interfaces just before halt or reboot.
-h Put all harddrives on the system in standby mode just before halt or poweroff.
-p When halting the system, do a poweroff. This is the default when halt is called as poweroff.

AUTHOR

Miquel van Smoorenburg, miquels@cistron.nl

SEE ALSO

shutdown(8), init(8)

Curiosidad:
La criptografía en EEUU, está considerada como arma, y está en el mismo saco que una bomba nuclear o la munición de un M-16.

3.2.1.2 Conceptos avanzados.

En este apartado lo que se pretende es dar unas nociones sobre gestión del sistema y ampliar las habilidades con la shell. A modo de resúmen podemos decir que lo que se va a ver es:

- Modificar el comportamiento de los comandos dentro de la shell.
- Cómo gestionar procesos[2] activos o inactivos.

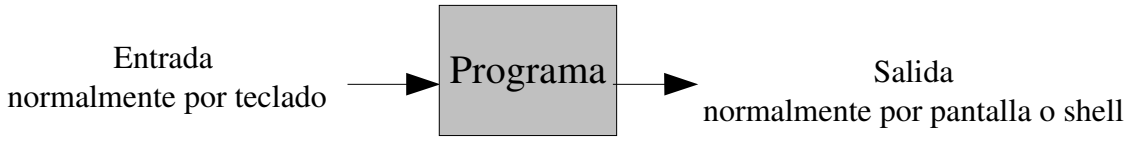
La razón de ser de estos conceptos avanzados es la siguiente, como todo sistema operativo Linux puede sufrir algún problema, por lo que hemos de saber eliminar ese problema para que el resto del sistema pueda funcionar con normalidad. A continuación veremos los conceptos de redirección.

3.2.1.2.1 Redireccionar salidas (>)

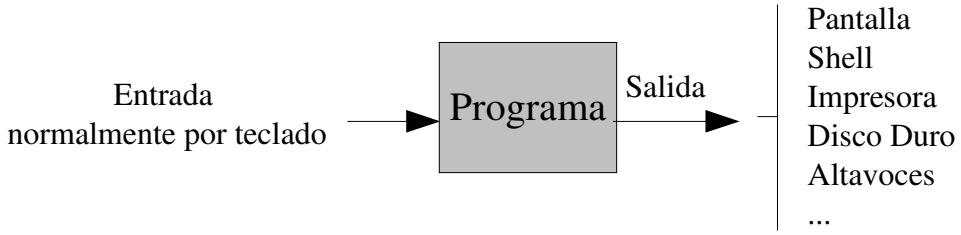
Antes de empezar, definamos la idea de redirección.

Normalmente los comandos de shell nos muestran el resultado por pantalla dentro de la misma shell, pero en determinadas ocasiones es más útil que el resultado (llamado salida), sea mostrado en otros sitios, por ejemplo en un fichero o en otro dispositivo.

Veámos un esquema, de la ejecución típica de un programa.



Ahora veámos cómo queda afectado el esquema por el uso de una redirección.



Es decir, que podemos decidir donde va a parar nuestra solicitud.

¿Cómo lo hacemos?

La sintaxi general es la siguiente:

programa **>** *RecursoDondeGuardarLaSalida*

donde:

programa es cualquier comando o programa que se use por la shell y *RecursoDondeGuardarLaSalida* es dónde guardaremos el resultado del programa a ejecutar.

Veámos un ejemplo, donde hacer un listado de un directorio y sacarlo por pantalla no es demasiado cómodo.

2 Proceso: Dígase de un software que se ejecuta tanto en primer plano (que se ve, por ejemplo un cliente de correo electrónico) como en segundo plano (invisible, por ejemplo un antivirus).

```
Usuario@MIPC:/usr/src/linux-2.4.21/kernel$ ls -la
total 1044
drwxr-xr-x    2 573      573          4096 2003-09-17 16:29 .
drwxr-xr-x   14 573      573          4096 2004-03-05 19:59 ..
-rw-r--r--    1 573      573          9878 2002-08-03 00:39 acct.c
-rw-r--r--    1 root     root          744 2003-08-15 17:14 acct.o
-rw-r--r--    1 root     root          369 2003-08-15 17:14 .acct.o.flags
-rw-r--r--    1 573      573          6364 2000-06-24 04:06 capability.c
-rw-r--r--    1 root     root         3464 2003-08-15 17:14 capability.o
-rw-r--r--    1 root     root          381 2003-08-15 17:14 .capability.o.flags
-rw-r--r--    1 573      573          4629 2001-10-11 18:17 context.c
-rw-r--r--    1 root     root         3856 2003-08-15 17:15 context.o
-rw-r--r--    1 root     root          407 2003-08-15 17:15 .context.o.flags
-rw-r--r--    1 root     root        15057 2003-08-15 17:13 .depend
-rw-r--r--    1 573      573          2871 2001-02-13 22:14 dma.c
-rw-r--r--    1 root     root         1488 2003-08-15 17:14 dma.o
-rw-r--r--    1 root     root          367 2003-08-15 17:14 .dma.o.flags
-rw-r--r--    1 573      573          6625 2002-02-25 19:38 exec_domain.c
-rw-r--r--    1 root     root         6352 2003-08-15 17:14 exec_domain.o
-rw-r--r--    1 root     root          415 2003-08-15 17:14 .exec_domain.o.flags
-rw-r--r--    1 573      573         14272 2002-11-28 23:53 exit.c
-rw-r--r--    1 root     root         8964 2003-08-15 17:14 exit.o
-rw-r--r--    1 root     root          369 2003-08-15 17:14 .exit.o.flags
-rw-r--r--    1 573      573         20748 2003-06-13 14:51 fork.c
-rw-r--r--    1 root     root        10724 2003-08-15 17:14 fork.o
-rw-r--r--    1 root     root          369 2003-08-15 17:14 .fork.o.flags
-rw-r--r--    1 573      573          1903 2001-04-20 23:15 info.c
-rw-r--r--    1 root     root         1888 2003-08-15 17:14 info.o
-rw-r--r--    1 root     root          369 2003-08-15 17:14 .info.o.flags
-rw-r--r--    1 573      573          3977 2000-06-29 17:07 itimer.c
-rw-r--r--    1 root     root         3064 2003-08-15 17:14 itimer.o
-rw-r--r--    1 root     root          373 2003-08-15 17:14 .itimer.o.flags
-rw-r--r--    1 root     root       216695 2003-08-15 17:15 kernel.o
-rw-r--r--    1 root     root          321 2003-08-15 17:15 .kernel.o.flags
-rw-r--r--    1 573      573          9993 2002-08-03 00:39 kmod.c
-rw-r--r--    1 root     root         6824 2003-08-15 17:15 kmod.o
-rw-r--r--    1 root     root          401 2003-08-15 17:15 .kmod.o.flags
-rw-r--r--    1 573      573         16003 2003-06-13 14:51 ksyms.c
-rw-r--r--    1 root     root        69576 2003-08-15 17:15 ksyms.o
-rw-r--r--    1 root     root          403 2003-08-15 17:15 .ksyms.o.flags
-rw-r--r--    1 573      573          1235 2001-09-17 04:22 Makefile
-rw-r--r--    1 573      573         29610 2003-06-13 14:51 module.c
-rw-r--r--    1 root     root        16732 2003-08-15 17:14 module.o
-rw-r--r--    1 root     root          373 2003-08-15 17:14 .module.o.flags
-rw-r--r--    1 573      573          3298 2002-11-28 23:53 panic.c
-rw-r--r--    1 root     root         2864 2003-08-15 17:14 panic.o
-rw-r--r--    1 root     root          371 2003-08-15 17:14 .panic.o.flags
-rw-r--r--    1 573      573          8205 2002-11-28 23:53 pm.c
-rw-r--r--    1 root     root         3692 2003-08-15 17:15 pm.o
-rw-r--r--    1 root     root          397 2003-08-15 17:15 .pm.o.flags
-rw-r--r--    1 573      573         18254 2003-06-13 14:51 printk.c
-rw-r--r--    1 root     root         9036 2003-08-15 17:14 printk.o
-rw-r--r--    1 root     root          405 2003-08-15 17:14 .printk.o.flags
-rw-r--r--    1 573      573          4816 2003-06-13 14:51 ptrace.c
-rw-r--r--    1 root     root         3448 2003-08-15 17:14 ptrace.o
-rw-r--r--    1 root     root          373 2003-08-15 17:14 .ptrace.o.flags
-rw-r--r--    1 573      573          7219 2002-11-28 23:53 resource.c
-rw-r--r--    1 root     root         4236 2003-08-15 17:14 resource.o
-rw-r--r--    1 root     root          377 2003-08-15 17:14 .resource.o.flags
-rw-r--r--    1 573      573         33904 2003-06-13 14:51 sched.c
-rw-r--r--    1 root     root        16640 2003-08-15 17:14 sched.o
-rw-r--r--    1 root     root          395 2003-08-15 17:14 .sched.o.flags
-rw-r--r--    1 573      573         29622 2003-06-13 14:51 signal.c
-rw-r--r--    1 root     root        16288 2003-08-15 17:15 signal.o
-rw-r--r--    1 root     root          405 2003-08-15 17:15 .signal.o.flags
-rw-r--r--    1 573      573          8910 2002-11-28 23:53 softirq.c
-rw-r--r--    1 root     root         9392 2003-08-15 17:14 softirq.o
-rw-r--r--    1 root     root          375 2003-08-15 17:14 .softirq.o.flags
-rw-r--r--    1 573      573         30988 2003-06-13 14:51 sys.c
-rw-r--r--    1 573      573         37968 2003-06-13 14:51 sysctl.c
```

```
# Antes de usar la redirección planteamos una
# situación que motivará el uso de la redirección.
# Listamos el contenido del directorio /usr/src/linux-2.4.21/kernel
```

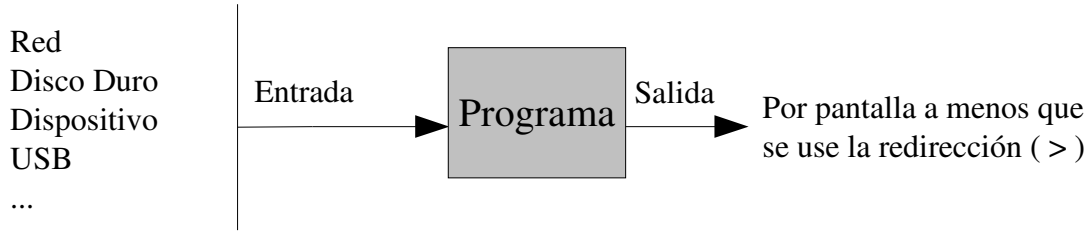
```
-rw-r--r--   1 root     root        16376 2003-08-15 17:14 sysctl.o
-rw-r--r--   1 root     root          373 2003-08-15 17:14 .sysctl.o.flags
-rw-r--r--   1 root     root        15456 2003-08-15 17:15 sys.o
-rw-r--r--   1 root     root          399 2003-08-15 17:15 .sys.o.flags
-rw-r--r--   1 573      573         11992 2002-11-28 23:53 time.c
-rw-r--r--   1 root     root         5924 2003-08-15 17:14 time.o
-rw-r--r--   1 root     root          369 2003-08-15 17:14 .time.o.flags
-rw-r--r--   1 573      573         22161 2002-11-28 23:53 timer.c
-rw-r--r--   1 root     root         9952 2003-08-15 17:15 timer.o
-rw-r--r--   1 root     root          371 2003-08-15 17:15 .timer.o.flags
-rw-r--r--   1 573      573          4073 2000-01-11 02:40 uid16.c
-rw-r--r--   1 root     root         3852 2003-08-15 17:15 uid16.o
-rw-r--r--   1 root     root          371 2003-08-15 17:15 .uid16.o.flags
-rw-r--r--   1 573      573          3111 2000-11-29 06:43 user.c
-rw-r--r--   1 root     root         2296 2003-08-15 17:15 user.o
-rw-r--r--   1 root     root          369 2003-08-15 17:15 .user.o.flags
Usuario@MIPC:/usr/src/linux-2.4.21/kernel$
```

Vaya hay muchos ficheros que no caben en una pantalla de la shell y necesitamos analizarlos bien a fondo, por lo que sería útil tener ese listado en un fichero de texto.

```
Usuario@MIPC:/usr/src/linux-2.4.21/kernel$ ls -la  > listado.texto      # Hacemos que ls nos guarde la salida en el
                                                                        # fichero listado.texto
Usuario@MIPC:/usr/src/linux-2.4.21/kernel$ less listado.texto           #Visualizamos el fichero con cualquier
...                                                                     #programa por ejemplo, less.
Usuario@MIPC:/usr/src/linux-2.4.21/kernel$
```

3.2.1.2.2 Redireccionar entradas (<)

Redireccionar las entradas es muy similar a redireccionar las salidas, de hecho son operaciones que se podrían considerar “hermanas”. La idea es la siguiente, dado un comando o programa que funciona bajo shell, si este programa ha de recibir muchos datos, como por ejemplo muchas pulsaciones de tecla para introducir un texto, en vez de tener que teclear ese texto cada vez que queramos usar el programa o comando, lo que hacemos es escribirlo en un fichero ascii, y usar la redirección.



¿Cómo lo hacemos?

La sintaxi general es la siguiente:

programa **<** *DatosDeEntradaAlPrograma*

donde:

programa es cualquier comando o programa que se use por la shell y *DatosDeEntradaAlPrograma* es dónde hemos guardado la infromación que se le ha de proporcionar al programa.

Ejemplo:

Para este ejemplo usaremos un programa sin demasiada utilidad práctica, pero resulta curioso, cowsay. A cowsay se le ha de proporcionar un texto para que después él lo muestre por pantalla de una forma divertida.

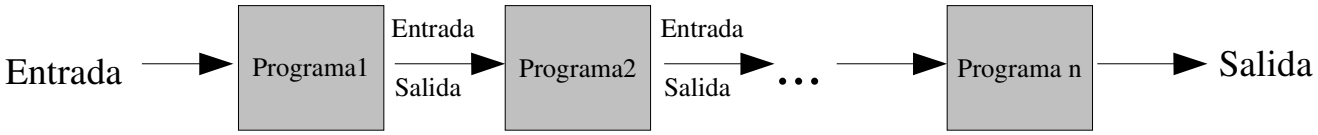
```
Usuario@MiPC:/home/prueba$ ls                                   # veámos que hay en el directorio
diario  diario2  docs  leeme.txt  nombreNuevo  textoParaCowsay
Usuario@MiPC:/home/prueba$ cat textoParaCowsay                  # veámos el contenido de textoParaCowsay
Hola, este texto es el que quiero que cowsay diga, pero como es largo y se supone que esta operación la he de
realizar muchas veces, sólo escribo esto una vez y lo guadro en un fichero ascii.

Usuario@MiPC:/home/prueba$ cowsay < textoParaCowsay             #Ahora en vez de teclear el mensaje usamos la
                                                                # redirección
 ____________________________________
/ Hola, este texto es el que quiero que  \
| cowsay diga, pero como es largo y se   |
| supone que esta operación la he de     |
| realizar muchas veces, sólo escribo    |
| esto una vez y lo guadro en un fichero |
\ ascii.                                 /
 ----------------------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||------w |
                ||      ||
Usuario@MiPC:/home/prueba$
```

3.2.1.2.3 Pipes (|)

Definamos el concepto, una pipe no es más que una relación entre procesos, es decir, una forma de que dos procesos o programas intercambien información. Se podría decir que una pipe hace el papel de una empresa de logística, un programa le dice a la pipe que le entregue un paquete con información a otro programa y es la pipe quien se encarga de distribuir el envío y de notificar la correcta llegada.

La idea de pipes en Linux es la siguiente, dado un programa o comando shell, queremos que lo que nos devuelve por pantalla sea utilizado por otro programa como entrada, es decir, en vez de usar redirecciones a ficheros ascii, para después usarlos como entrada al segundo programa, lo hacemos mediante una pipe de forma automatizada, veámos el siguiente esquema.

Este es el comportamiento normal de un programa. Veámos como queda usando varios programas y varias pipes para aprovechar las salidas y las entradas.



En esta secuencia de programas vemos ilustrada la idea de pipe, reaprovechar las salidas de los programas como entradas de otros con el fín de obtener una salida global.

La sintaxi general es la siguiente:

programa1 -*opciones* **|** **programa2 -*opciones*** **|** **programa3 -*opciones*** **|** **...** **|** **programaN -*opciones***

donde:

| es el operador que se usa para crear una pipe.

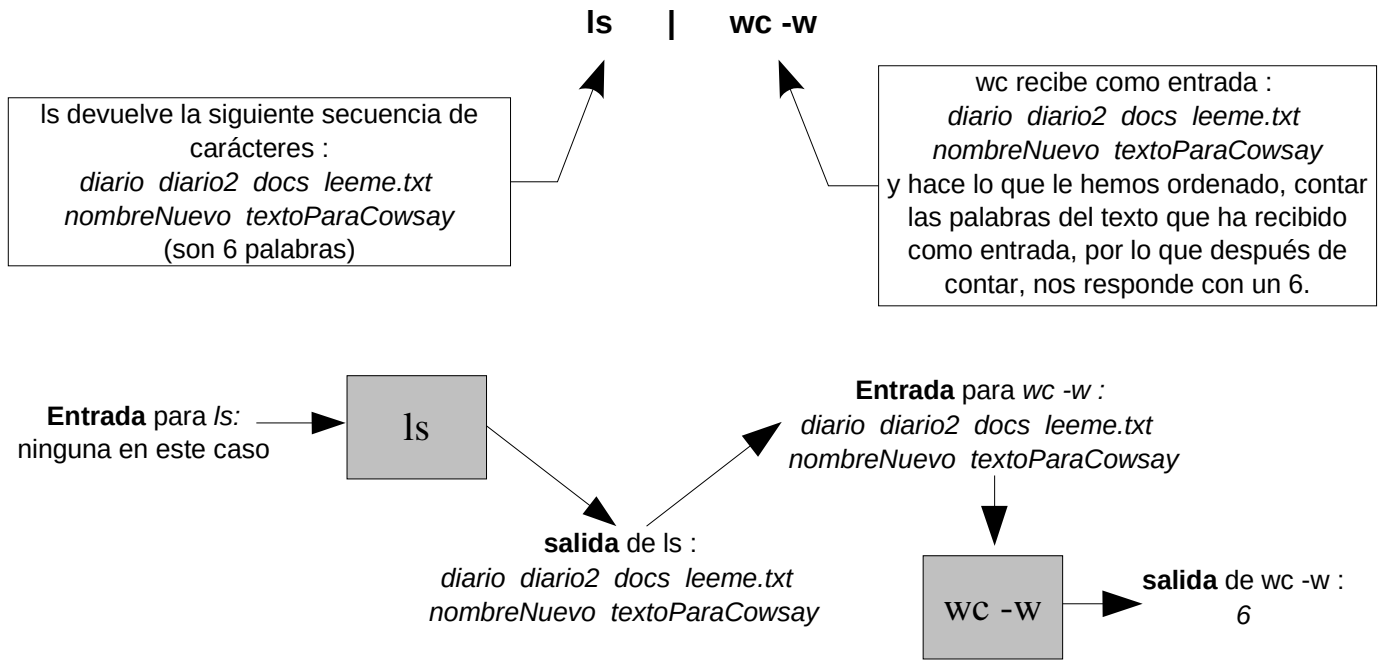
Ejemplos.

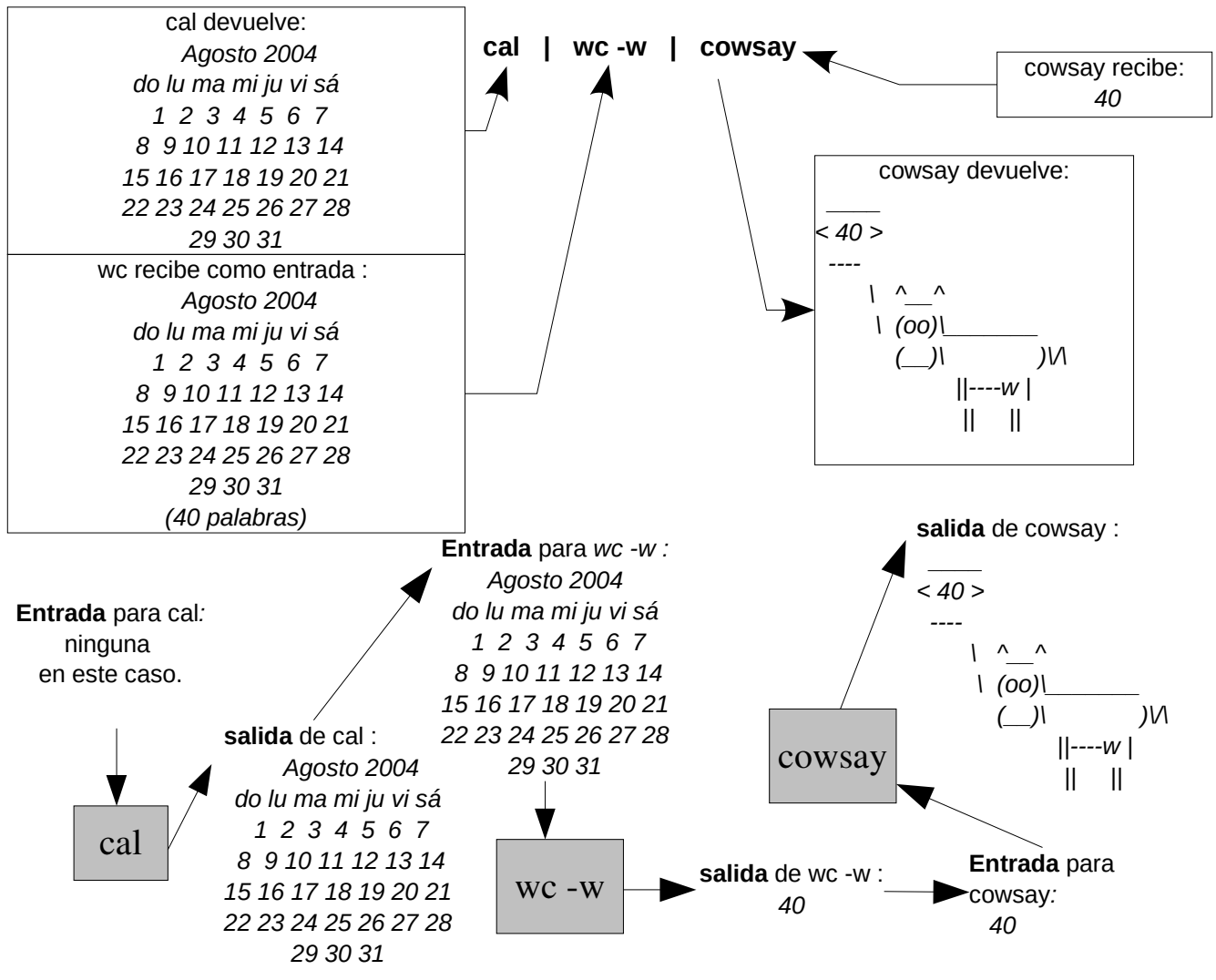
Nota: el programa wc nos devuelve el número de palabras dentro de un texto.

```
Usuario@MiPC:/home/prueba$ ls                          # veámos el contenido del directorio.
diario diario2 docs leeme.txt nombreNuevo textoParaCowsay
Usuario@MiPC:/home/prueba$ ls | wc -w                  # contemos cuántos ficheros hay
6
Usuario@MiPC:/home/prueba$
```

Curiosidad:
"No hay ninguna razón para que alguien quiera un ordenador en su casa."
Ken Olson, presidente y fundador de Digital Equipment Corp., 1977

Dibujemos el esquema para ver que ha pasado en la ejecución :





3.2.1.3 Gestión de procesos.

Aunque el título pueda asustar, el tema es muy sencillo, la gestión de procesos a nivel de usuario, consiste en tener el control de lo que está en marcha en nuestro PC, y si algo no nos gusta o ha fallado, poder eliminarlo y así evitar que moleste a los demás procesos, en definitiva, dicho a lo bruto, gestionar procesos es: cerrar los programas que se han quedado colgados.

Otros sistemas operativos nos ofrecen una pequeña ventana donde se listan los programas activos y nos dejan cerrarlos, eso si tenemos suerte ya que dicha ventana no siempre aparece ;-) . En Linux este procedimiento se lleva a cabo de una forma que no puede fallar, a través de la consola.

Procedimiento:

1. Listar todos los procesos del sistema.
2. Obtener el PID, o identificador, del proceso en discordia.
3. Cerrarlo.

¿Cómo se hace ?

Para realizar cada paso vamos a necesitar un comando diferente, para listar los procesos usaremos :

```
Usuario@MiPC/$ ps -A
```

-A indica que nos muestre TODO (All) , para más información consultar el manual de linux ($*man ps*)

De ese listado nos interesa el PID, la primera columna, anotamos ese PID.

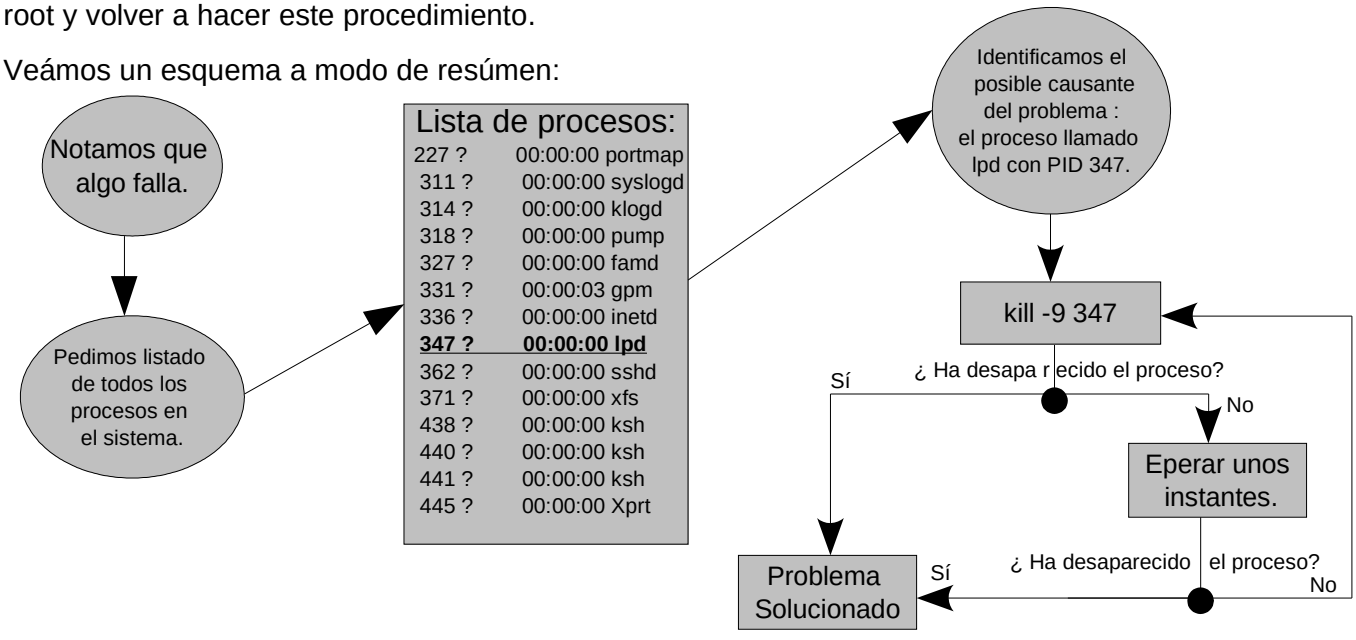
Para cerrar el programa se hace :

```
Usuario@MiPC/$ kill -9 PID
```

-9 hace referencia a la acción de cerrar.

NOTA : A veces no podremos cerrar determinados procesos, una posible solución es identificarnos como root y volver a hacer este procedimiento.

Veámos un esquema a modo de resúmen:



3.2.1.4 Detalles en la ejecución de programas desde la shell: *./ , & , &&.*

En este apartado sobre la shell del sistema, vamos a mencionar ciertos aspecto útiles en su uso relativos a la ejecución de programas.

Para ejecutar un fichero tenemos dos opciones, una es que ese fichero se encuentre en los directorios /bin o /sbin , por lo que con teclear su nombre ya podedmos ejecutarlo, o dos, que el fichero está en cualquier otro sitio, para ello hemos de emplear el operador **./ ,** para usar el operador es preciso ir al directorio del fichero y una vez dentro usar : ./NombreFichero .

```
Usuario@MiPC/$ ls
Docs libreta.txt dibujo.bmp agenda
Usuario@MiPC/$ ./agenda
---- Se ejecuta el programa-----
            ...
---- Se cierra el programa-----
Usuario@MiPC/$
```

Aquí han pasado un par de cosas interesantes: Una es que mediante el operador **./** la shell ha sabido qué hacer con ese fichero, y lo ha ejecutado, otra es que mientras ese programa estaba en funcionamiento, la shell ha quedado *bloqueada*, es decir, no podíamos hacer nada con ella. Para evitar el bloqueo de una shell al ejecutar algo, hemos de usar el operador **&** , veámos un ejemplo:

```
Usuario@MiPC/$ ls
Docs libreta.txt dibujo.bmp agenda
Usuario@MiPC/$ ./agenda &                    # Ejecutamos agenda.
Usuario@MiPC/$ ls                            # Agenda sigue en marcha pero podemos seguir usando la
Docs libreta.txt dibujo.bmp agenda           # consola.
Usuario@MiPC/$
FIN de agenda [457]                          # Cuando el programa agenda se cierra, se nos da un aviso
Usuario@MiPC/$                               # en la shell, pero sin perder el control de la misma.
```

La consola del sistema, también nos permite hacer una ejecución en serie, de comandos mediante una sola línea de instrucciones, es decir, podemos llamar a varios programas para que se ejecuten uno detrás de otro, con una sola línea de comandos, para esto se usa el operador **&&,** veámos un ejemplo:

```
Usuario@MiPC/$ ls  && ls -la && ls /usr/src
Docs libreta.txt dibujo.bmp agenda                                  # primer ls

                                                                    _
drwxr-xr-x   5 Usuario  Usuario       4096   2004-08-03 10:43 .     #  \
drw-r--r--    4 Usuario  Usuario      4096   2004-07-04 12:08 Docs  #    \
-rw-r--r--    4 Usuario  Usuario     70012   2004-07-01 14:42 libreta.txt  #     -> ls -la
-rw-r--r--    4 Usuario  Usuario     12097   2004-07-01 15:15 dibujo.bmp    #    /
-rwxr-xr-x    4 Usuario  Usuario     20480   2004-07-01 14:42 agenda        # _/
linux-2.4.20  linux-2.4.20.tar.gz  linux-2.4.21  linux-2.4.21.tar  rpm     # ls /usr/src
Usuario@MiPC/$                                  # se ha producido la ejecución de 3 comandos especificados
                                                # en una única línea de la shell.
```

Curiosidad:

En Francia para poder encriptar un documento, se le ha de pedir permiso al primer ministro.

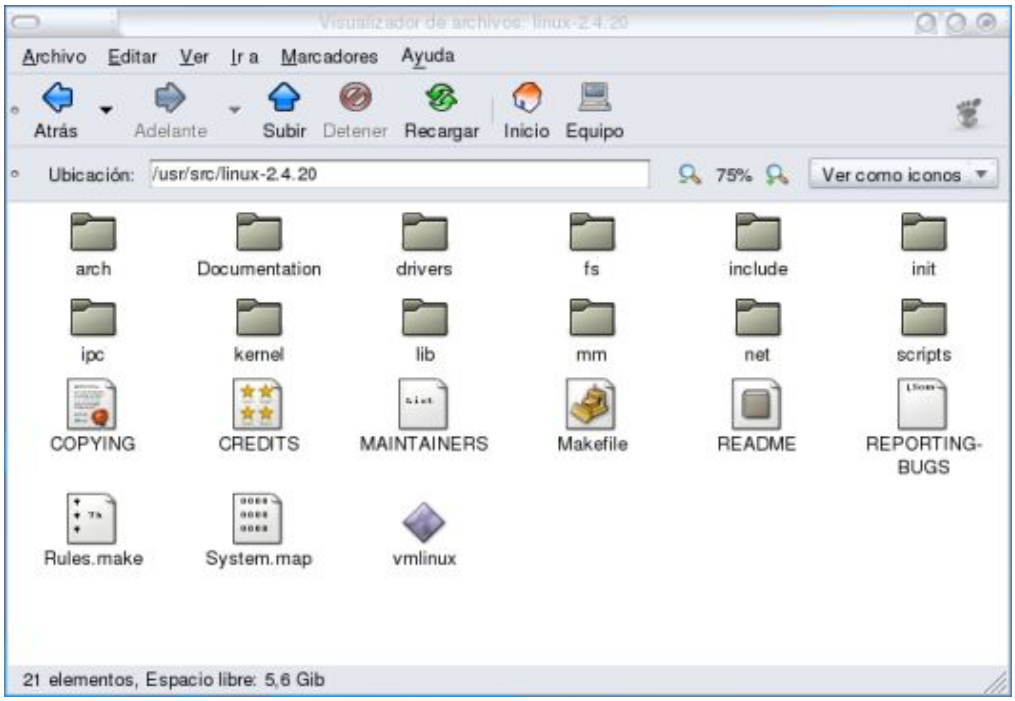
3.2.2 Sistema Xwindow.

Xwindow hace referencia al entorno gráfico de un sistema Linux, como todo entorno gráfico, uno de sus objetivos es el de hacer más agradable las tareas al usuario, y realmente lo consigue. A la hora de escribir un manual sobre el sistema Xwindow de Linux, hay muchos problemas, ya que a diferencia de otros sistemas operativos, las X, como se les llama coloquialmente, varian mucho, es decir, cada cierto tiempo se sacan nuevas versiones de los escritorios, donde puede variar sustancialmente la distribución de opciones, por lo que un buen consejo es el siguiente : Dejarse llevar y disfrutar de lo que se ve y adaptarse de forma tranquila y amena a cada versión, para ello es **IMPRESCINDIBLE** ir despacio y **LEER lo que pone la pantalla**, ya que hay cuantiosa información útil por todo el entorno. Afortunadamente la adaptación entre versiones suele ser muy sencilla.

Conceptos comúnes en entornos gráficos basados en ventanas.

NOTA: En Linux existen varios gestores de ventanas los dos más pupulares son : Gnome, (http://www.gnome.org/) y KDE (http://www.kde.org/). Estos dos gestores representan bastante bien la idea de entorno de ventanas y sus diferencias no son significativas en cuanto a uso, por eso los tomaremos como referencia.

Vayamos paso a paso, identifiquemos el concepto de ventana (una imágen vale más que mil palabras) :



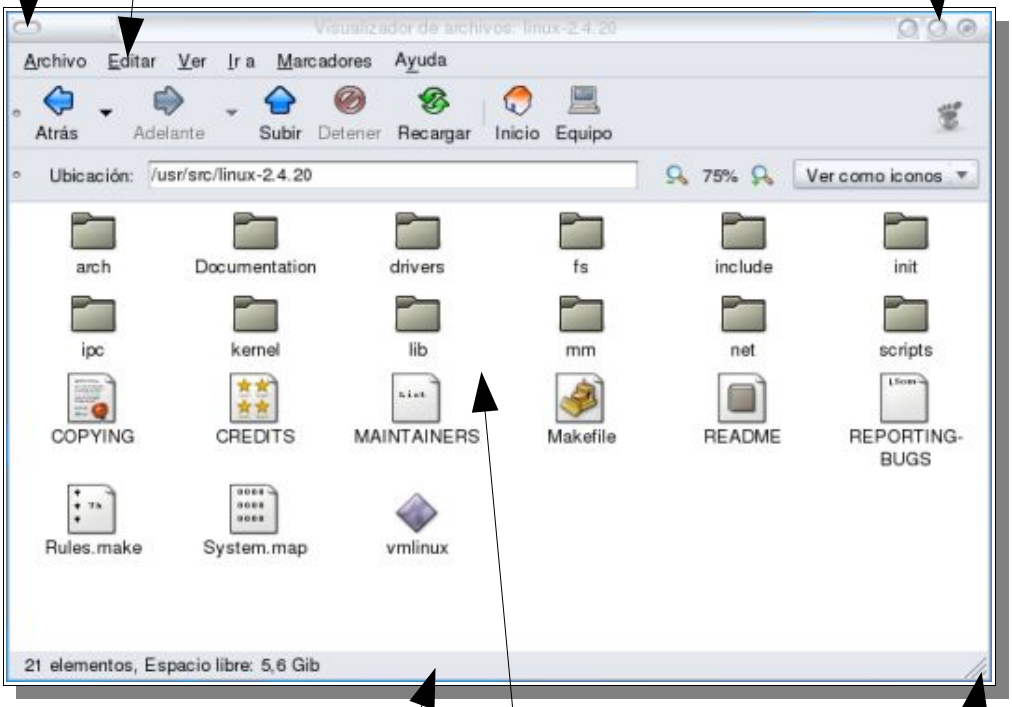
Dar una definición formal de lo que es una ventana, puede dar lugar a mucha controversia, por lo que es mejor que cada cual se genere su propio concepto de lo que es una ventana. La parte positiva es que una ventana, por muchos cambios de aspecto que sufra, sigue siendo una ventana, así que conserva sus propiedades básicas.

Distingamos esas partes básicas dentro de una ventana.

Botones de control y visualización de la ventana.
De derecha a izquierda :
CERRAR, MAXIMIZAR,MINIMIZAR

Botón de control general de la ventana.
De él surge un menú donde nos suele permitir , maximizar,minimizar y cerrar la ventana, además de proporcionarnos más opciones que varian según el programa y el entorno gráfico.

Menú de opciones clásico.
Este conjunto de opciones varia para cada programa, lo que si suele ser algo constante es su presencia. Nos permiten tener control sobre la aplicación que estemos usando.



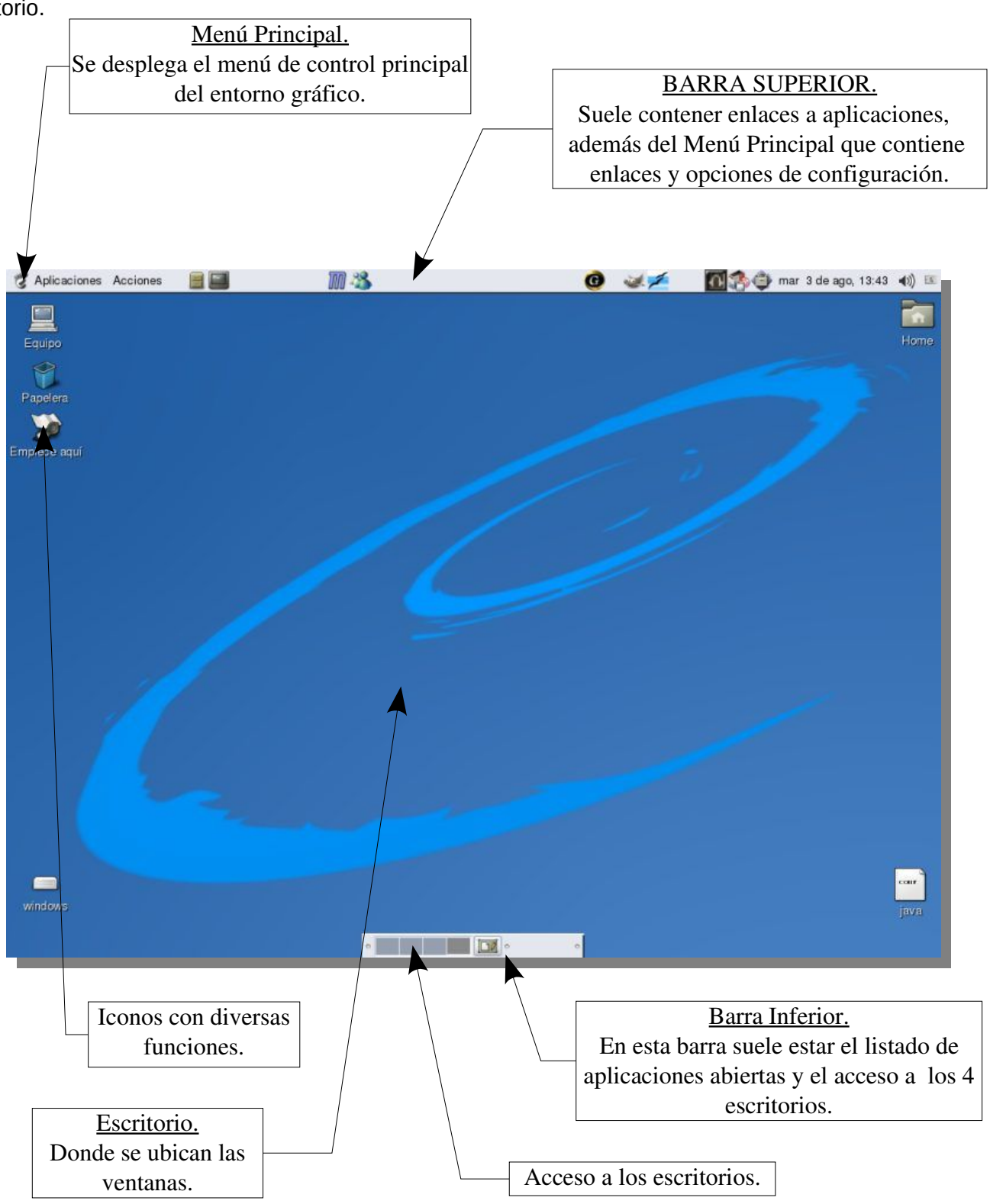
Visualización del contenido de la ventana.

Barra de Estado.
Proporciona información adicional al usuario. Esta barra no siempre está presente.

Esquina de modificación de tamaño.
Arrastrando esta esquina el tamaño de la ventana se modifica.

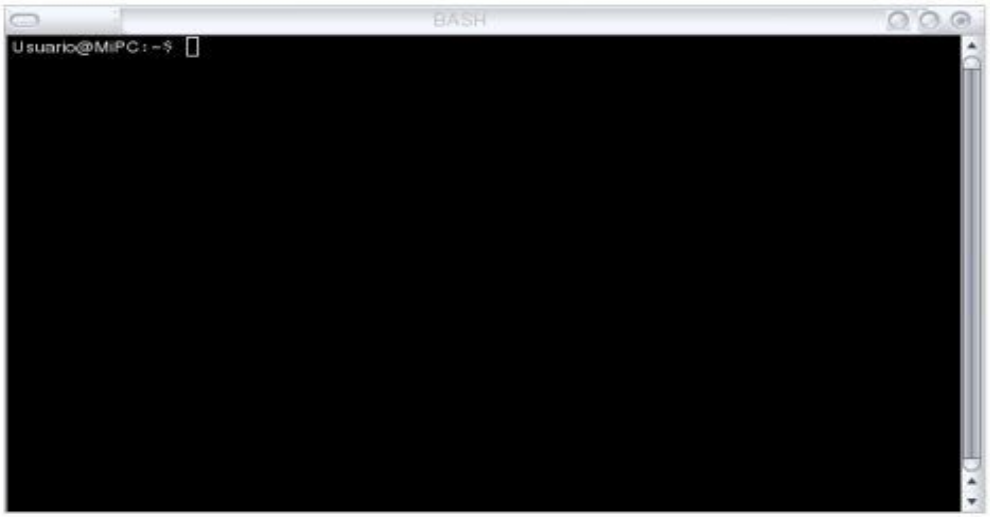
3.2.2.1 Entorno de escritorio.

Una vez que ya sabemos que es una ventana, veámos donde se agrupan las ventanas. Las ventanas están dentro de lo que se llama, escritorio, en Xwindow no nos limitamos a tener un único escritorio, si no que hay un cierto número de ellos accesibles de forma rápida. En general los entornos gráficos vienen con 4 escritorios inicialmente accesibles, pero antes de entrar en detalle veámos un escritorio.



3.2.2.2 Ideas útiles para el sistema Xwindow.

- El hecho de que usemos un entorno gráfico no nos obliga a depender de él al 100%, podemos seguir usando la consola de varias formas: Una de ellas es pulsar las teclas CTRL+ALT+F1 ó F2 .. F5 para acceder a una sesión en consula pura, para volver al entorno gráfico pulsar CTRL+ALT+F6 ... F9, atención que dependiendo de la versión del entorno gráfico las teclas "Fx" pueden variar. La segunda manera es abir una consola dentro de las X's, en general se suele llamar *terminal*, por lo que para abrir un terminal se ha de buscar la opción dentro de los menús ya que puede variar. Como norma general el icono que representa al terminal tiene un aspecto similar a éste : 
  Este debería ser el aspecto aproximado de un terminal dentro de la mayoría de entornos gráficos:



- Al usar un terminal en las X's, es recomendado usar el operador **&,** a la hora de ejecutar programas .
- Los menús principales están representados por los siguientes iconos en sus respectivas barras de tareas :

  Gnome :  **KDE** : 

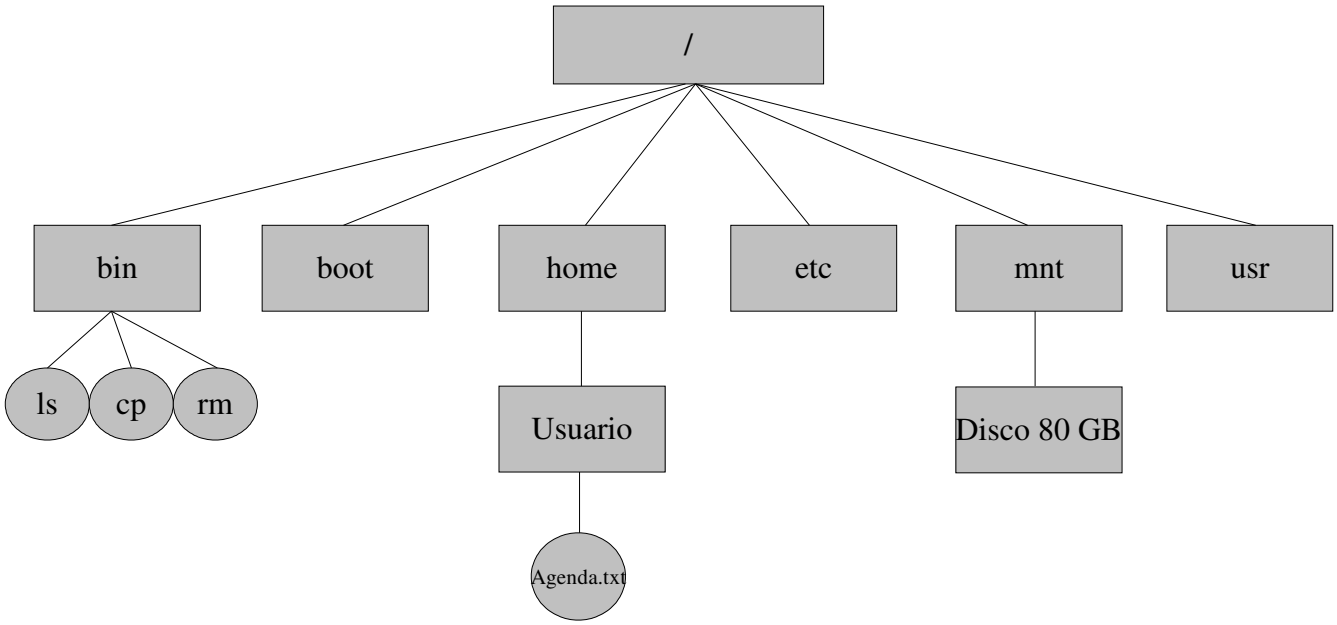
- Para configurar ciertos aspectos del escritorio, en concreto fondos de pantalla y salvapantallas, se suele hacer pulsando el botón derecho del ratón sobre el fondo de pantalla actual y seleccionar la opción de *configurar escritorio* o *cambiar fondo de escritorio*.
- Una manera de investigar y aprender cómo se usa un entrono gráfico podría ser: Probar qué efecto tiene el click izquierdo y derecho sobre las barras, menús, iconos varios etc...

3.3 El Sistema de ficheros.

3.3.1 Introducción.

Lo que se prentende ver en este tema es, cómo están organizados los ficheros dentro de un entorno Linux, es decir, dónde se guardan los documentos, dónde se instalan los programas etc ...

La característica más relevante del sistema de ficheros, obviando todo el apartado técnico, sería la siguiente : El sistema de ficheros de Linux se basa en una estructuración arborescente de los datos. ¿Qué quiere decir ? en esencia que TODOS los ficheros o directorios parten del mismo sitio, en otros sistemas de ficheros, ese origen sería una letra que identifica a la unidad, en linux no hay unidades, lo que hay es el sistema de ficheros y éste puede integrar varias unidades pero las trata como si fueran un directorio más, no se les da un tratamiento específico por ser unidades de almacenamiento diferentes. Veámos una representación de la idea.



A partir de ahora haremos referencia al sistema de ficheros de Linux por su nombre, EXT2 ó EXT3 según la versión que se tenga, las diferencias entre los dos son una cuestión puramente técnica por lo que no las vamos a mencionar, además las diferencias son totalmente transparentes para el usuario ya que no alteran su uso. Para empezar a calentar motores, vamos a ver el significado de los directorios que vienen por defecto en Linux.

Directorio.	*Descripción del contenido.*
/	Directorio raíz. Donde comienza el árbol de directorios.
/bin	Binarios, ejecutables, por ejemplo algunos comandos de shell , ls , cp , mv ...
/boot	Fichero de inicio (núcleo), y otros ficheros de carga.
/dev	Ficheros de dispositivos.
/etc	Ficheros de configuración de la máquina. Algunos programas guardarán sus ficheros de configuración en /etc o /usr/etc.
/etc/X11	Ficheros de configuración del sistema X11.
/home	Directorio donde se guardarán los directorios persinales de cada usuario, como por ejemplo Luís, que sería : /home/Luis.
/lib	Bibliotecas compartidas (shared) o librerías dinámicas, necesarias para el funcionamiento del sistema.
/mnt	Directorio donde se montarán las unidades externas, como otros discos duros o particiones.
/proc	Información sobre los procesos en ejecución.
/sbin	Directorio que contiene comandos, sólo ejecutables para el SuperUsuario o root.
/tmp	Este directorio contiene ficheros temporales que pueden borrarse sin previo aviso.
/usr	Contiene ficheros de ciertos programas, además del código fuente del sistema operativo.
/usr/etc	Directorio donde se guardan los ficheros de configuración de los distintos programas.
/usr/lib/X11	Librerías para las X.
/usr/lib/zoneinfo	Ficheros para la información de la zona horaria.
/usr/local	Aquí es donde van típicamente los programas que son locales a la máquina.
/usr/local/bin	Aquí van los binarios de los programas locales a la máquina.
/usr/local/etc	Ficheros de configuración instalados localmente.
/usr/local/man	Ayudas.
/usr/local/src	Código fuente para los programas instalados localmente.
/usr/man/<locale>/man[1-9]	Aquellos sistemas que den cabida a varios usuarios de distintas nacionalidades, podrán tener en la cadena <locale>, el lenguaje al que pertenece cada ayuda.
/usr/sbin	Programas para la administaración del sistema, sólo para el root.
/usr/src/linux	Código fuente de Linux.
/usr/tmp	Directorio que contiene información temporal .
/var	Contenedor de información, como registros de último acceso, colas de impresión, peticiones, menús del entorno gráfico, ... en definitiva un poco de todo.
/var/log	Ficheros "log".
/var/tmp	Como /tmp este directorio contiene ficheros temporales, almacenados durante un tiempo no especificado.

Curiosidad:

Las siglas **TWAIN** (scanners y cámaras digitales) significan : **T**echnology **W**ithout **A**n **I**nteresting **N**ame.

3.3.2 Acceso a directorios.

Como hemos visto, todos los directorios parten de la raíz (/), así que para acceder a ellos, tanto en órdenes como *cd* o como *cp*, se hará de dos formas :

- **Path(camino) Absoluto:** es aquel en que se escribe la ruta completa de acceso al directorio/fichero, por ejemplo : /home/Ignacio/trabajo.documento .
- **Path Relativo:** De esta forma nos ahorramos escribir la ruta de acceso completa, por ejemplo si estamos dentro del directorio /home, para acceder al documento trabajo.documento haremos : cat Ignacio/trabajo.documento .

3.3.3 Contenido común de un directorio.

Aunque un directorio esté recién creado, simpre contiene dos "ficheros", de hecho dos accesos a directorios, veámoslo:

```
Usuario@MiPC:/home/prueba$ mkdir vacio
Usuario@MiPC:/home/prueba$ cd vacio/
Usuario@MiPC:/home/prueba/vacio$ ls
Usuario@MiPC:/home/prueba/vacio$ ls -la
total 8
drwxr-sr-x    2 Usuario  staff       4096 2004-08-10 11:46   .
drwxr-sr-x    5 Usuario  staff       4096 2004-08-10 11:46   ..
Usuario@MiPC:/home/prueba/vacio$
```

el directorio **.** y el directorio **..** siempre están presentes, pero ¿ qué simbolizan?

El directorio actual se representa con un punto (.) y el anterior o padre se representa por dos puntos (..), por eso cuando hacemos cd .. , vamos al directorio anterior, ya que ese símbolo (..), siempre está presente en todo directorio, aunque este recién creado y por lo tanto vacío.

Ejemplos.

```
Usuario@MiPC:/home/prueba$ ls
diario  diario2  dibujos  docs  leeme.txt  nombreNuevo  textoParaCowsay        # docs y dibujos son 2 directorios
Usuario@MiPC:/home/prueba$ cd dibujos/                                  # accedemos a dibujos mediante su path relativo
Usuario@MiPC:/home/prueba/dibujos$ cd ..                                # volvemos a prueba.
Usuario@MiPC:/home/prueba$cd /home/prueba/docs                          # - Accedemos a docs con su path absoluto
Usuario@MiPC:/home/prueba/docs$  cd ../dibujos/                         # - Accedemos a dibujos mediante su path relativo,
Usuario@MiPC:/home/prueba/dibujos$                                      # indicando que queremos ir al directorio anterior ( .. )
                                                                        # y dentro de ése, al directorio dibujos.
Usuario@MiPC:/home/prueba/dibujos$ cd                                   # - Accedemos al directorio home del usuario actual.
Usuario@MiPC:/home/Usuario $ cd .././prueba/docs/                       # - Accedemos al directorio docs de una forma
                                                                        # especial, indicamos que queremos ir al directorio
                                                                        # anterior al actual ( .. ), que corresponde a /home , y
                                                                        # después queremos acceder al directorio donde
                                                                        # estamos dentro de la línea de comandos ( . ), que
                                                                        # corresponde a /home y finalmente accedemos a
                                                                        # prueba/docs.
                                        # Accesos con información redundante.

Usuario@MiPC:/home/prueba/docs$ cd ../docs/../docs/../dibujos/
Usuario@MiPC:/home/prueba/dibujos$  cd /home/prueba/dibujos/../dibujos/../docs/
```

3.3.4 Propiedades de los ficheros (permisos y propietarios).

Como en Linux no se usan las extensiones, como en otros sistemas operativos, parece que no hay manera de saber si un fichero es un ejecutable o un documento, además si usamos la órden ls, a secas, no se ve si lo que sale por pantalla corresponde a un ejecutable o a un directorio por ejemplo. Pues las propiedades de los ficheros hacen referencia a estas cuestiones.

Un fichero tiene de forma general 2 propiedades básicas, sus *permisos* y su *propietario*. Analicemos qué significado tiene la idea de **permisos**. Preguntémonos qué se suele hacer con un fichero, pues de forma rápida solemos responder : Mirarlo, modificarlo y si es un programa abrirlo. Bien, pues eso es lo que representan los permisos, nos dicen qué podemos hacer con un determinado fichero, de hecho los permisos van más allá, nos dicen qué se puede hacer con un fichero en función de quién seamos. El siguiente paso es ver de forma precisa los permisos, para eso retomaremos la órden *ls* de la shell, también se puede hacer con un visualizador de ficheros en entrono gráfico y pulsar botón derecho sobre su icono y acceder al menú *propiedades* o *permisos* según el entorno.
Antes de ver los permisos tal y como los muestra linux, veámos cuáles hay:

Permiso	Descripción
Lectura (**r**)	Si un fichero tiene este permiso significa que podemos acceder a él, es decir, ver su contenido.
Escritura (**w**)	Si un fichero tiene este permiso significa que podemos modificarlo.
Ejecución (**x**)	Si un fichero tiene este permiso significa que podemos ejecutarlo como si fuera un programa normal y corriente.
Permisos Múltiples.	Los ficheros pueden tener combinaciones de los tres anteriores.

Para acabar de aclarar la idea de cada permiso veámos una tabla con las operaciones que se pueden hacer según los permisos.

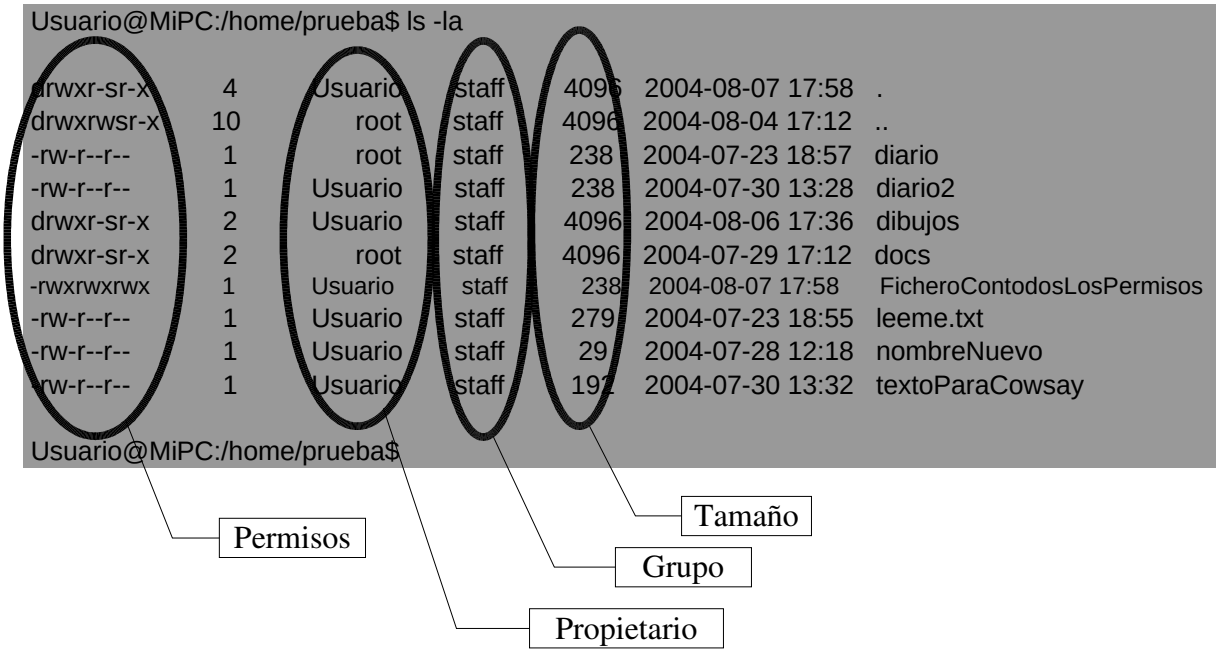
Permiso	Operaciones
Lectura (**r**)	Si un fichero tiene **sólo** este permiso : **PODEMOS** : borrarlo, copiarlo, moverlo, visualizar su contenido. **NO PODEMOS:** modificarlo,ejecutarlo.
Escritura (**w**)	Si un fichero tiene **sólo** este permiso : **PODEMOS**: borrarlo, modificarlo (sin abrirlo), moverlo. **NO PODEMOS:** copiarlo, ejecutarlo, visualizar su contenido.
Ejecución (**x**)	Si un fichero tiene **sólo** este permiso : **PODEMOS**: borrarlo, ejecutarlo, moverlo. **NO PODEMOS:** copiarlo, modificarlo, visualizar su contenido .

Permiso	Operaciones
Permisos Múltiples(I)	Si un fichero tiene este permiso, Lectura + Escritura (**rw**) : **PODEMOS**: borrarlo, copiarlo, modificarlo (abriéndolo, o no, antes), moverlo, ver su contenido. **NO PODEMOS:** ejecutarlo.
Permisos Múltiples(II)	Si un fichero tiene este permiso, Lectura + Ejecución (**rx**) : **PODEMOS**: borrarlo, copiarlo, moverlo, ver su contenido, ejecutarlo. **NO PODEMOS:** modificarlo.
Permisos Múltiples(III)	Si un fichero tiene este permiso, Escritura + Ejecución (**wx**) : **PODEMOS**: borrarlo,modificarlo (sin abrirlo), moverlo, ejecutarlo. **NO PODEMOS:** copiarlo,ver su contenido.

En el siguiente gráfico, se ve como se presentan al usuario los permisos en el entorno gráfico Gnome.

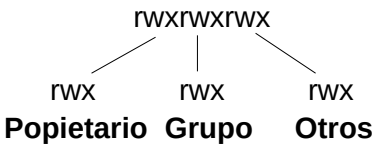
Curiosidad:
"La teoría sobre los gérmenes de Louis Pasteur es una ridícula ficción."
Pierre Pachet, Profesor de Fisiología en Toulouse, 1872.

Ahora veámos cómo nos muestra linux los permisos en consola.



Vaya, que forma más peculiar, expliquémosla :

Cada fichero tiene permisos para 3 tipos de usuario, el **creador o propietario** del fichero, el **grupo** y **otros**, de ahí que hayan 3 series de letras en la exposición de los permisos en la consola.



Recordar:

r	**R***ead* (lectura)
w	**W***rite* (escritura)
x	*e***X***ecute* (ejecución)

Vale, pero ¿ qué quiere decir *propietario*,*grupo* y *otros* ?

El usuario **propietario** de un fichero es aquel que lo creó, antención que Linux permite cambiar el propietario de un fichero.

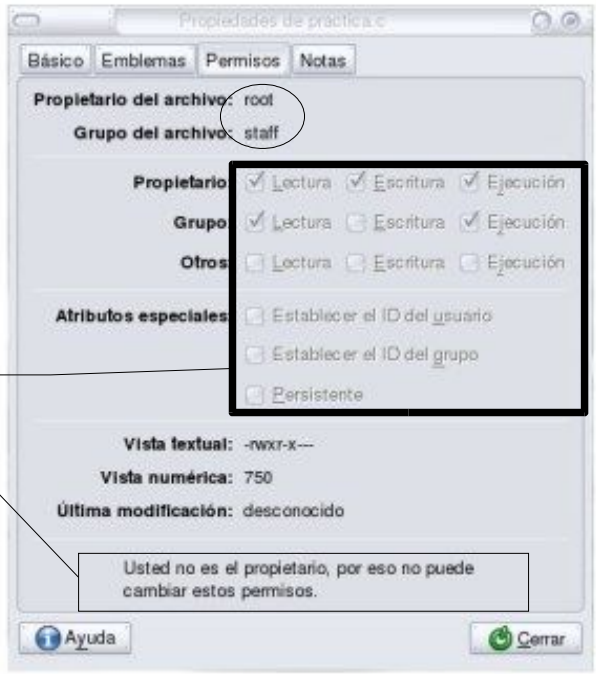
El **Grupo** hace de referencia al grupo de trabajo que puede acceder a ese fichero, esto está orientado para un sistema multiusuario, por ejemplo, si estamos en una empresa quizás nos interese que sólo el departamento de contabilidad pueda acceder al fichero *nóminas.documento* , pues ponemos que el grupo sea contabilidad.

Otros hace referencia a todo usuario que no sea el propietario, o que no pertenezca al grupo de trabajo.

3.3.5 Cómo cambiar permisos o propietarios.

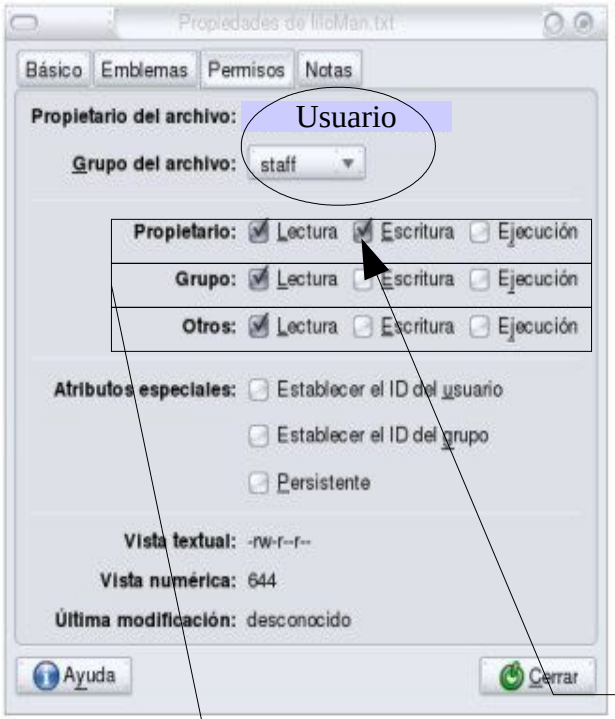
Para poder cambiar los permisos de un fichero/directorio, hemos de estar identificados como el propietario del fichero/directorio o como root. Una vez hecho esto, tenemos dos formas de cambiar permisos, mediante un entorno gráfico o por la shell, empecemos por el entorno gráfico, usaremos como referencia Gnome, pero en otros entornos no debería variar demasiado el aspecto de las ventanas implicadas. Para llegar a las ventanas aquí expuestas, se suele proceder de la siguiente manera: Encontrar el fichero con un navegador del sistema de ficheros, y hacer click derecho sobre él y seleccionar propiedades u otro sinónimo.

En este primer caso vemos algo que suele pasar, no somos ni el propietario del fichero ni el root, de hecho en este ejemplo coincide que el propietario del fichero es el propio root, por lo que no podemos variar los permisos .



Permisos no disponibles

Ayuda de Gnome.



Ahora ya somos el propietario del fichero o el root, y podemos proceder al cambio de permisos. En general en los entornos gráficos, los permisos vienen marcados con casillas de verificación, por lo que su cambio es sencillo, además nos señala a quién le corresponde cada permiso que activamos o desactivamos. Para cambiarlos lo único que hemos de hacer es pulsar sobre las casillas de verificación que deseemos alterar.

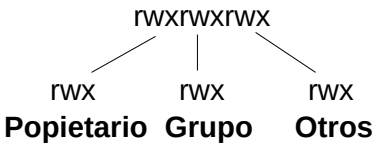
Casillas de verificación.

Separación por tipo de usuario.

NOTA : Como suele ser habitual en los entornos gráficos de Linux, este procedimiento puede llegar a ser sensiblemente diferente entre entornos y versiones de los mismos.

Cambiar los permisos desde la shell puede resultar para algunos una tarea complicada, pero la verdad es que es sólo en apariencia. Expliquémos un poco la teoría que hay detrás.

Como ya hemos comentado antes, los permisos se agrupan según el tipo de usuario, el propietario, el grupo de trabajo y el resto de usuarios.



Recordar:

r	**R***ead* (lectura)
w	**W***rite* (escritura)
x	*e***X***ecute* (ejecución)

Para cambiar los permisos en Linux por la shell, se usa la órden **chmod**, la sintaxi general es la siguiente:

chmod *permisos* fichero/directorio

donde:

permisos son los permisos que le queremos dar a *fichero/directorio.*

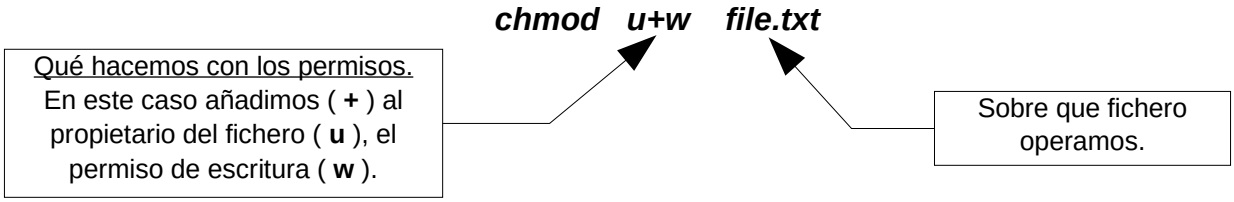
La sintaxi de *permisos* es un poco exótica, pero permite dos formas una numérica, y otra simbólica, veámos primero la simbólica.

Para cambiar permisos hemos de especificar primero a quién se los cambiamos y después cuáles quitamos y cuáles añadimos, siempre siguiendo esta convención de símbolos :

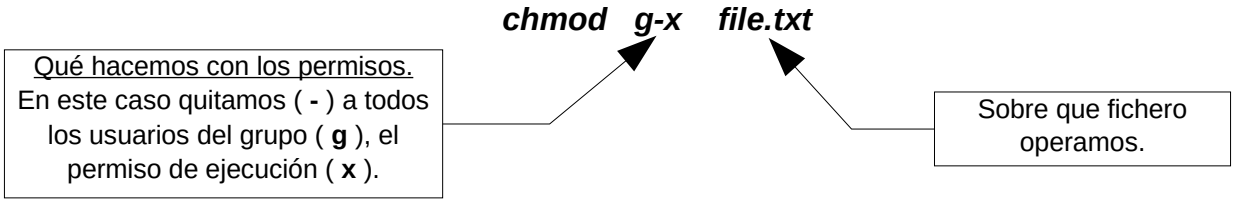
Letra	Significado
Letras que hacen referencia al usuario al cual le vamos a cambiar los permisos.	
u	user , **propietario** del fichero o directorio.
g	group, **grupo.**
o	other , **otros.**
a	all , a **todos** los usuario es decir a : **u, g , o**
Letras relativas a los permisos.	
r	read, **lectura.**
w	write, **escritura.**
x	execute, **ejecución**.
Símbolos varios.	
+	**Añadir** permiso.
-	**Quitar** permiso.

Una vez ya sabemos el significado de los símbolos que usaremos, veámos como aplicarlos.

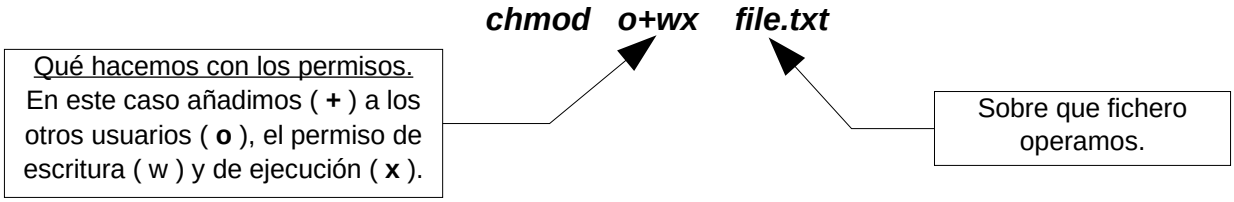
Añadir un permiso.



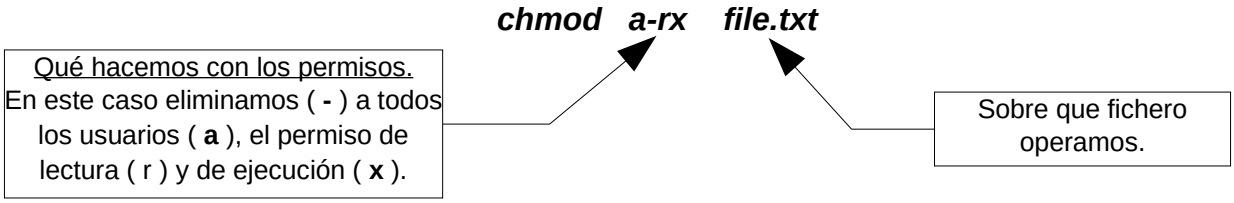
Quitar un permiso.



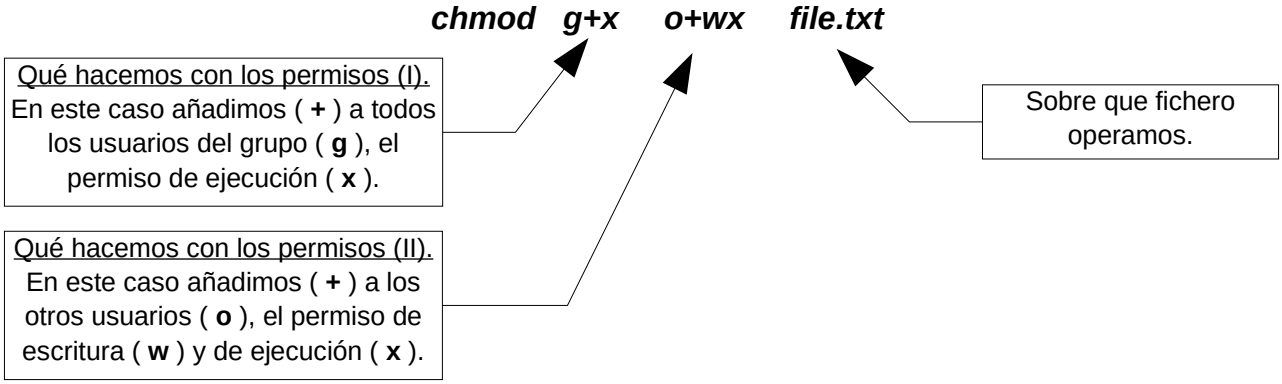
Añadir varios permisos.



Suprimir varios permisos.



Error típico:



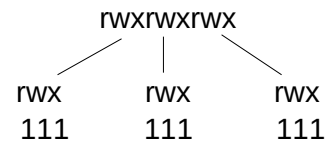
Lo que nos dice la consola :

```
Usuario@MiPC/$ chmod g+x o+wx file.txt
chmod: fallo al obtener los permisos de `o+wx': No existe el fichero o el directorio
```

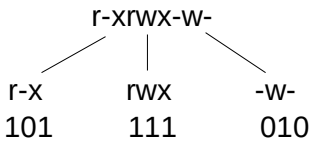
Ha interpretado el segundo conjunto de parámetros "*o+wx*" como el nombre del fichero, por lo que sólo podremos cambiar los permisos de usuario en usuario.

Ahora veámos como podemos hacer los mismo pero que el parámetro *permisos* sea un número.

La idea que se usa en linux es la de asignar a cada permiso un número, un **1** si está activado y un **0** si está desactivado. Veámos un ejemplo donde un fichero tiene todos los permisos activados.

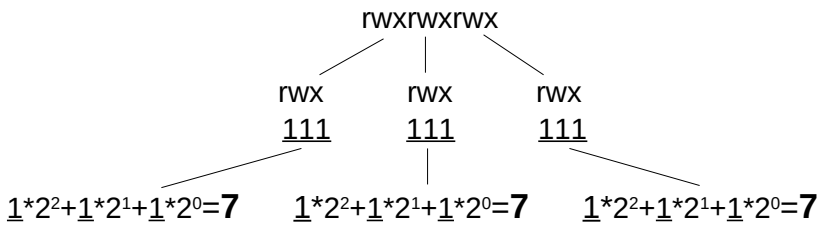


Ahora uno más variado.

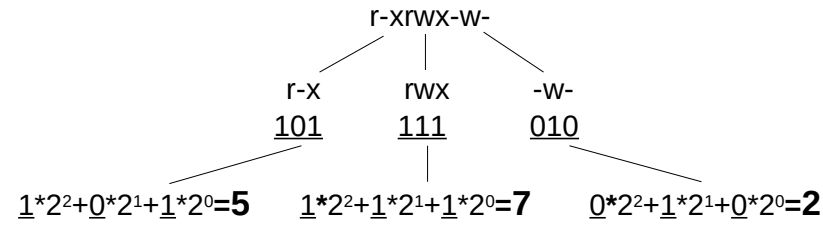


Si para cambiar los permisos fuera necesario escribir esa secuencia de unos y ceros sería demasiado largo, por lo que en Linux se adoptó la siguiente idea : en vez de ponerlos en base binária, ponerlos en decimal, de hecho se usa la base octal, ya que el número más grande que se puede poner es el siete y el rango de valores va de 0 a 7, y que cada número en octal represente a uno de los tres tipos de usuario.

Llegados a este punto uno puede pensar que es un sacrilegio, una locura usar eso pero veámos que con un poco de práctica puede resultar más rápido que usando símbolos.



Pues para cambiar los permisos usaremos : **chmod 777 file.txt** y esto añadirá a todos los usuarios los tres permisos. Veámos otro ejemplo:



Pues para cambiar los permisos usaremos : **chmod 572 file.txt** y esto dejara los permisos de la siguiente manera: el propietario se quedará con los permisos de lectura y ejecución (rx), el grupo con los tres permisos (rwx) y el resto de usuarios se quedan con el de escritura (w).

Para facilitar un poco el uso de los números en la siguiente página hay un listado de las combinaciones más frecuentes.

Permisos	Número	Significado
-rw-------	**600**	Sólo el propietario tiene el derecho de leer y escribir.
-rw-r--r--	**644**	Sólo el propietario tiene los permisos de leer y escribir; el grupo y los demás sólo pueden leer.
-rwx------	**700**	Sólo el propietario tiene los derechos de leer, escribir y ejecutar el fichero.
-rw-rw-rw-	**666**	Todo el mundo puede leer y escribir en el fichero.
-rwxr-xr-x	**755**	El propietario tiene los derechos de leer, escribir y ejecutar; el grupo y los
-rwx--x--x	**711**	El propietario tiene los derechos de lectura, escritura y ejecución; el grupo y los demás sólo pueden ejecutar.
---x--x--x	**111**	Todo el mundo puede ejecutar el fichero.
-r--r--r--	**444**	Todo el mundo puede leer el fichero.
-rwxrwx---	**770**	El propietario y el grupo pueden leer, escribir y ejecutar el fichero, el resto de usuario no peuden acceder al fichero.
-rwxrwxrwx	**777**	Todo el mundo puede leer, escribir y ejecutar.

Tabla de traducción octal – decimal.

Octal	*Decimal*
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

Curiosidad:
"Los ordenadores en el futuro no creo que pesen más de una tonelada y media."
Popular Mechanics, forecasting the relentless march of science, 1949

3.3.6 Cambio de propietario y grupo.

3.3.6.1 Cambio de propietario.

Para cambiar de propietario se usa el comando *chown* (**CH**ange **OWN**er), la sintaxi es la siguiente:

chown ***Usuario*** ***Nombre_del_fichero/Directorio***

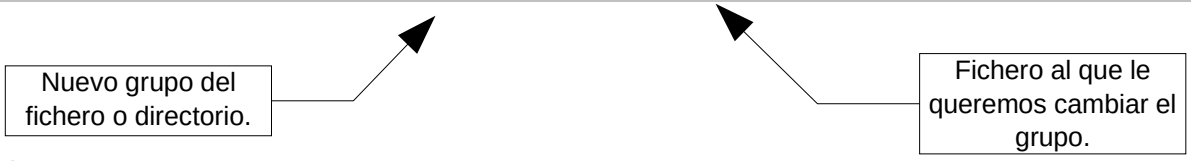
Ejemplo:

```
Usuario@MiPC:/$ su                                          # Nos identificamos como superusuario.
Password: ********
MiPC:/home/prueba# ls -la
total 44
drwxr-sr-x   5      Usuario   staff     4096 2004-08-10 11:46      .
drwxrwsr-x  10      root      staff     4096 2004-08-12 13:37      ..
-rw-r--r--   1      Usuario   staff      279 2004-07-23 18:55      leeme.txt
-rw-r-xrwx   1      Usuario   staff       29 2004-07-28 12:18      nombreNuevo
-rw-r--r--   1      Usuario   staff      192 2004-07-30 13:32      textoParaCowsay
drwxr-sr-x   2      Usuario   staff     4096 2004-08-10 11:46      vacio
MiPC:/home/prueba# chown root leeme.txt                     # Cambiamos el proietario a leeme.txt para que ahora sea el root su nuevo dueño.
MiPC:/home/prueba# ls -la
total 44
drwxr-sr-x   5      Usuario   staff     4096 2004-08-10 11:46      .
drwxrwsr-x  10      root      staff     4096 2004-08-12 13:37      ..
-rw-r--r--   1      root      staff      279 2004-07-23 18:55      leeme.txt
-rw-r-xrwx   1      Usuario   staff       29 2004-07-28 12:18      nombreNuevo
-rw-r--r--   1      Usuario   staff      192 2004-07-30 13:32      textoParaCowsay
drwxr-sr-x   2      Usuario   staff     4096 2004-08-10 11:46      vacio
```

3.3.6.2 Cambio de grupo.

Para cambiar de grupo se usa el comando *chgrp* (**CH**ange **GR**ou**P**), la sintaxi es la siguiente:

chgrp ***grupo*** ***Nombre_del_fichero/Directorio***

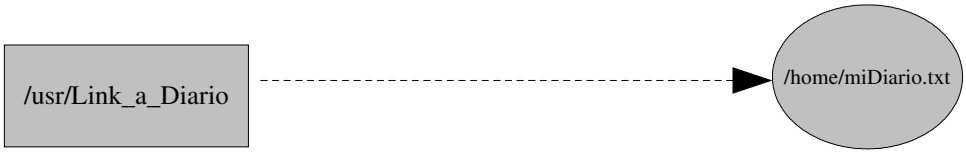


Ejemplo:

```
Usuario@MiPC:/$ su                                          # Nos identificamos como superusuario.
Password: ********
MiPC:/home/prueba# ls -la
total 44
drwxr-sr-x   5      Usuario   staff     4096 2004-08-10 11:46      .
drwxrwsr-x  10      root      staff     4096 2004-08-12 13:37      ..
-rw-r--r--   1      Usuario   staff      279 2004-07-23 18:55      leeme.txt
-rw-r-xrwx   1      Usuario   staff       29 2004-07-28 12:18      nombreNuevo
-rw-r--r--   1      Usuario   staff      192 2004-07-30 13:32      textoParaCowsay
drwxr-sr-x   2      Usuario   staff     4096 2004-08-10 11:46      vacio
MiPC:/home/prueba# chgrp Oficina leeme.txt                  # Cambiamos el grupo a leeme.txt para que ahora sea  Oficina.
MiPC:/home/prueba# ls -la
total 44
drwxr-sr-x   5      Usuario   staff     4096 2004-08-10 11:46      .
drwxrwsr-x  10      root      staff     4096 2004-08-12 13:37      ..
-rw-r--r--   1      Usuario   Oficina    279 2004-07-23 18:55      leeme.txt
-rw-r-xrwx   1      Usuario   staff       29 2004-07-28 12:18      nombreNuevo
-rw-r--r--   1      Usuario   staff      192 2004-07-30 13:32      textoParaCowsay
drwxr-sr-x   2      Usuario   staff     4096 2004-08-10 11:46      vacio
```

3.3.7 Tipos de Ficheros.

EXT2/EXT3 , tienen una particularidad, que para muchos usuarios sin ganas de profundizar en los detalles técnicos les resultará chocante, Linux (por consiguiente EXT2/EXT3) considera fichero a casi todo, por ejemplo, trata como si fuera un fichero los directorios, los podríamos considerar ficheros que contienen ficheros, trata como fichero a los dispositivos, tanto impresoras como cámaras digitales, incluso otras unidades las trata como ficheros, de hecho las trata como directorios, por consiguiente ficheros que contienen ficheros que pueden ser directorios a su vez. Como podemos ver, la importancia del sistema de ficheros en Linux va más allá del simple almacenamiento de documentos. Pues otra maniobra sobre ficheros que nos permite EXT2/EXT3 se basa en los tipos de ficheros que nos dejan crear, veámos la lista.

TIPO	Descripción
Fichero plano	La idea más general, desde ficheros ascii, hasta ejecutables pasando por dibujos, música digital , etc ..
Directorio	Fichero que agrupa ficheros “dentro” de él, sirve para ordenar de forma inteligible para el usuario, los ficheros.
Link	Fichero que representa a otro fichero, cuando se accede al link, en realidad, se accede al fichero apuntado por ese link.  Link fichero de texto Enb la siguiente secuencia de comandos se obtiene siempre la misma respuesta entre ellos : Usuario@MiPC /home$: cat miDiario.txt Día 12 : Fui a comprar el coche ... Día 13 : Aún no tengo el seguro ... Usuario@MiPC /usr$: cat Link_a_Diario Día 12 : Fui a comprar el coche ... Día 13 : Aún no tengo el seguro ...

3.3.7.1 Creación de Links.

Para crear links entre ficheros se usa el comando *ln* (**L**i**N**k), la sintaxi de ln es la siguiente:

ln *fichero_Fuente nombre_del_Link*

Veámos un ejemplo:

```
Usuario@MiPC:/home/prueba$ ls
diario  dibujos  FicheroContodosLosPermisos  nombreNuevo     vacio
diario2  docs    leeme.txt                   textoParaCowsay

Usuario@MiPC:/home/prueba$ cd dibujos/                      # Dentro de dibujos crearemos un link al fichero diario
Usuario@MiPC:/home/prueba/dibujos$ ln ../diario Link_A_diario
Usuario@MiPC:/home/prueba/dibujos$ ls
Link_A_diario
Usuario@MiPC:/home/prueba/dibujos$ more Link_A_diario       # Comprobamos que acceder al link, es lo mismo que
                                                            # acceder al fichero apuntado por ese link

--------------------------------
ESTE ES EL CONTENIDO DEL FICHERO
diario
--------------------------------

DIA 12:
      Iré.
DIA 13:
      No Fui.
DIA 14:
      Iré.
DIA 15:
      Volví a no ir.
DIA 16:
      Al final sí que fuí.

--final del fichero diario--
Usuario@MiPC:/home/prueba/dibujos$ cd ..
Usuario@MiPC:/home/prueba$ more diario

--------------------------------
ESTE ES EL CONTENIDO DEL FICHERO
diario
--------------------------------

DIA 12:
      Iré.
DIA 13:
      No Fui.
DIA 14:
      Iré.
DIA 15:
      Volví a no ir.
DIA 16:
      Al final sí que fuí.

--final del fichero diario--
Usuario@MiPC:/home/prueba$
```

3.4 La ayuda de Linux.

3.4.1 MAN e INFO.

En este apartado vamos a ver de que forma Linux nos ayuda, tanto a resolver problemas como a aprender a usar el sistema en general.

En toda aplicación desarrollada en Linux, es requisisto la generación de una documentación para ayudar al usuario, para acceder a esa documentación se pueden usar dos comandos en la consola : *man* e *info* .Notar que puede darse el caso en que las dos órdenes den información diferente de un mismo programa, por lo que es útil consultar las dos. La sintaxi de uso es la siguiente:

man *nombre_del_programa*
info *nombre_del_programa*

Una vez introducido el comando se nos mostrará por pantalla dicha información sobre el programa, dado que el desarrollo de Linux es internacional, es muy probable que la mayor parte de la documentación esté escrita en inglés.

Veámos un ejemplo:

```
Usuario@MiPC:~$ man pwd
Dando formato a pwd(1); aguarde, por favor...

PWD(1)                    User Commands                    PWD(1)
NAME
       pwd - print name of current/working directory
SYNOPSIS
       pwd [OPTION]
DESCRIPTION
       NOTE:  your  shell may have its own version of pwd which will supercede  the version described here. Please refer to your shell's
documentation for details about the options it supports.
       Print the full filename of the current working directory.
       --help display this help and exit
       --version
              output version information and exit
AUTHOR
       Written by Jim Meyering.
REPORTING BUGS
       Report bugs to <bug-coreutils@gnu.org>.
COPYRIGHT
       Copyright (C) 2003 Free Software Foundation, Inc.
       This is free software; see the source for copying conditions.  There is
       NO warranty; not even for MERCHANTABILITY or FITNESS FOR  A  PARTICULAR
       PURPOSE.
SEE ALSO
       The  full  documentation for pwd is maintained as a Texinfo manual.  If the info and pwd programs are properly installed at your site, the
command  info pwd   should give you access to the complete manual.

pwd 5.0.91                October 2003                    PWD(1)
```

Ahora usemos el comando info para acabar de dcumentar la órden pwd.

```
Usuario@MiPC:~$ info pwd
File: coreutils.info,  Node: pwd invocation,  Next: stty invocation,  Up: Working context
`pwd': Print working directory
==============================
`pwd' prints the fully resolved name of the current directory.  That is, all components of the printed name will be actual directory names--none will
be symbolic links.
  Because most shells have a built-in command by the same name, using the unadorned command name in a script or interactively may get you
different functionality than that described here.
  The only options are alone `--help' or `--version'.  *Note Common options:: .

Usuario@MiPC:~$
```

3.4.2 Metodología de resolución de problemas.

Para afrontar futuros problemas plantearemos ahora una metodología válida para salir airosos de una situación comprometida. En este punto es donde el uso de internet es indispensable, ya que es ahí donde está la fuente del conocimiento sobre Linux.

Conceptos previos.

En Linux cuando algo no funciona se nos suele proporcionar cierta informacion relativa al elemento que ha fallado, ésta generalmente se nos presenta en la consola, en forma de mensaje de error, no los menospreciemos, nos pueden salvar la vida en más de una ocasión.

Véamos un ejemplo:

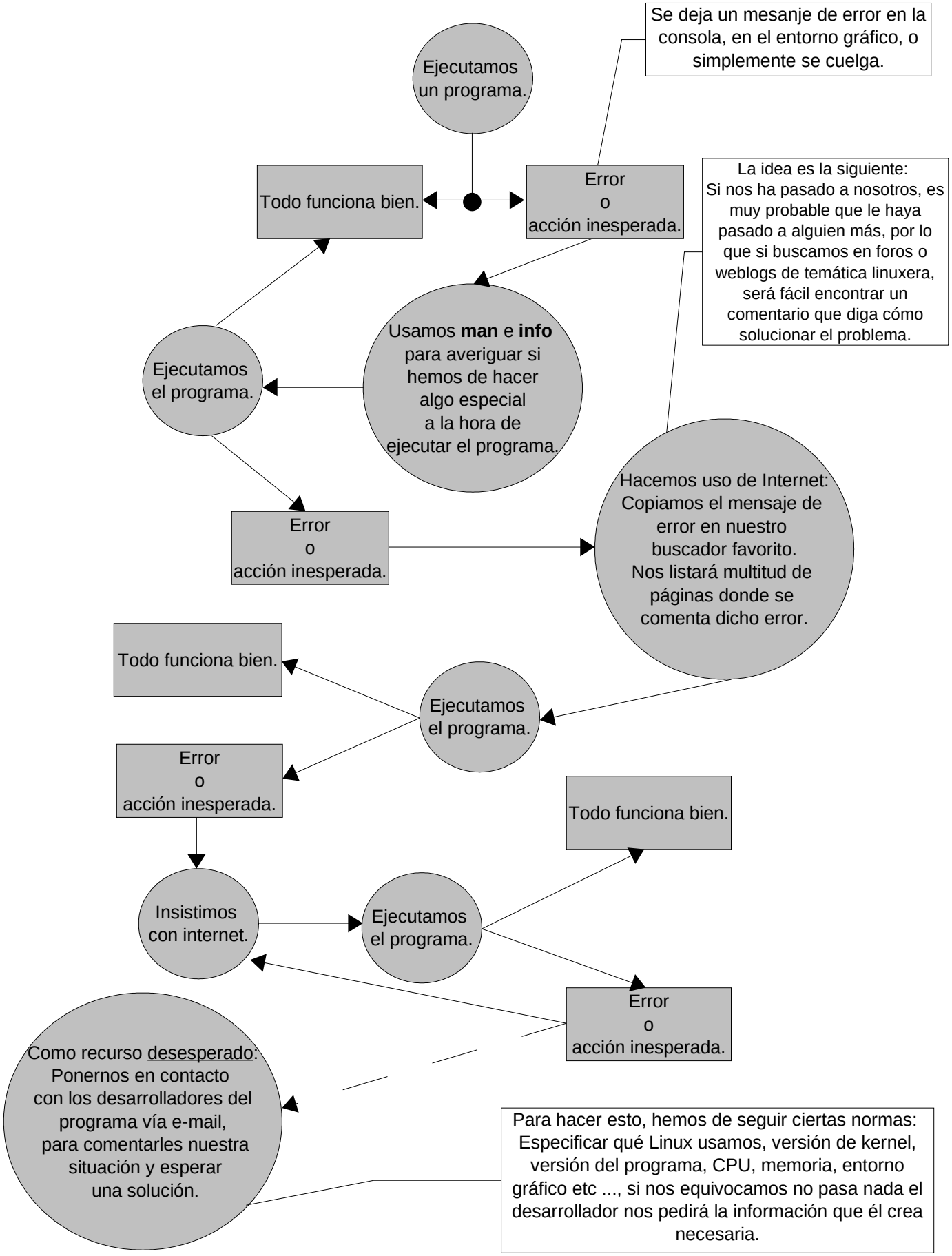
Vamos a ejecutar un programa, por ejemplo un editor de texto sencillo llamado *nedit* :

```
Usuario@MiPC:~$ nedit &
XmFontListCreate() is an obsolete function!
Usuario@MiPC:~$
```

En la consola ha aparecido un mensaje de advertencia de que algo ha fallado, pues es ese tipo de mensajes que hemos de tener en cuenta a la hora de aplicar el procedimiento de resolución de problemas.

Antes de pasar al esquema y para relajarnos un poco, veámos una tira cómica que nos explica la cualidad principal del usuario de Linux, la **PACIENCIA**.

Procedimiento de Resolución de problemas.



3.4.3 Ayuda Online.

Dirección.	*Contenido.*
http://www.frikis.org/ * * * *	Configurar y compilar el Kernel. Grabadoras de CD en Linux. Configurar ADSL y cable-modem. Configurar la impresora. Conectar a internet con modem. Configurar tarjetas graficas Nvidia. Configurar el ratón en consola y en X. Configurar un ratón USB. Ripeando DVDs a Divx con mencoder (mplayer). Pasando Divx a VCD, SVCD, CVCD etc. Divx y DVD a VCD con mencvcd. Imagen en el arranque de Linux con Frame Buffer. TV en Linux. Escaners en Linux. Pasar de ext2 a ext3. Aceleración DRI para 3dfx, 3D labs, ATI, Intel y Matrox. Cómo grabar múltiples VCDs/AVIs en un DVD. Cámaras digitales en Linux. Configurar un lcd usb. Configurar ProFTPd. IPPL. Configuración de Intel PRO/Wireless 2100.
http://www.linuxparatodos.net/ * * *	Preparativos para instalar GNU/Linux®. Cómo crear un disquete de arranque. Cómo crear cuentas de usuario. Breve lección de comandos básicos. Breve lección de actualización, instalación y desinstalación de software en Linux®. Cómo actualizar el Kernel a partir de paquetes RPM®. Cómo configurar correctamente los parámetros de red. Cómo configurar una Conexión ADSL. Introducción a APT para RPM. Cómo actualizar el sistema utilizando yum y apt-get. Las muchas opciones para instalar software en distribuciones basadas sobre RPM. Servidores Cómo configurar vsftpd (Very Secure FTP Daemon). Cómo configurar OpenSSH. Cómo configurar el sistema para sesiones gráficas remotas. Cómo configurar un escáner en red. Servidores de Archivos. Cómo configurar un servidor NFS. Cómo configurar SAMBA. Servidor de correo. Cómo Configurar Postfix 1.1.x con SASL. Cómo configurar Sendmail para redes corporativas. Apache. Cómo configurar Apache para ejecución de guiones CGI. Cómo instalar correctamente Java a partir de paquete RPM. Cómo instalar correctamente los controladores de NVidia. Cómo instalar la extensión (plug-in) Flash para Mozilla. OpenOffice.org

Dirección.	*Contenido.*
http://bulma.net/ *** * ***	Artículos muy variados.
http://es.tldp.org/ *** ***	Tutoriales muy variados, además de mucha información sobre software libre.
http://www.hospedajeydominios.com/mambo/documentacion-manual_linux.html *** * * * ***	Todas las preguntas más frecuentes sobre instalación, configuración y actualización de Linux. **MUY BUENO.**
http://www.insflug.org/ *** * * 1/2**	Lleno de *How to's* , desde configurar entorno gráfico hasta servidores FTP.

Curiosidad:
"Creo que hay una cuota de mercado mundial para unos cinco ordenadores."
Thomas Watson, IBM, 1943.

Licencia de Uso y Distribución.

creative commons
COMMONS DEED

Attribution-NonCommercial-ShareAlike 2.0

Puedes hacer:

- Copiarlo, distribuirlo, enseñarlo, mejorar la obra.
- Hacer obras derivadas.

Pero bajo las siguientes condiciones :

Citar al autor original de la obra.

No uses este trabajo con **fines comerciales**.

Las obras derivadas se han de distribuir bajo estas **mismas condiciones**.

- Las restricciones mencionadas pueden ser omitidas con el **EXPRESO** permiso del propietario del copyright.
- El texto legal se encuentra en :

http://creativecommons.org/licenses/by-nc-sa/2.0/legalcode

Software empleado en el desarrollo:

OpenOffice.org 1.1

the gimp 2.0

debian

Linux
Registered User #328276

Ibán Martínez Gómez
nnset@spymac.com