4. El sistema de ficheros.

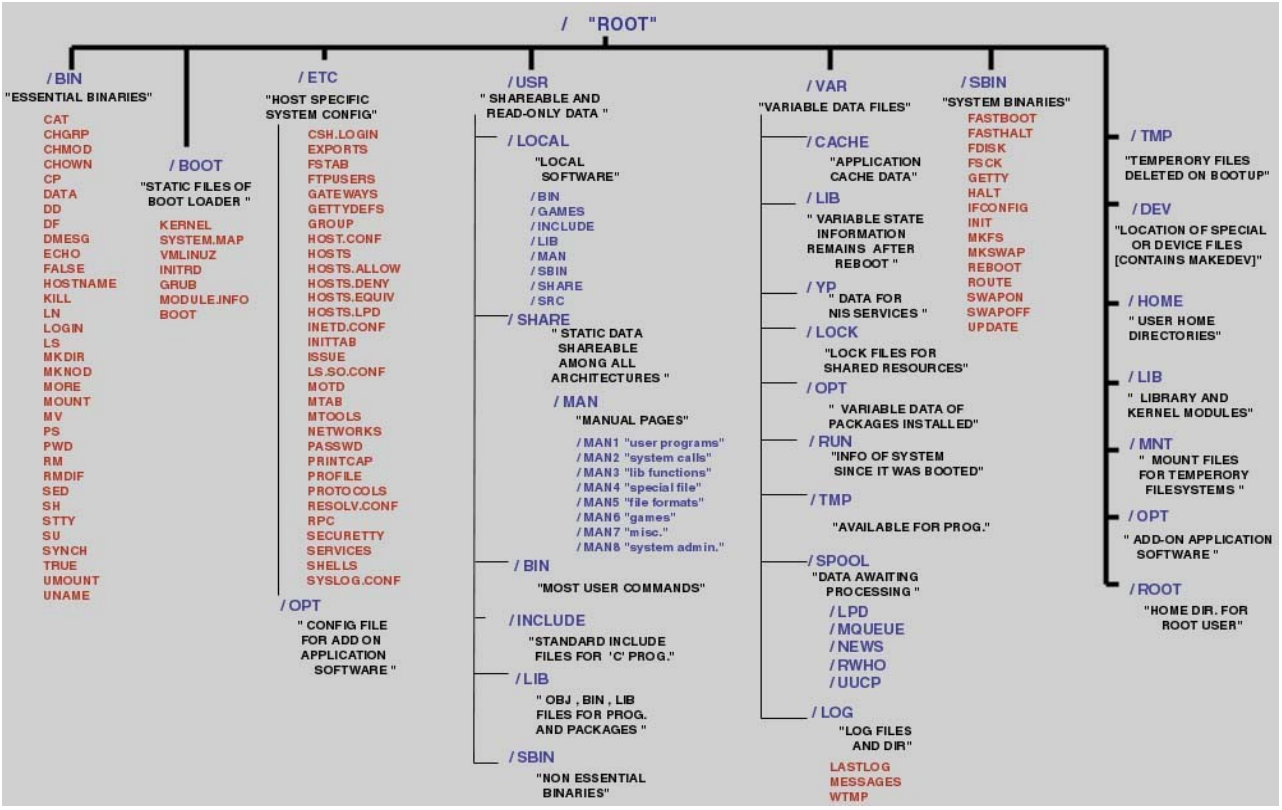
4.1. Jerarquía del sistema de ficheros.

Todo sistema operativo necesita guardar multitud de archivos: desde los de la configuración del sistema, los de log, los de los usuarios, etc. En general, cada sistema operativo utiliza su propio sistema de ficheros, caracterizándolo en muchos aspectos como pueden ser el rendimiento, la seguridad, la fiabilidad, etc.

GNU/Linux es capaz de leer/escribir archivos con cualquiera de los sistemas de ficheros que actualmente existen, aunque para su propia raíz y directorios principales es necesario un sistema de ficheros que le permita ciertas operaciones. Generalmente, se suele utilizar el tipo ext3. El ext3 es el más típico y extendido. Su rendimiento es bastante bueno, incorpora todo tipo de mecanismos de seguridad, es muy fiable e incorpora una tecnología llamada de *journaling*. Una de las principales ventajas de esta tecnología es que si hay un corte en el suministro de energía y el ordenador se apaga sin cerrarse adecuadamente, los sistemas de recuperación de ficheros son más efectivos.

Una característica muy importante de todos los sistemas operativos basados en UNIX es que todos los dispositivos del sistema se pueden tratar como si fueran ficheros. Igualmente, cuando queramos acceder al contenido de un CD, disquete o cualquier otro dispositivo de almacenamiento, deberemos montarlo en un directorio ya existente en el sistema y navegaremos por él como si se tratara de una carpeta más.

Lo primero que debemos tener claro es que todo el sistema de ficheros parte de una misma raíz, a la cual nos referiremos con el carácter "/ ". Es el origen de todo el sistema de ficheros y sólo existe una. Para organizar los ficheros adecuadamente, el sistema proporciona lo que llamaremos directorios (o carpetas), dentro de las cuales podemos poner archivos y más directorios. De este modo conseguimos una organización jerárquica como la que vemos en la siguiente figura:



4.2 Directorios del sistema.

La mayoría de los sistemas operativos del mercado siguen el estándar FHS (http://www.pathname.com/fhs/), donde se especifican las principales características que debería tener cualquier sistema oprativo. Entre ellas está la distribución en directorios que tenemos que hacer de nuestros archivos para tenerlos correctamente organizados y poder localizarlos de forma rápida y sencilla. En la mayoría de distribuciones basadas en GNU/Linux se siguen estas recomendaciones, encontrando los siguientes directorios principales:

- `/bin/`: comandos básicos para todos los usuarios del sistema.
- `/boot/`: archivos estáticos necesarios para el arranque del sistema.
- `/dev/`: dispositivos del sistema.
- `/etc/`: archivos de configuración del sistema y de las aplicaciones instaladas en el mismo.
- `/home/`: directorio para poner las carpetas *home* de los usuarios.
- `/lib/`: librerías esenciales para el núcleo del sistema y módulos del mismo.
- `/mnt/`: punto de montaje temporal para dispositivos.
- `/proc/`: procesos y variables del núcleo del sistema.
- `/root/`: directorio *home* para el *root* del sistema.
- `/sbin/`: comandos especiales para el *root* del sistema.
- `/tmp/`: archivos temporales. Según la distribución utilizada (o la configuración que utilicemos) se borran al arrancar el sistema o cada cierto período de tiempo.
- `/usr/`: segunda estructura jerárquica, utilizada para almacenar todo el software instalado en el sistema.
- `/var/`: directorio para los s*poolers* de impresión, ficheros de log, etc.

Es muy recomendable conservar y no eliminar ninguno de estos directorios (o los que por defecto nos crea la distribución que utilizamos), ya que son básicos para el buen funcionamiento del sistema. Generalmente, los procesos de instalación de nuevas aplicaciones necesitan que exista la organización dada y muchos de los archivos de configuración de los programas deben estar en determinados directorios. Lo que sí que podemos hacer sin ningún tipo de restricción es crear nuevos directorios en la raíz del sistema o en cualquier otra carpeta.

4.3. Moviéndonos por los directorios

Para movernos por la estructura de directorios debemos utilizar los comandos para listar contenidos y cambiar de carpeta. Cuando entramos en el sistema, es usual que el login nos sitúe en nuestro directorio home, que generalmente se suele referenciar con el carácter "~". Si queremos ver lo que hay en el directorio donde estamos situados, podemos listar los contenidos utilizando el comando ls. Debemos tener en cuenta que por defecto el comando no nos muestra los archivos que empiezan por un punto. Con el parámetro "-a" sí que nos mostraría absolutamente todos los ficheros. En todos los directorios existe una entrada "." y otra "..". El punto es la referencia al directorio actual, mientras que los dos puntos seguidos hacen referencia al directorio inmediatamente superior (en el árbol de jerarquías) al actual. Naturalmente, cuando estamos situados en la raíz del sistema de ficheros, la entrada ".." no existirá porque nos encontramos en el nivel superior.

Para cambiar de directorio podemos utilizar el comando `cd`. Si no le pasamos ningún parámetro, por defecto nos situará en nuestro directorio *home*. Generalmente, se le suele indicar dónde queremos ir, pasándolo de forma absoluta o relativa. De forma relativa significa que partiremos del directorio donde estamos en el momento de ejecutar el comando. Por ejemplo, si estamos en el directorio `/usr/ bin/` y queremos ir al `/root/`, deberíamos introducir el siguiente comando: "`cd ../../root`" (los dos primeros puntos indican `/usr/` y los siguientes la raíz "/" del sistema, a partir de la cual ya podemos acceder a `/root/`). De forma absoluta siempre partimos de la raíz, de manera que el comando que utilizaríamos para el ejemplo anterior sería: "`cd /root`". Para saber en qué directorio estamos, podemos utilizar el comando `pwd`.

4.4. Enlaces.

Otros mecanismos que nos proporcionan la gran mayoría de sistemas de ficheros son los que llamamos *enlaces*. Un enlace es un puente a un archivo o directorio perteneciente al sistema; una referencia que podemos poner en cualquier sitio que nos interese y que actúa como un acceso directo a cualquier otro. Este mecanismo nos permite acceder a carpetas o archivos de forma más rápida y cómoda, sin tener que desplazarnos por la jerarquía de directorios.

Vamos a verlo con un ejemplo: imaginemos que somos un usuario (user1) que necesita acceder frecuentemente al directorio /usr/share/man/man3/. En lugar de escribir el largo comando que nos situaría en el directorio en cuestión cada vez que necesitáramos desplazarnos a él, podemos crear un enlace en nuestro propio directorio que nos redireccione directamente hacia allí.

El comando "ln -s /usr/share/man/man3 mmm" nos crearía este puente, que hemos llamado "mmm". El usuario sólo debería escribir (desde su directorio *home*) "cd mmm" y automáticamente el sistema lo redirigiría hacia /usr/share/man/man3/. Es importante tener en cuenta que al hacer un "cd .." para ir al directorio superior, volveríamos al directorio *home* y no a usr/share/man/, ya que hemos accedido a él a partir de nuestro enlace.

Al crear el enlace del ejemplo anterior hemos pasado el parámetro "-s" al comando. Ello indica que queremos crear un enlace simbólico. Los enlaces simbólicos significan que sólo estamos creando un apuntador o puente hacia el fichero o directorio, de forma que si borrásemos el fichero destino, el enlace no apuntaría a ninguna parte. Si no ponemos el parámetro "-s" se crearía lo que llamamos un enlace fuerte (*hard link*) que, a diferencia del anterior, hace un duplicado del fichero. De hecho, internamente no es exactamente un duplicado, es como dos entradas que apuntan a los mismos datos. De este modo, si modificamos uno u otro, los dos quedan iguales. La ventaja de este tipo de enlace es que si borramos cualquiera de las dos copias del fichero la otra todavía se conserva. Este tipo de enlace no se utiliza demasiado porque complica la gestión y manipulación de los ficheros (siempre es mejor tener una sola copia de los archivos). Además, si hacemos un enlace fuerte de un directorio, todos los archivos y subdirectorios que contuviera también se deberían referenciar. Por esta razón sólo el *root* del sistema puede hacer enlaces fuertes de directorios. Otra diferencia es que con un enlace simbólico podemos ver a qué fichero estamos apuntando, mientras que con uno fuerte no podemos (debido al mecanismo que se utiliza internamente para ellos).

4.5. Permisos.

En cualquier sistema operativo multiusuario necesitamos que los ficheros que guardamos en nuestro disco puedan tener una serie de propiedades que nos permitan verlos, modificarlos o ejecutarlos para los usuarios que nosotros definamos. Aunque hay varias alternativas para hacer esto, GNU/Linux utiliza el sistema clásico de UNIX, que, combinado con todos los mecanismos de gestión de usuarios y grupos, nos permite cualquier configuración posible. Lo que interesa es definir, para cada fichero o directorio, a qué usuario y grupo pertenece y qué permisos tiene para cada uno de ellos, así como para el resto de usuarios del sistema. Ejecutando "ls -l" veremos cómo en cada archivo del directorio donde estamos aparece una línea parecida a la siguiente:

```
-rwxr-xr-x 1 user1 grupo1 128931 Feb 19 2000 gpl.txt
```

Los primeros diez caracteres (empezando por la izquierda) nos indican los permisos del fichero de la siguiente manera:

- Carácter 1: esta entrada nos indica si es un fichero o un directorio. En caso de ser un fichero, aparece el carácter "-", mientras que por los directorios aparece una "d".

- Caracteres 2, 3, 4: nos indican, respectivamente, los permisos de lectura, escritura y ejecución para el propietario del fichero. En caso de no tener el permiso correspondiente activado, encontramos el carácter "-" y si no "r", "w " o "x", según si lo podemos leer (*Read*), escribir (*Write*) o ejecutar (*eXecute*). En el tercer carácter, además, podemos encontrarnos una "s", que nos indica si el archivo es de tipo SetUserId, que quiere decir que al ejecutarlo obtendrá los permisos del propietario del fichero. Si sólo tiene el permiso "x", cuando el programa se ejecuta lo hace con los permisos de quien lo haya lanzado.

- Caracteres 5, 6, 7: estos caracteres tienen exactamente el mismo significado que anteriormente, pero hacen referencia a los permisos concedidos a los usuarios del grupo al que pertenece el fichero.

- Caracteres 8, 9, 10: igual que en el caso anterior, pero para los otros usuarios del sistema.

- Después de estos 10 caracteres encontramos una cifra que nos indica el número de enlaces fuertes que tiene el fichero. Para los directorios, este número indica cuántas carpetas hay dentro de él además de los enlaces fuertes que tiene (cuando no hay ninguno, el número es 2, debido a la

gestión interna del operativo). A continuación vemos el propietario y el grupo del archivo, seguido del tamaño (en *bytes*) que ocupa y la fecha de la ultima modificación. En todos los ficheros se guarda su fecha de creación, último acceso y modificación, que podemos manipular con el comando touch. Al final hay el nombre del fichero, dónde se diferencian minúsculas de mayúsculas y podemos tener todo tipo de caracteres sin ningún problema.

Para cambiar los permisos de un determinado archivo podemos utilizar el comando chmod. Debemos tener en cuenta que sólo el propietario del archivo (o el *root*) puede cambiar estos permisos, ya que si no, el mecanismo no tendría ningún sentido. Podemos utilizar este comando de muchas maneras diferentes, pero las dos más frecuentes son las siguientes:

- El primer modo de utilizarlo es del estilo "chmod XXX nombreArchivo". Las "X " deben ser tres números entre 0 y 7. El primer número indica los permisos que queremos establecer para el usuario, el segundo, para el grupo y el tercero, para el resto. Para interpretar correctamente los permisos que daremos utilizando los números del 0 al 7, debemos hacer uso de la representación binaria del número en cuestión, de forma que el primer dígito indicará el permiso de escritura, el segundo, el de lectura y el tercero, el de ejecución. En cada caso un 0 indica que no se da el permiso en cuestión y el 1 indica que sí que se da. En la siguiente tabla podemos ver esta relación:

Representación decimal	Representación binaria	Significado
0	000	—
1	001	`--x`
2	010	`-w-`
3	011	`-wx`
4	100	`r--`
5	101	`r-x`
6	110	`rw-`
7	111	`rwx`

- El otro modo de utilizar el comando es indicando de forma explícita qué permiso queremos dar o eliminar del fichero. La manera de hacerlo es indicando, primero, si nos referimos a los permisos del usuario, grupo o al resto con las letras "u", "g" u "o" respectivamente. Seguidamente, debemos añadir un "+" o "- " según si queremos añadir o eliminar el atributo, que indicaremos con "r", "w",

"x " o "s" (este último para el SetUserId). Además, podemos hacer todas las combinaciones posibles, refiriéndonos a más de un permiso y/o usuarios. Por ejemplo, "chmod go+r gpl.txt"daría el permiso de lectura al grupo y a los otros usuarios para el fichero gpl.txt.

Para cambiar el propietario de un fichero existe el comando chown, que sólo puede utilizar el *root* por razones de seguridad. Para cambiar el grupo de un determinado archivo, se puede utilizar el comando chgrp. Como podemos suponer, cuando un usuario crea un nuevo archivo, el sistema pone como propietario al usuario que lo ha creado y lo da como perteneciente al grupo primario del mismo usuario. Los permisos que se ponen por defecto al crear un nuevo archivo los podemos configurar con el comando umask, al que debemos pasar la misma notación de tres números decimales entre 0 y 7 que veíamos anteriormente pero complementados. Por ejemplo, si queremos que nuestros ficheros se inicialicen con los permisos "rw-r--r--", deberíamos escribir "umask 133".

4.6. Manipulación de ficheros y directorios.

Ahora que ya sabemos movernos correctamente por la jerarquía de directorios, también necesitamos saber cómo copiar, eliminar y manipular correctamente otros aspectos de los ficheros. El comando rm es el que se encarga de eliminar los archivos que le indiquemos. Para eliminar un directorio, podemos utilizar el comando rmdir, aunque sólo lo borrará cuando éste esté vacío (si quisiéramos borrar completamente un directorio y todo su contenido, podríamos utilizar "rm -r"). Para copiar archivos de un lugar a otro tenemos el comando cp, al que siempre debemos indicar el fichero o directorio origen y el lugar o nombre de destino, aunque sea en el directorio actual. De este modo, si queremos copiar el archivo /home/user1/gpl.txt en el directorio actual (y con el mismo nombre) deberíamos escribir "cp /home/user1/gpl.txt .". Si en lugar de copiar los archivos queremos moverlos de sitio, podemos utilizar el comando mv.

Un mecanismo muy útil que nos proporciona el sistema son los *patterns* ('patrones'). Hasta ahora hemos visto cómo aplicar ciertas operaciones sobre un determinado archivo. Cuando estamos manipulando un sistema, en muchos casos nos interesará aplicar alguna de las operaciones que hemos visto pero sobre un grupo grande de ficheros. Los patrones nos permitirán aplicar las operaciones que queramos especificando en una sola instrucción varios ficheros que cumplan con una serie de características concretas. Debemos verlos como plantillas de nombres, de manera que el carácter "*" significa cualquier cadena de caracteres posibles y el "?" nos sirve como comodín a cualquier carácter. De este modo, si queremos listar todos los archivos que empiecen por "s", que después tengan cualquier otro carácter, les siga una "a", y después cualquier otra cadena, podríamos utilizar "ls s?a*". Entre "[]" podemos incluir otros caracteres, indicando que el patrón tiene éxito si se encuentra alguno de ellos en el nombre. Por ejemplo, si quisiéramos referenciar todos los archivos que empiecen por "a" o por

"b" y que continúan con cualquier otra cadena, podríamos escribir el *pattern* "[ab]*". Si después de "[" pusiéramos el carácter "! " ("[!ab]*") indicaríamos que el *pattern* coincide con cualquier archivo que no empiece por "a " o "b".

Naturalmente, los *patterns* los podemos utilizar con cualquiera de los comandos que hemos visto y la mayoría de los que veremos a continuación. Además, la mayor parte de los comandos de listado, eliminación, copia, etc. de ficheros también permiten que se les pase un parámetro (generalmente "-r") para realizar las acciones respectivas de forma recursiva. De este modo, se irá entrando y ejecutando la instrucción correspondiente en todos los archivos y directorios, a partir de donde nos encontramos y hasta llegar al último nivel de la jerarquía.

Otro tipo de operación muy útil es la búsqueda de ficheros. Tenemos varios comandos que nos permiten realizar búsquedas de diferentes tipos sobre todos los ficheros del sistema.

`find`	Es el comando más versátil para realizar esta acción. Nos permite filtrar los ficheros para encontrar desde los que tienen un determinado nombre, los modificados o creados a partir de una cierta fecha, los que tienen ciertos permisos, etc. Su única desventaja es que no utiliza ningún tipo de mecanismo para acelerar la búsqueda, con lo cual, éstas pueden tardar bastante.
`locate`	Se trata de otro comando, pero, a diferencia del anterior, utiliza una base de datos interna que se actualiza periódicamente y nos permite hacer búsquedas bastante más rápidas. Debemos tener en cuenta, sin embargo, que los resultados no siempre estarán actualizados, además de que no podemos realizar búsquedas tan versátiles como con find.
`whereis`	Por último, whereis está orientado a la búsqueda de los archivos binarios (los ejecutables), de ayuda o los de código fuente de un determinado programa.

4.7. Tipos y contenido de ficheros.

Los archivos que tenemos en nuestro sistema pueden ser de muchos tipos diferentes: ejecutables, de texto, de datos, etc. A diferencia de otros sistemas que utilizan la extensión del archivo para determinar de qué tipo son, GNU/Linux utiliza un sistema denominado de magic numbers, determinando con un número mágico el tipo de fichero según sus datos (se pasan una serie de tests que intentan determinar de qué tipo es el fichero). El comando file nos lo indica.

Si necesitamos ver el contenido de un fichero, uno de los comandos básicos es el cat. Pasándole el nombre o nombres de los archivos que queremos ver, se muestra por pantalla. Debemos intentar no mostrar ficheros ejecutables o de datos por pantalla, ya que el volcado de caracteres no imprimibles nos dejaría la consola con carateres no comprensibles (siempre la podemos reiniciar tecleando reset o tset). Para ficheros muy extensos, nos irán mucho mejor los comandos less o more, que permiten desplazarnos por el fichero de forma progresiva. Si el tipo de fichero es binario y queremos ver qué contiene, podemos utilizar los comandos hexdump u od para ver el contenido de forma hexadecimal u otras representaciones. strings nos buscará las cadenas de caracteres dentro de un fichero binario y las mostrará por pantalla.

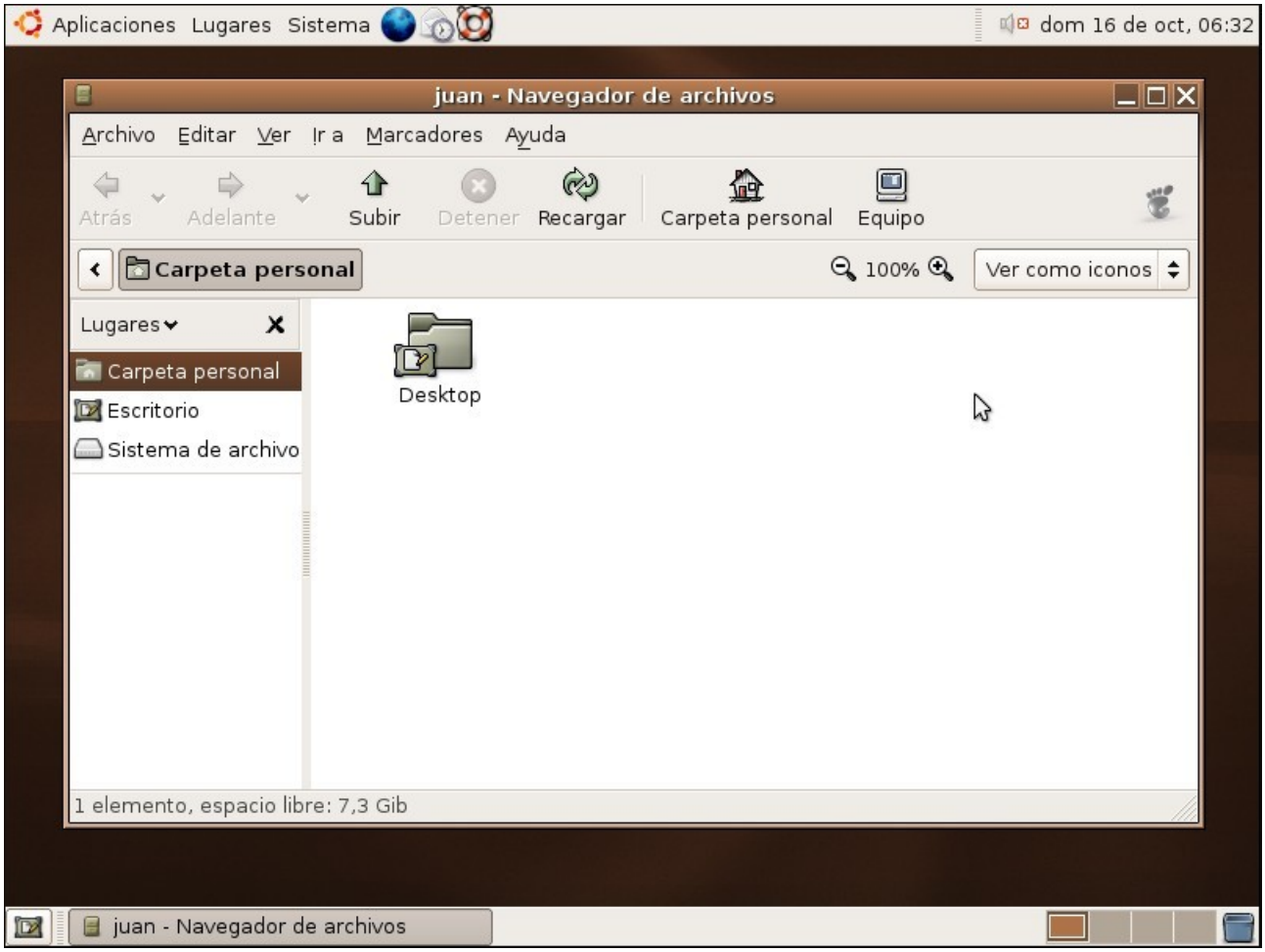
Otro tipo de comandos muy útiles son los que nos buscan un cierto patrón en el contenido de los ficheros. Con el comando grep le podemos pasar como segundo parámetro el nombre del archivo y como primero el *pattern* que queramos buscar (con la sintaxis que veíamos anteriormente, extendida a otras opciones). Además, el comando nos permite otras múltiples acciones, como contar el número de líneas donde aparece el patrón (parámetro "-c"), etc. Con cut podemos separar en campos el contenido de cada línea del fichero especificando qué carácter es el separador, muy útil en tareas de administración del sistema para su automatización. También podemos coger un determinado número de líneas del principio o fin de un archivo con los comandos head y tail respectivamente. Con wc podemos contar el número de líneas o palabras, la máxima longitud de línea de un fichero, etc.

Finalmente, para acabar con esta sección de manipulación de ficheros, lo único que nos falta por ver es cómo comparar diferentes archivos. Igual que con las otras operaciones, tenemos varios comandos que nos permiten hacerlo. diff, cmp y comm realizan comparaciones de diferentes formas y métodos en los ficheros que indicamos. sdiff, además, permite mezclarlos a nuestra elección.

4.7 Navegando con Nautilus

Nautilus es la aplicación que nos permite navegar por los archivos de nuestro disco duro o incluso de otros ordenadores conectados por red.

Podemos abrirlo yendo a “Lugares -> Carpeta personal”.



Vemos que Nautilus esta formado por (de arriba a abajo y de izquierda a derecha):

Menú (Archivo, editar, etc...). Aquí encontraremos todas las acciones que podemos llevar a cabo con el navegador de disco.

Iconos

- Atrás: Volver a la última carpeta visitada según el historial.
- Adelante: Ir a la siguiente carpeta según el historial.
- Subir: Ir a la carpeta que contiene la carpeta actual, por ejemplo si estamos en la carpeta “/home/usuario”, iriamos a “/home”.
- Detener

- Recargar
- Carpeta personal: Ir a nuestra carpeta de usuario.
- Equipo: Mostrar los dispositivos del sistema para poder montarlos/expulsarlos o acceder a su contenido.

Barra de navegación: Muestra en forma de botón toda la jerarquía de carpetas desde la raíz del sistema hasta la carpeta que estamos visualizando. Nos permite movernos a una carpeta superior con un solo click.

Si quisiéramos acceder a la ruta real podríamos pulsar CTRL+L.

En el lado derecho se puede modificar el tamaño con el que queremos que se muestren los archivos, y la forma en la que queremos visualizar dichos ficheros:

• Ver como iconos: Cada archivo y carpeta estará representado por un icono junto a su nombre.

• Ver como lista: Los archivos se mostrarán en forma de listado con información adicional en cada línea.

El listado será en forma de árbol de forma que se nos permite ver el interior de un fichero sin tener que hacer doble click sobre el, simplemente haciendo un click en el triangulo que aparece al lado de cada carpeta.

Panel lateral izquierdo: Nos permite seleccionar que tipo de información queremos que sea mostrada. Podemos elegir entre:

• Lugares: Nos muestra el contenido del menú del sistema "Lugares". Aquí aparecerán enlaces directos a nuestra carpeta personal, servidores y marcadores.
• Información: Muestra información sobre la carpeta actual.
• Árbol: Nos sitúa dentro del árbol de directorios y nos permite cambiar de carpeta fácilmente.
• Histórico: Últimas carpetas que hemos ido visitando.
• Notas: Nos permite añadir notas la carpeta que estamos visualizando.
• Emblemas: Podemos hacer click y arrastrar dichos iconos sobre cualquier carpeta para remarcarlas.