



UD6 Sistema operativo GNU/Linux

Sistemas informáticos

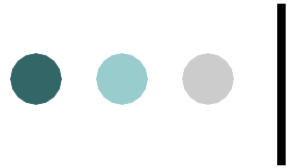


Contenidos

1. [Historia](#)
2. [Distribuciones de Linux](#)
3. [Instalación de Ubuntu](#)
4. [Escritorio](#)
5. [Administrador de archivos](#)
6. [Sistema de ficheros](#)
7. [Shell de Linux](#)
8. [Comandos de gestión de ficheros y directorios](#)
9. [Configuración del sistema](#)
10. [Instalación de software](#)
11. [Arranque y parada del sistema](#)
12. [Particiones](#)
13. [Usuarios y grupos](#)
14. [Permisos](#)
15. [Compartir recursos](#)
16. [Gestión de procesos](#)
17. [Servicios](#)
18. [Programación de tareas](#)
19. [Copias de seguridad](#)



1. Historia



1. Historia

- En la década de los sesenta las grandes compañías de ordenadores **no daban el valor que hoy día se da al software**. En su gran mayoría eran fabricantes de ordenadores que obtenían sus principales ingresos vendiendo sus grandes máquinas.
- Las universidades tenían **permiso para coger y estudiar el código** fuente de los sistemas operativos para fines docentes. Los mismos usuarios podían pedir el código fuente de drivers y programas para adaptarlos a sus necesidades. Se consideraba que **el software no tenía valor por sí mismo** si no estaba acompañado por el hardware que lo soportaba.
- A finales de los 60, los laboratorios Bell de AT&T diseñaron un sistema operativo llamado **UNIX**, caracterizado por la buena gestión de los recursos del sistema, su estabilidad y su compatibilidad con el hardware de diferentes fabricantes. Este último hecho fue importantísimo pues hasta entonces todos los fabricantes tenían sus propios sistemas operativos incompatibles con el resto.



1. Historia

- Poco a poco, las grandes empresas empezaron a tomar conciencia del valor del software: primero fue IBM la que en 1965 **dejó de dar el código fuente** de su sistema operativo, más tarde Digital Research empezó a vender el suyo, etc. Este hecho hizo que todas las compañías se dieran cuenta de que el software **podía ser muy rentable** y les podía aportar grandes beneficios.
- Desde entonces, la mayoría de empresas empezaron a poner reticencias a dejar el código fuente de sus programas y sistemas operativos y empezaron a **vender sus programas** como un valor añadido a su hardware.

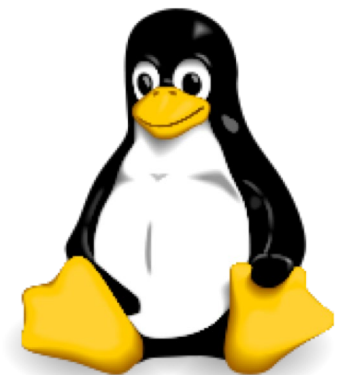


1. Historia

- En este entorno cada vez más cerrado, **Richard Stallman** (que trabajaba en el MIT, Massachusetts Institute of Technology) se sintió indignado al comprobar que cada vez era más difícil conseguir el código fuente de los programas que utilizaba para adaptarlos a sus necesidades, por lo que en 1983 decidió iniciar un gran proyecto para intentar abrir otra vez el código fuente de los programas.
- Consciente de que no podría conseguir que las compañías cedieran en este punto, se propuso crear su propio sistema operativo y aplicaciones iniciando un proyecto llamado **GNU** (GNU is Not Unix).
- En este proyecto empezó a describir el concepto de **software libre** y para qué creía necesario que programadores y desarrolladores de alrededor del mundo contribuyeran con él para conseguir su objetivo.
- En 1985, creó la Fundación del Software Libre (FSF) y desarrolló la Licencia pública general de GNU (GNU GPL).

1. Historia


- En este contexto, un profesor de Universidad en Holanda, Andrew Tanenbaum, decidió escribir un sistema operativo para que sus estudiantes pudieran estudiarlo. Su idea era escribir un sistema operativo sencillo que pudiera ser estudiado y modificado por cualquiera que quisiera.
- En 1987 comenzó el desarrollo y llamó a su proyecto mini UNIX, dando lugar a **MINIX**. Al no utilizar ni una sola línea de código del UNIX de AT&T, no hay ninguna restricción en coger el código, utilizarlo y modificarlo libremente.
- Linus Torvalds, estudiante de la Universidad de Helsinki, inspirado en MINIX, decidió crear en 1991 su propio núcleo de sistema operativo, con el nombre de **Linux**. Su idea era crear un UNIX para PC para que todos los que quisieran lo pudieran utilizar en su ordenador.
- Publicó su idea en un grupo de noticias y rápidamente hubo muchas personas en todo el mundo que ayudaron en su desarrollo.





1. Historia

- En seguida mucha gente se interesó por el código, sobre todo el proyecto GNU (GNU No es UNIX), creado por Richard Stallman.
- En el momento de la publicación de Linux, el proyecto GNU tenía bastantes programas para tener un sistema operativo pero le faltaba el núcleo.
- De esta forma, se unieron los esfuerzos de Torvalds y el proyecto GNU para acabar creando GNU/Linux.



1. Historia

- Hoy en día son las distribuciones las encargadas de distribuir núcleos estables a sus usuarios. Estos núcleos se basan en el núcleo distribuido por Linus Torvalds y el equipo de programadores del núcleo. Este núcleo se conoce como “vanilla” y se puede descargar desde: www.kernel.org.
- Cada distribución coge estos núcleos vanilla y añade sus propias modificaciones. Algunas corrigen algunos fallos particulares del código, añaden modificaciones para hacer cosas curiosas o mejoran las facilidades del kernel (por ejemplo, para soportar dispositivos USB muy nuevos).
- Por tanto, aunque todos los núcleos derivan del mismo núcleo fuente, no todos los núcleos son iguales.



2. Distribuciones de Linux



2. Distribuciones de Linux

- Una distribución Linux es un conjunto de programas que, junto con la última versión del núcleo, están preparados para ser instalados.
- La mayoría de las distribuciones están, en mayor o menor medida, desarrolladas y dirigidas por sus comunidades de desarrolladores y usuarios.
- En algunos casos están dirigidas y financiadas completamente por la comunidad, como ocurre con Debian, mientras que otras mantienen una distribución comercial y una versión de la comunidad, como hace Red Hat con Fedora y CentOS, o SuSe con OpenSuSe.
- Una distribución live es una distribución almacenada en un DVD o en un lápiz de memoria, que permite usar el sistema operativo sin tener que instalarlo.

2. Distribuciones de Linux

- Hay muchas distribuciones, cada una con sus versiones.
- Algunas son Ubuntu, Fedora, Linux Mint, openSUSE, Mandriva, Arch Linux, elementary OS, Tails,...



- También hay distribuciones dirigidas para ser usadas en servidores, como Ubuntu Server, Red Hat Enterprise Linux, Suse Linux Enterprise Server, CentOS, Debian, Mageia o Slackware.

<http://distrowatch.com/>

<http://www.genbeta.com/linux/31-distribuciones-de-linux-para-elegir-bien-la-que-mas-necesitas/>

2. Distribuciones de Linux

- Ubuntu es una de las distribuciones más utilizadas en estaciones de trabajo gracias a su gran facilidad de uso.
- La versión para servidores también está incrementando su demanda.
- Ubuntu está basada en el código de Debian y está desarrollada por la empresa Canonical, que lanzó la primera versión en octubre de 2004. Además tiene el apoyo de una comunidad de desarrolladores voluntarios.
- La empresa ha conseguido que Ubuntu venga preinstalado en PCs, tanto a nivel de particulares como empresarial.





2. Distribuciones de Linux

- Canonical se comprometió a sacar dos versiones al año, en abril y octubre.
- Hay dos tipos de versiones:
 - Normal o estándar: Tienen período de soporte reducido, a veces de 9 meses y otras de 18 meses.
 - LTS (Long Term Support): Tienen un período de soporte más amplio, de 5 años en las ediciones Desktop y Server.
- Durante el período de soporte, se crean parches de seguridad y mantenimiento que solucionan problemas detectados. Lo que no hay son actualizaciones en las aplicaciones incorporadas.
- Para saber los detalles, hay que leer las *Release notes* de la versión.



3. Instalación de Ubuntu



3. Instalación de Ubuntu

- En la actualidad, las distribuciones de Linux disponen de un programa de instalación en modo gráfico que facilita todo el proceso de instalación del sistema.
 - Algunos de los pasos de este proceso son:
 - Asignar una contraseña al superusuario (*root*).
 - Ajustar los parámetros de la red.
 - Definir y crear las particiones. En algunos casos se tiene la opción de realizar esta tarea de forma automática por parte del programa de instalación.
 - Una recomendación general es crear, al menos, las siguientes particiones:
 - /
 - /home
 - swap
 - Pero esto dependerá del uso que se le vaya a dar al equipo.
- <https://help.ubuntu.com/community/DiskSpace>



3. Instalación de Ubuntu

- Los requisitos hardware para instalar Ubuntu son:
 - Procesador: 2 GHz.
 - Memoria RAM: 2 GiB.
 - Espacio disponible en disco: 25 GB.
- Para conocer las novedades de una versión, se pueden leer las *Release notes*:
<https://wiki.ubuntu.com/BionicBeaver/ReleaseNotes>



4. Escritorio



4. Escritorio

1. Entornos de escritorio
2. Escritorio de Ubuntu

4.1 Entornos de escritorio

- En Linux no existe un único entorno gráfico como ocurre en Windows, sino que existen varios, siendo KDE y GNOME los entornos de escritorio más extendidos.
- Es posible tener instalado más de un entorno de escritorio y arrancar el sistema utilizando el que queramos.
- Realmente son sistemas de escritorio que funcionan sobre un servidor gráfico común llamado X-Windows.
- Ubuntu tiene por defecto GNOME.



4.1 Entornos de escritorio

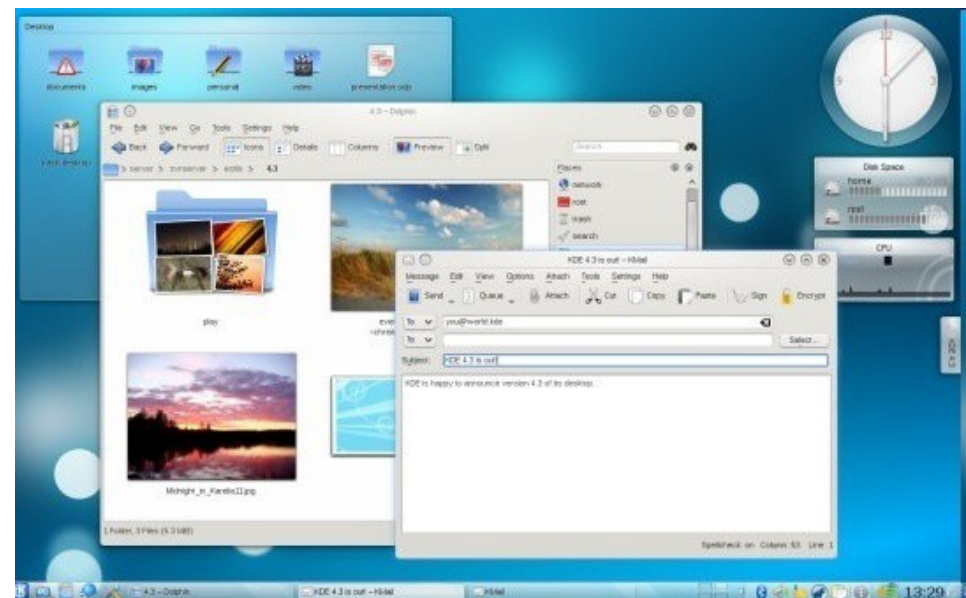
o KDE:

- **K** Desktop **E**nvironment.

- Es un proyecto que surgió para crear entornos de escritorio para diversos sistemas operativos, intentando que fueran parecidos a los de Windows o Mac OS X.

- Para usarlo con Ubuntu, se puede instalar Kubuntu o instalar el paquete **kubuntu-desktop** y así poder elegir entre uno u otro en el inicio de sesión.

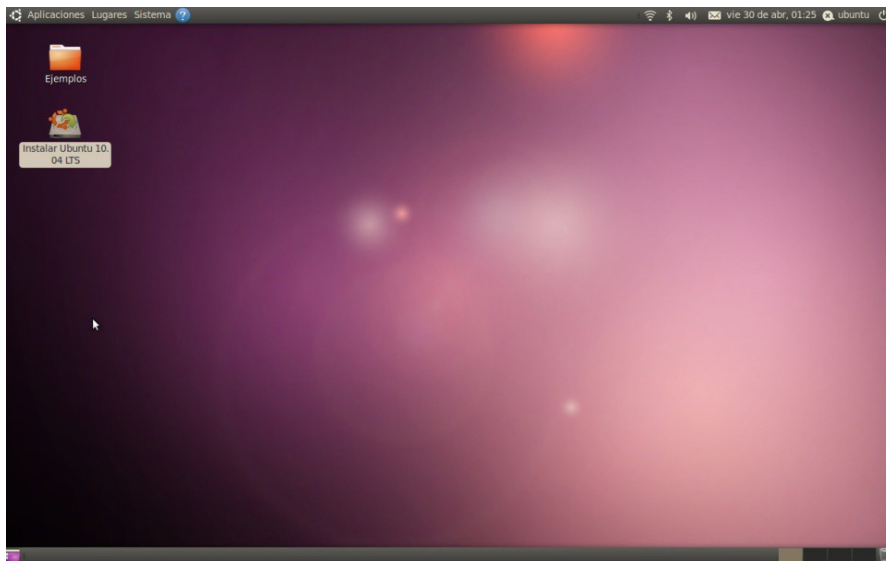
- Sus aplicaciones suelen empezar o contener la letra K.



4.1 Entornos de escritorio

o GNOME:

- **GNU Network Object Model Environment.**
- Forma parte del proyecto GNU y tiene licencia GPL.
- Surgió como alternativa al KDE, intentando ser más intuitivo y más atractivo que el KDE, simple, fácil de utilizar y disponible en muchos idiomas.



GNOME 2

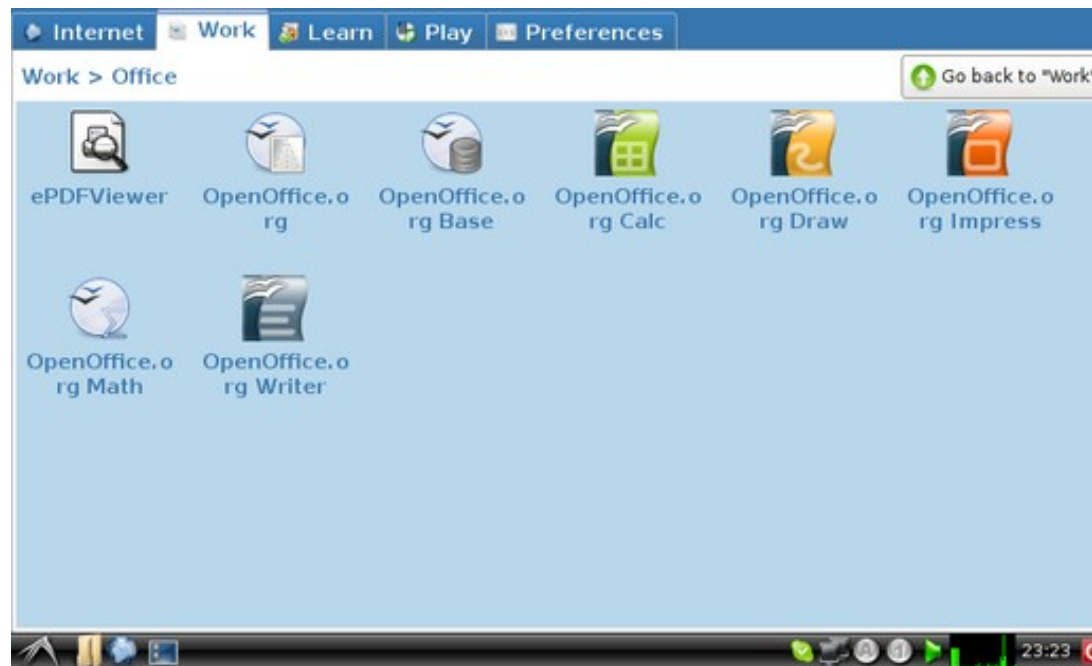


GNOME 3

4.1 Entornos de escritorio

o Lxde:

- Lightweight **X**11 **D**esktop **E**nvironment
- Escritorio ligero que consume pocos recursos y ahorra energía, pensado para equipos antiguos o netbooks.

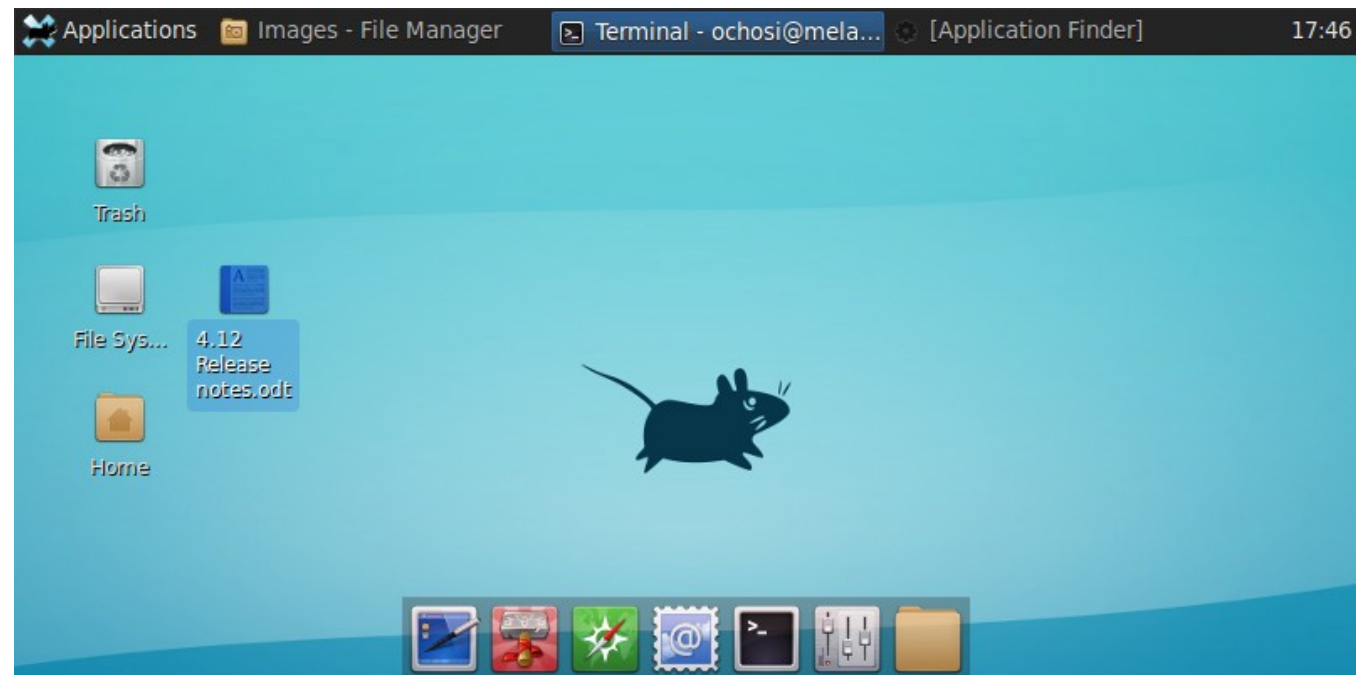


4.1 Entornos de escritorio



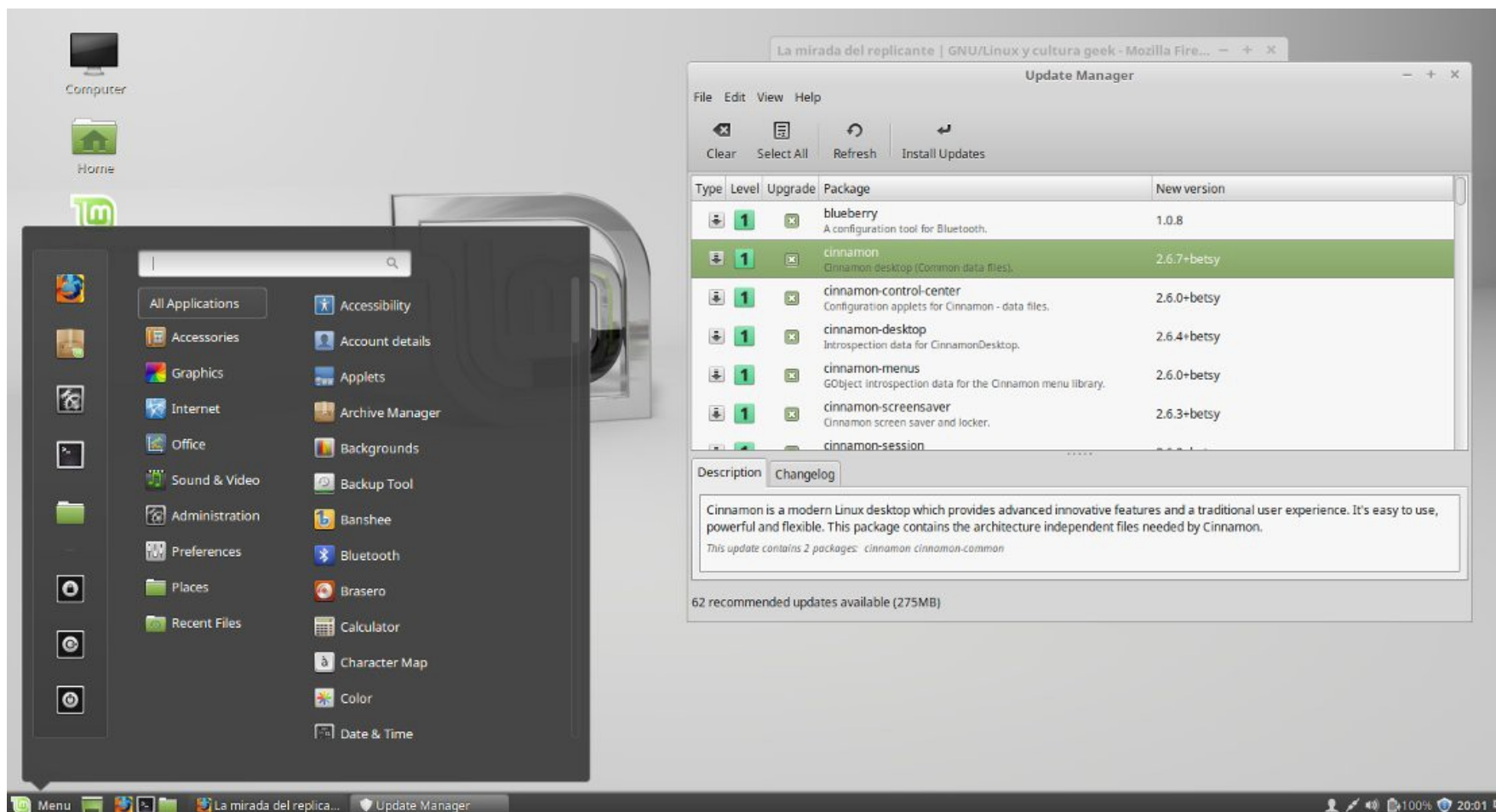
o Xfce:

- Es un entorno de escritorio ligero para sistemas tipo UNIX.
- Su objetivo es ser rápido y usar pocos recursos del sistema, sin dejar de ser visualmente atractivo y fácil de usar.



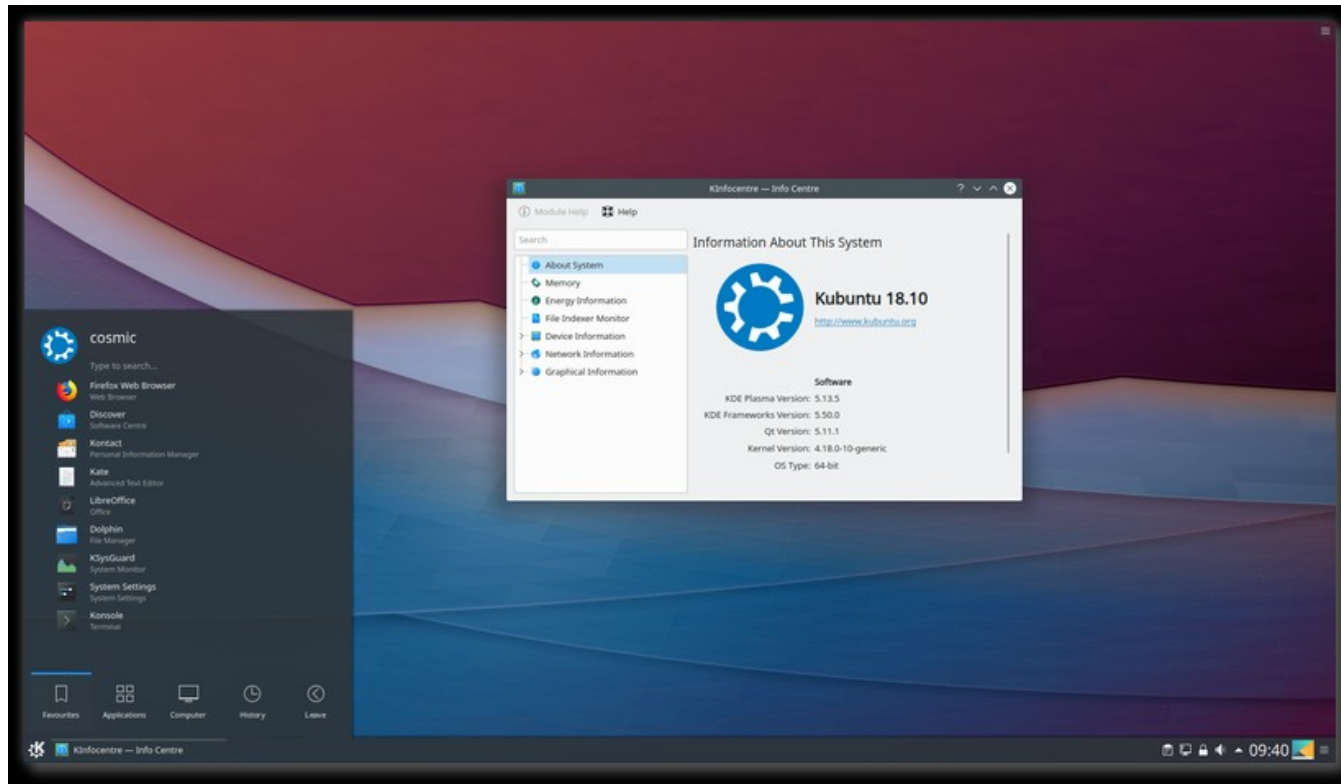
4.1 Entornos de escritorio

- Cinnamon:
 - Desarrollado por Linux Mint.

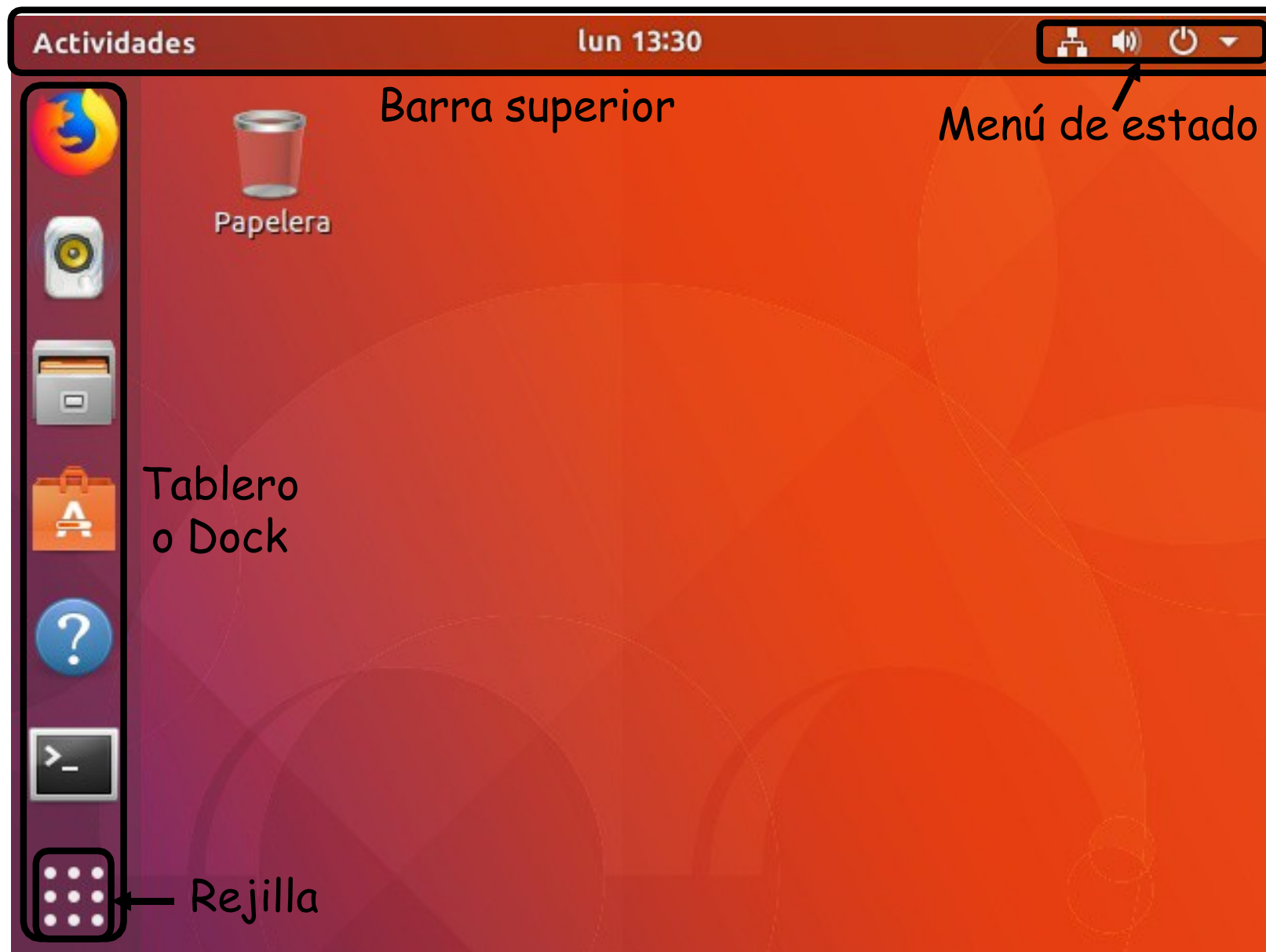


4.1 Entornos de escritorio

- Por ejemplo, Ubuntu con KDE:

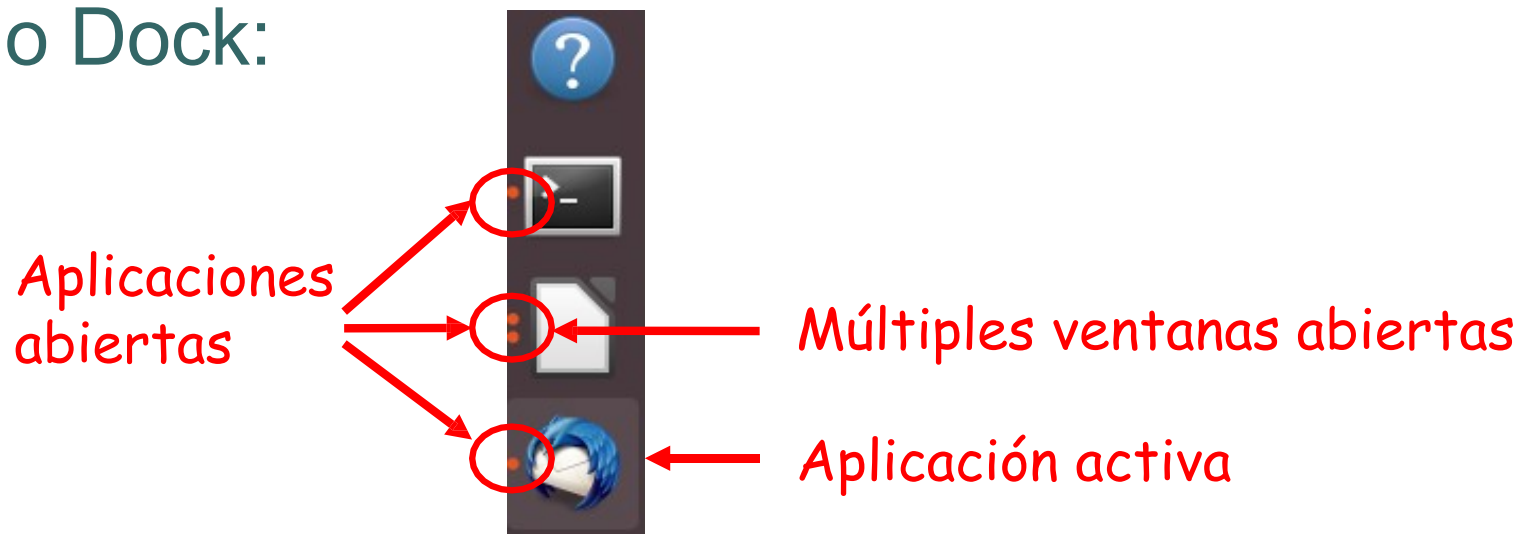


4.2 Escritorio de Ubuntu



4.2 Escritorio de Ubuntu

○ Tablero o Dock:



● Añadir aplicaciones al tablero:

- Buscar la aplicación en la rejilla, pulsar con el botón derecho sobre su icono y seleccionar “Añadir a los favoritos”.

● Quitar aplicaciones del tablero:

- Botón derecho sobre su icono en el tablero y seleccionar “Quitar de los favoritos”.

● Organizar aplicaciones en el tablero:

- Arrastrar el icono hasta la posición deseada.

4.2 Escritorio de Ubuntu

- En la parte inferior está el botón de rejilla, que muestra todas las aplicaciones instaladas.





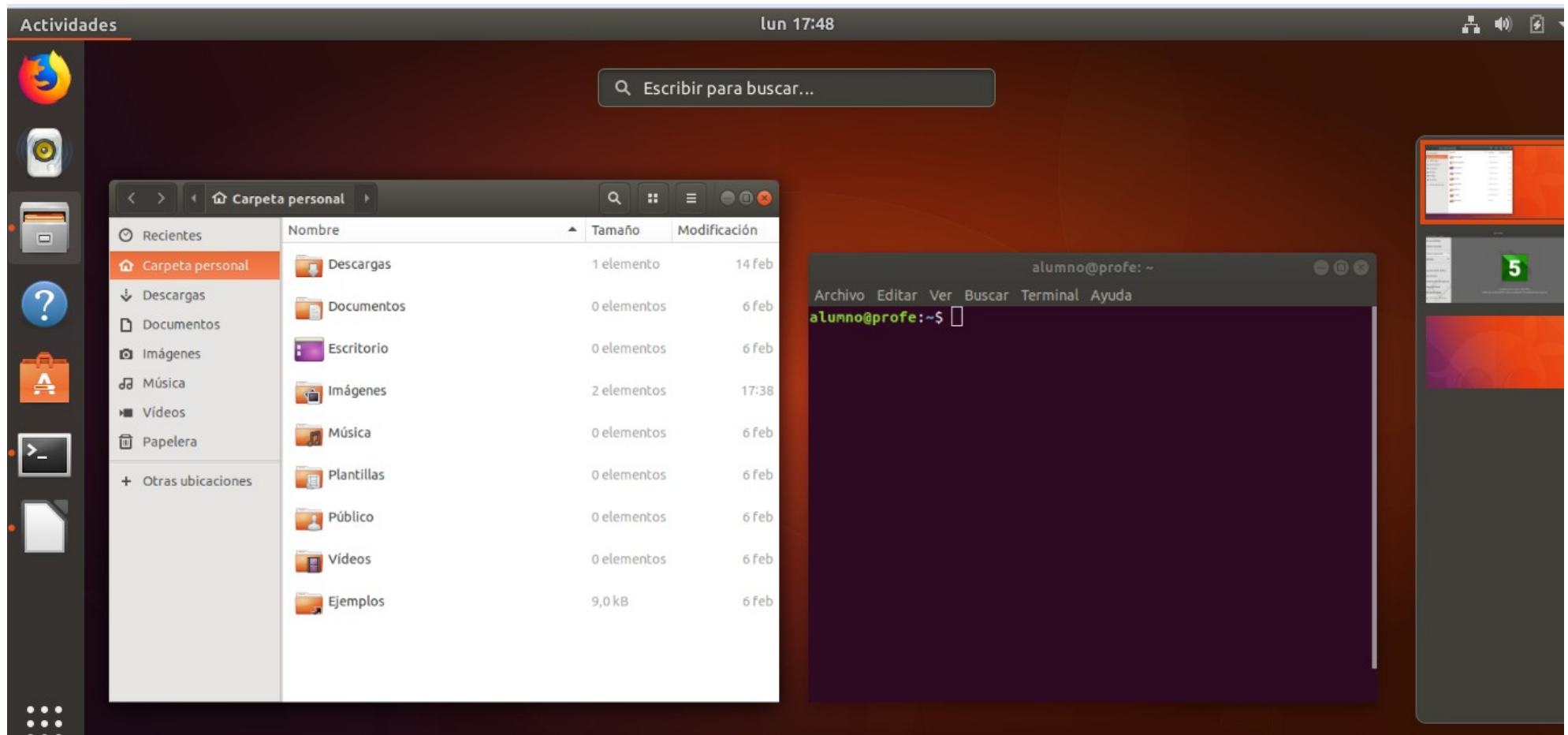
4.2 Escritorio de Ubuntu

o Actividades:

- Muestra todas las aplicaciones abiertas y las áreas de trabajo.
- Para escoger entre ventanas abiertas también se pueden pulsar las teclas → Alt + Tab
- Desde Actividades se pueden arrastrar aplicaciones a áreas de trabajo diferentes.

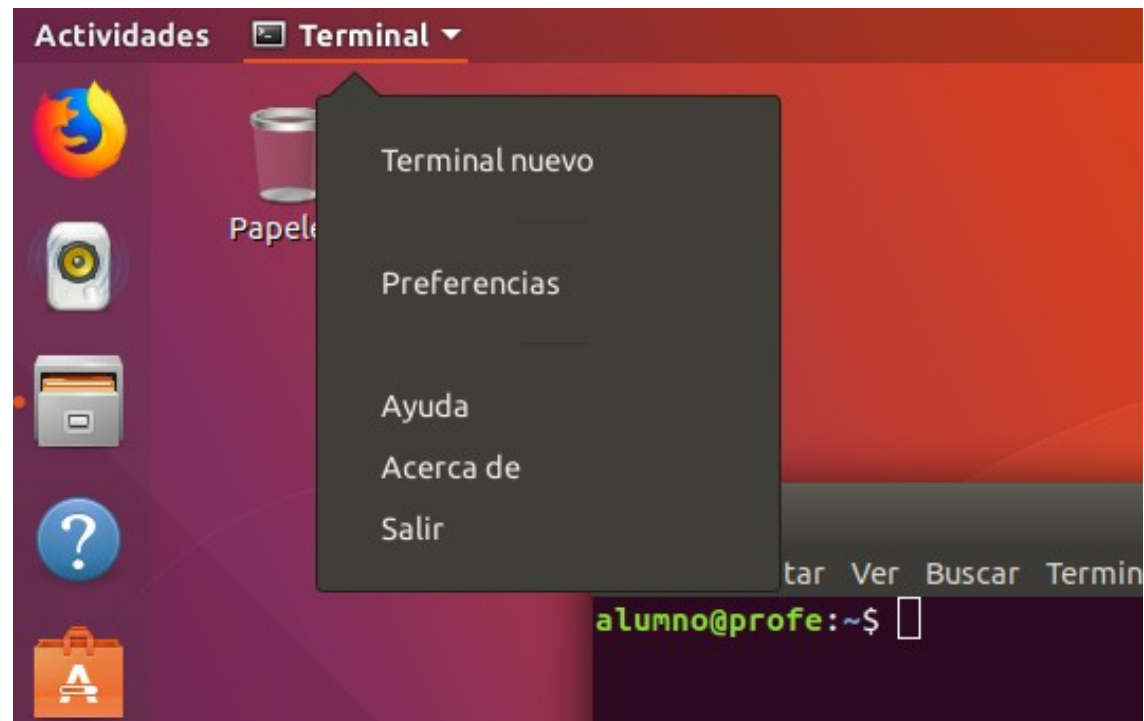


4.2 Escritorio de Ubuntu



4.2 Escritorio de Ubuntu

- El menú de una aplicación abierta aparece junto al botón de Actividades.





4.2 Escritorio de Ubuntu

o Menú de estado:

- Contiene una serie de indicadores del sistema que permiten monitorizar y ajustar el estado de algunas cosas.
- También cerrar la sesión, bloquear, apagar o reiniciar.

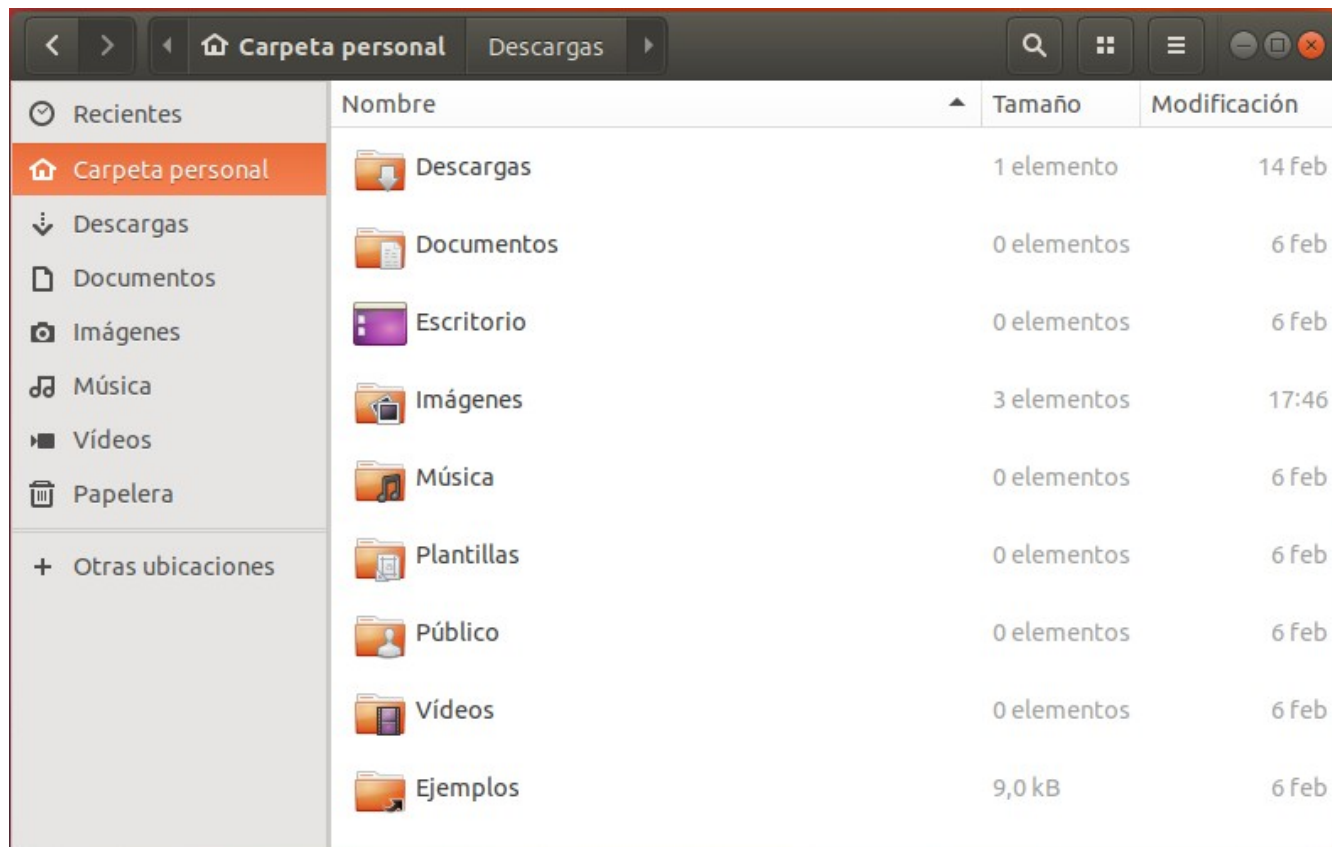




5. Administrador de archivos

5. Administrador de archivos

- El Administrador o Navegador de archivos en Ubuntu se denomina **Archivos**, pero la aplicación en sí, se llama **Nautilus** y es parte de Gnome.



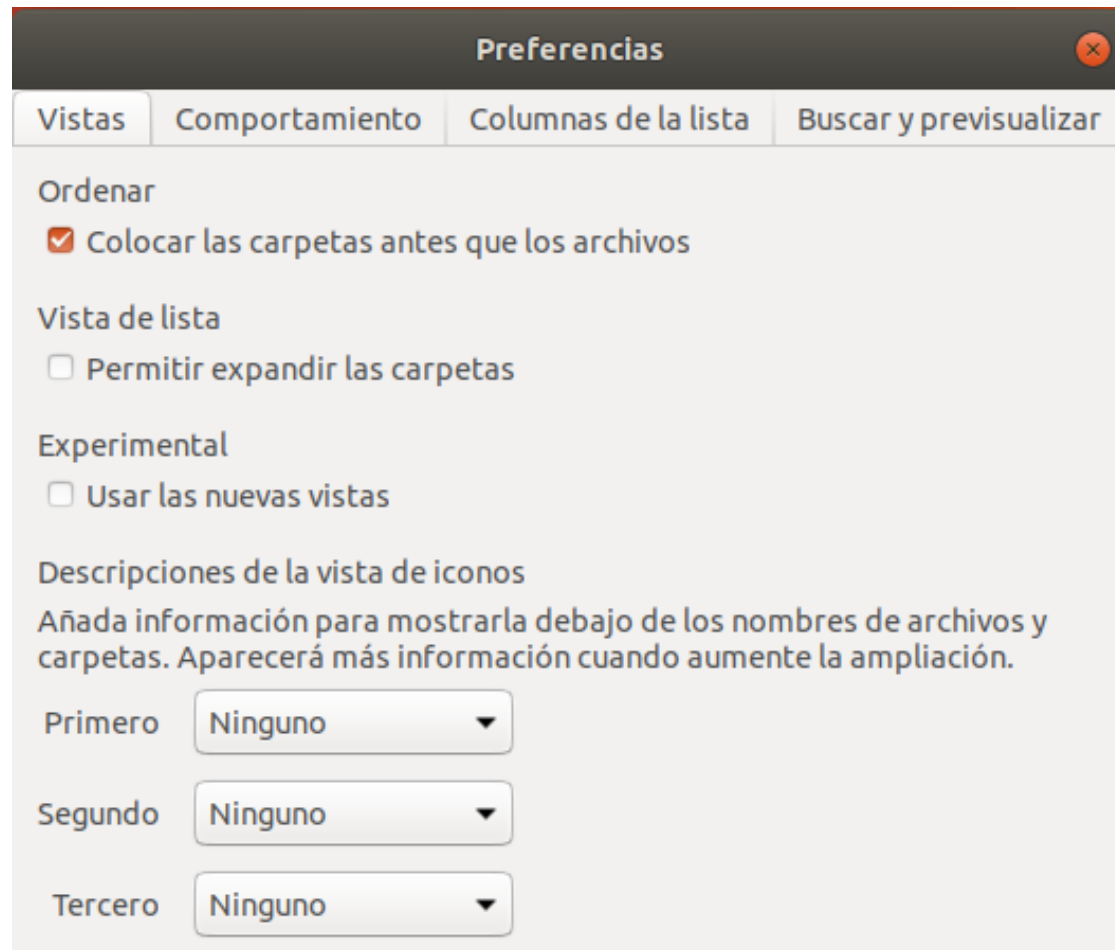
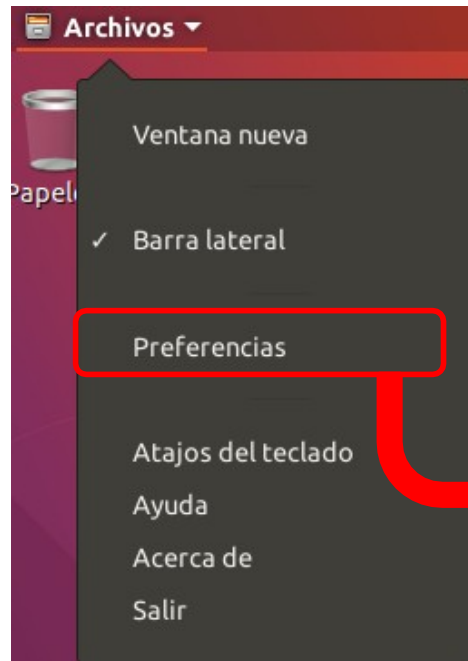


5. Administrador de archivos

- Con Nautilus se puede:
 - Crear
 - Eliminar
 - Copiar
 - Mover
 - Renombrar
 - Crear Accesos directos
 - Cambiar Propiedades
- tanto para archivos como para carpetas.

5. Administrador de archivos

- Para personalizar Nautilus:





6. Sistema de ficheros



6. Sistema de ficheros

1. Diferencias con Windows
2. Principales directorios y su función
3. Ficheros y directorios

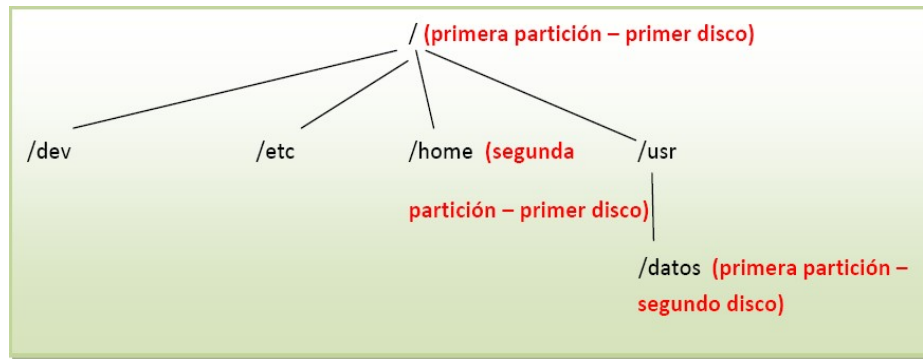


6.1 Diferencias con Windows

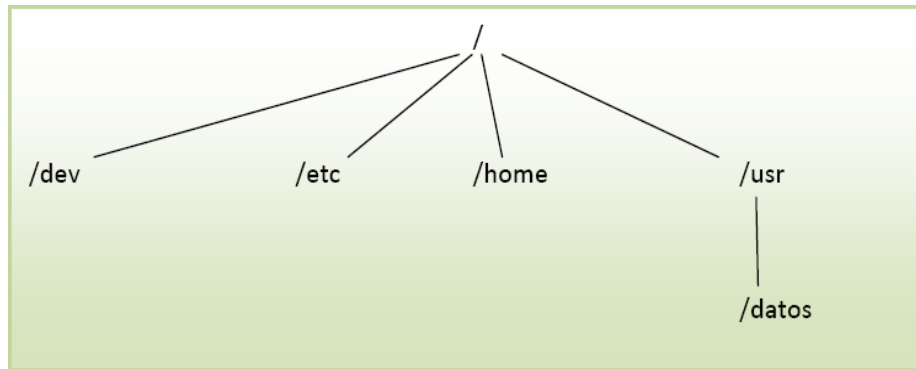
- En **Windows** a cada partición y a cada dispositivo de almacenamiento se le asigna una letra de unidad que lo identifica y además cada unidad tiene una raíz del sistema de ficheros (\).
 - Por ejemplo, si se tiene un equipo con dos discos duros y tres particiones (dos en el primero y una en el segundo disco) habrá tres letras de unidad, cada una de las cuales identificará una partición.
 - Por otro lado, la unidad de DVD, tarjetas SD o los lápices de memoria que se conecten tendrán también otras letras de unidad y su raíz del sistema de ficheros.

6.1 Diferencias con Windows

- En Linux solamente existe una raíz del sistema de ficheros (representada por $\rightarrow /$) y cada partición se integra dentro de éste.
 - En el ejemplo anterior, en Linux, podría quedar un dibujo así:



- Pero en el esquema de directorios no se aprecian los discos ni las particiones, parece que todo forme parte del mismo sistema de ficheros:



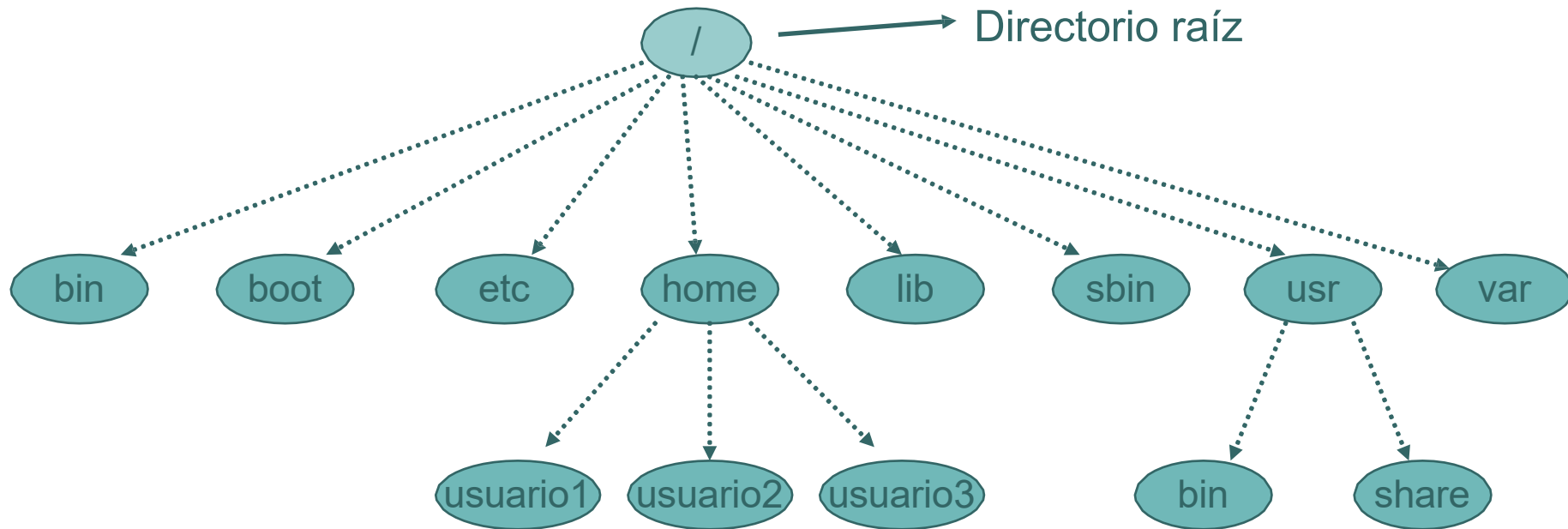
- En Linux no se asignan letras de unidad y hay un sistema de ficheros común donde se integran las diferentes particiones y dispositivos.



6.1 Diferencias con Windows

- Cada dispositivo hardware tiene su correspondiente fichero de dispositivo especial, el cual lo identifica dentro del sistema. Estos ficheros son utilizados por el sistema y se encuentran en `/dev`.
- La nomenclatura que siguen los discos en Linux es:
 - `sda` identifica el primer disco.
 - `sdb` identifica el segundo, etc.
- El siguiente dígito identifica la partición dentro del disco:
 - `sda1` es la primera partición del primer disco.
 - `sda2` es la segunda, etc.
- Ejemplos:
 - `sdb2` → segunda partición del segundo disco.
 - `sd1` → primera partición del tercer disco.

6.2 Principales directorios y su función



- Durante la instalación se crean unos directorios, siendo algunos de ellos:
 - **/bin** → programas, ficheros ejecutables por cualquier usuario.
 - **/sbin** → programas que sólo puede ejecutar el superusuario.
 - **/home** → directorios personales de los usuarios.
 - **/root** → directorio personal del superusuario.

6.2 Principales directorios y su función

- **/boot** → ficheros necesarios para arrancar el sistema, excepto los de configuración. En este directorio se almacenan los ficheros correspondientes al cargador de arranque (grub) y el núcleo del sistema operativo. Este último recibe el nombre de *vmlinuz*.
- **/etc** → ficheros de configuración del sistema.
- **/tmp** → ficheros temporales e información temporal de los programas.
- **/usr** y **/opt** → utilidades y programas instalados.
- **/dev** → ficheros asociados a dispositivos hardware. Cada dispositivo hardware tiene asociado un fichero de dispositivo especial. Algunos ejemplos son:

```
crw-rw----. 1 root lp    6, 1 nov 15 18:26 /dev/lp1
brw-rw----. 1 root disk 8, 0 nov 15 18:26 /dev/sda
brw-rw----. 1 root disk 8, 1 nov 15 18:26 /dev/sda1
crw--w----. 1 root tty   4, 1 nov 15 18:26 /dev/tty1
```

- **/var** → ficheros de eventos del sistema o logs.
- **/proc** → directorio actualizado automáticamente por el sistema. En él se guarda información sobre el núcleo, los procesos y el hardware del sistema.



6.3 Ficheros y directorios

- Los ficheros son los elementos básicos para la organización lógica de la información en el sistema de ficheros.
- Tipos de ficheros en Linux:
 - Ficheros ordinarios o regulares (-):
 - De texto, imágenes, sonido, ejecutables, etc.
 - Directorios (*d*):
 - Ficheros especiales que agrupan otros ficheros de una forma estructurada.
 - Ficheros especiales o de dispositivos (*b*, *c*):
 - Representan los dispositivos conectados a un ordenador (impresoras, discos, terminal, etc.). Su tratamiento es similar a un fichero ordinario.
 - Enlaces simbólicos (*l*):
 - Ficheros que contienen la ruta de acceso a otros ficheros o directorios.
 - Tuberías y sockets (*p*, *s*):
 - Utilizados para la comunicación entre procesos.

6.3 Ficheros y directorios

○ Nombre de un fichero:

- El nombre puede tener entre 1 y 255 caracteres.
- Se puede utilizar cualquier carácter excepto la barra inclinada / .
- No es recomendable emplear los caracteres con significado especial (= \ ^ ~ ' " ` * ; - ? [] () ! & ~ < >). Para usar ficheros con estos caracteres o espacios hay que introducir el nombre del fichero entre comillas.
- Las letras mayúsculas y minúsculas se consideran diferentes.

○ Ruta o *path*:

- **Ruta absoluta**: Secuencia de directorios que se ha de recorrer para acceder al fichero desde el directorio raíz.
- **Ruta relativa**: Secuencia de directorios que se ha de recorrer para acceder al fichero desde el directorio actual.

○ Directorios singulares:

- / → raíz.
- . → directorio actual.
- .. → directorio padre.
- ~ → *home* del usuario. (~ → AltGr + 4)



7. Shell de Linux



7. Shell de Linux

1. Shell: definición y características
2. Ayuda
3. Comodines
4. Redirecciones



7.1 Shell: definición y características

- El shell es el intérprete de comandos: actúa como interfaz entre el usuario y el sistema.
- La interacción con el shell se hace a través de terminales. Hay dos formas de acceder a un terminal:
 - En Ubuntu, introducir en el cuadro de búsqueda la palabra *terminal*.
 - En cualquier distribución, pulsando la siguiente combinación de teclas: **Ctrl+Alt+F1** a **Ctrl+Alt+F7**. En Linux existen 7 consolas desde las cuales entrar en el sistema, 6 de ellas en modo texto y una en modo gráfico. Se puede pasar de una a otra pulsando estas combinaciones.
- Al abrir un terminal nos aparece el indicador o prompt. En Ubuntu por defecto es:
`nombre_usuario@nombre_equipo:ruta_absoluta_directorio_actual$`



7.1 Shell: definición y características

- Hay diferentes intérpretes, algunos de ellos son:
 - Bourne Shell (sh)
 - Bourne Again Shell (bash) → el más usado
 - Korn Shell (ksh)
 - Tenex C Shell (tcsh)
- Una **orden** o **comando** es un fichero ejecutable del sistema. No se envía al sistema hasta que se pulsa la tecla Enter.

orden [-opciones] [argumentos]


- **Opciones:**
 - Van precedidas de un guión. Ej: ls -l
 - Pueden agruparse varias opciones. Ej: ls -la
 - El orden de las opciones no es significativo.
- **Argumentos o parámetros:**
 - Van separados por espacio o tabulador.



7.1 Shell: definición y características

○ Características:

- Se **distingue** entre mayúsculas y minúsculas.
- Permite el **autocompletado** de comandos y ficheros. Se pueden teclear las primeras letras de un comando o nombre de fichero, pulsar la tecla tabulador, y dejar que el sistema lo complete.
- Se pueden ejecutar **varios comandos** sin que estén conectados entre sí de ningún modo, tecleándolos en la misma línea y separándolos por punto y coma (;).
- Permite **encadenar** varias **órdenes**.
- Se puede **redireccionar** la **E/S**.
- Dispone de un **historial** de comandos para poder recuperar los ya utilizados mediante las teclas del cursor.
- Proporciona un lenguaje de programación: **shell scripts**.



7.2 Ayuda

- Hay varias maneras de ver la ayuda sobre una orden, siendo algunas de las más usadas:
 - **man orden** → Permite subir y bajar con las flechas y RePag y AvPag. Al salir (pulsando “q”), desaparece la información de la ventana de terminal. Los documentos de las páginas *man* están almacenados en un formato comprimido. El comando **man** descomprime y formatea estas páginas para su correcta visualización.
 - **info orden** → Muestra la información de forma parecida a como lo hace man.
 - **orden --help** → Muchos comandos disponen también de esta ayuda que se muestra de forma más resumida. La información permanece en la ventana de terminal.

7.3 Comodines

- Los comodines o metacaracteres son caracteres que se utilizan en lugar de otros caracteres que el sistema rellena. Los comodines más frecuentes son:
 - * → cualquier número de caracteres o ninguno.
 - ? → un solo carácter.
 - [] → un solo carácter de los indicados.
 - Ejemplos:
 - `ls file*` →
file
file1
file55.sh
file.old
 - `ls file?` →
file0
file1
 - `ls [0-9]*` → lista los ficheros cuyo nombre empiece por un número.



7.4 Redirecciones

- El kernel abre para cada orden tres ficheros:
 - `/dev/stdin` → entrada estándar (0): teclado.
 - `/dev/stdout` → salida estándar (1) : pantalla.
 - `/dev/stderr` → salida de error (2): pantalla.
- Las operaciones de redirección permiten cambiar la E/S estándar.
 - **Redirección de entrada:**
 - `orden < fichero` → toma el fichero como entrada de la orden. No se usa mucho.
 - **Redirección de salida:**
 - `orden > fichero` → guarda en el fichero el resultado de la orden, borrando lo que hubiera en el fichero.
 - `orden >> fichero` → añade al final del fichero el resultado de la orden.
 - **Redirección de salida de error:**
 - `orden 2> fichero` → guarda en el fichero los errores que se puedan producir al ejecutar la orden, borrando lo que hubiera en el fichero.
 - `orden 2>> fichero` → añade al final del fichero los errores que se puedan producir al ejecutar la orden.
 - También se pueden redirigir la salida estándar y la salida de error al mismo sitio, representándolas con el símbolo: `&`.



8. Comandos de gestión de ficheros y directorios



8. Comandos de gestión de ficheros y directorios

1. Gestión de directorios
2. Gestión de ficheros
3. Visualizar ficheros
4. Edición de ficheros
5. Búsqueda de ficheros
6. Compactado de ficheros y directorios



8.1 Gestión de directorios

o **pwd**

- Print working directory.
- Muestra por pantalla la ruta completa del directorio actual.

o **cd [directorio]**

- Cambia el directorio de trabajo. Va al directorio que se indique, bien con una ruta absoluta, bien con una ruta relativa.
- Sin opciones ni argumentos → Va al directorio HOME del usuario.

o **ls [opciones] [directorio]**

- Muestra el contenido de un directorio en orden alfabético.
- Opciones:
 - -l → listado en formato largo, mostrando más información.
 - -a → lista también los ficheros y directorios ocultos.
 - -d → lista los nombres de directorios como ficheros, en lugar de mostrar su contenido.
 - -R → listado recursivo (lista todos los subdirectorios).

o **mkdir [opciones] directorio**

- Crea un directorio.
- Opciones:
 - -p → permite crear los directorios padres que no existan.

o **rmdir directorio**

- Borra un directorio, siempre y cuando esté vacío.



8.2 Gestión de ficheros

- **cp [opciones] fichero1 fichero2**
 - Copia el contenido de fichero1 en fichero2.
 - Opciones:
 - -R → copia recursivamente. Para copiar directorios.
 - -i → pide confirmación antes de sobrescribir ficheros que existan en el destino.
 - -v → muestra por pantalla los ficheros copiados.
- **rm [opciones] fichero**
 - Borra ficheros y directorios.
 - Opciones:
 - -R → indica comportamiento recursivo. Para borrar directorios.
 - -i → pide confirmación antes de eliminar.
 - -v → muestra por pantalla los ficheros eliminados.
- **mv [opciones] fichero1 fichero2**
 - Mueve ficheros y directorios dentro del sistema de ficheros.
 - Se usa también para renombrar un fichero o directorio.
 - Opciones:
 - -i → pide confirmación antes de sobrescribir ficheros que existan en el destino.
 - -v → muestra por pantalla los ficheros movidos.
- **touch fichero**
 - Crea un fichero vacío.
 - Si el fichero existe → modifica su fecha de actualización.



8.3 Visualizar ficheros

○ `cat [opciones] [fichero(s)]`

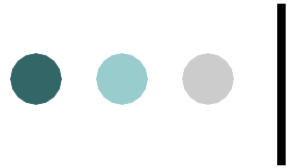
- Muestra en la salida estándar lo que se introduce por la entrada estándar.
- Si se pasa un fichero como parámetro, muestra su contenido. Si se pasan varios ficheros, los concatena.

○ `more [fichero]`

- Muestra el contenido de un fichero página a página. Indica el porcentaje de fichero visualizado.
- Para avanzar una línea → Enter.
- Para avanzar una página → Barra espaciadora.
- Para terminar antes de llegar al final → q.
- Permanece la información en la ventana de terminal.

○ `less [fichero]`

- Muestra el contenido de un fichero página a página.
- Para avanzar una línea → Enter.
- Para avanzar una página → Barra espaciadora.
- Permite subir y bajar con las flechas y RePag y AvPag o buscar patrones en el fichero, entre otras cosas.
- Al salir (pulsando “q”), desaparece la información de la ventana de terminal.



8.4 Edición de ficheros

- En Linux existen muchos editores de texto disponibles (vi, vim, nano, Emacs, joe,...).
- El visual editor (**vi**) es el único que se encuentra en cualquier sistema Unix. Fue el primer editor de pantalla completa que existió y, aunque no es fácil de usar, es una herramienta muy potente.
- El **nano** es un editor bastante sencillo de usar.



8.5 Búsqueda de ficheros

o `find [ruta] [opciones]`

- Busca los ficheros que cumplen las opciones especificadas. Comienza a partir de la ruta indicada y continúa por todos sus subdirectorios.
- Si no se indica una ruta, busca a partir del directorio actual.
- Opciones:
 - `-name "nombre"` → Se pueden usar metacaracteres (*,?,...).
 - `-type tipo` → f (fichero ordinario), d (directorio), l (enlace), etc.
 - `-user propietario` → Propietario de los ficheros.
 - `-group grupo` → Grupo de los ficheros.
 - `-links n` → Número de enlaces de los ficheros.
 - `-perm permisos` → Permisos exactos en formato octal o simbólico. Con un menos (-) delante de los permisos, indica que al menos se han de tener esos permisos.
 - `-maxdepth n` → Indica cuántos niveles de subdirectorios se ha de descender. n=1 → directorio actual, n=2 → un nivel de subdirectorios. Esta opción se ha de poner la primera.
- Se pueden hacer acciones con los ficheros encontrados. Para hacer referencia a éstos, se usan las {}:
 - `-exec comando \;` → ejecuta el comando sin pedir confirmación.
 - `-ok comando \;` → ejecuta el comando solicitando confirmación.



8.5 Búsqueda de ficheros

● Ejemplos:

- `find / -name "test*" →` partiendo desde el directorio raíz, muestra los ficheros (y directorios) cuyo nombre comience por *test*.
- `find / -maxdepth 2 -name "test*" →` partiendo desde el directorio raíz, muestra los ficheros (y directorios) cuyo nombre comience por *test*, mirando sólo en el directorio indicado y un nivel de subdirectorios.
- `find ~ -user alumno -type d →` partiendo desde el directorio home del usuario, muestra los directorios cuyo propietario sea *alumno*.
- `find -type f -name "*ejercicio*" →` partiendo del directorio actual, busca los ficheros ordinarios cuyo nombre contenga la palabra *ejercicio*.
- `find -type f -exec cp {} /tmp \; →` partiendo del directorio actual, busca los ficheros ordinarios y los copia en el directorio /tmp.
- `find -perm 633 -ok -exec rm {} \; →` partiendo del directorio actual, busca los ficheros con permisos iguales a 633 y los elimina, solicitando confirmación antes de eliminar.



8.6 Compactado de ficheros y directorios

o `tar [c|t|x|z|j]f fichero.tar fich1 fich2 ...`

- Guarda varios ficheros juntos en un solo fichero (normalmente .tar).
- Opciones:
 - -c → crea.
 - -x → extrae.
 - -z → comprime con gzip.
 - -j → comprime con bzip2.
 - -f → referencia a un fichero (esta opción siempre la última).
 - -t → muestra el contenido.
- Ejemplos:
 - Empaquetar:

```
[alumno@server tmp]$ ls
fich1 fich2 fich3
[alumno@server tmp]$ tar cf todos.tar *
[alumno@server tmp]$ ls
fich1 fich2 fich3 todos.tar
[alumno@server tmp]$ tar cf solo1y2.tar fich1 fich2
[alumno@server tmp]$ ls
fich1 fich2 fich3 solo1y2.tar todos.tar
```



8.6 Compactado de ficheros y directorios

- Ver contenido (sin extraer):

```
[alumno@server tmp]$ tar tf todos.tar
fich1
fich2
fich3
[alumno@server tmp]$ tar tf sololy2.tar
fich1
fich2
```

- Desempaquetar:

```
[alumno@server tmp]$ rm fich?
[alumno@server tmp]$ ls
sololy2.tar todos.tar
[alumno@server tmp]$ tar xf todos.tar
[alumno@server tmp]$ ls
fich1 fich2 fich3 sololy2.tar todos.tar
```


8.6 Compactado de ficheros y directorios

- Comprimir con **gzip** (**.tar.gz**):

```
[alumno@server tmp]$ ls
fich1 fich2 fich3
[alumno@server tmp]$ tar czf todos.tar.gz *
[alumno@server tmp]$ ls
fich1 fich2 fich3 todos.tar.gz
```

- Extraer:

```
[alumno@server tmp]$ ls
todos.tar.gz
[alumno@server tmp]$ tar xf todos.tar.gz
[alumno@server tmp]$ ls
fich1 fich2 fich3 todos.tar.gz
```

- Comprimir con **bzip2** (**.tar.bz2**):

```
[alumno@server tmp]$ tar cjf sololy2.tar.bz2 fich1 fich2
[alumno@server tmp]$ ls
fich1 fich2 fich3 sololy2.tar.bz2
```

- Extraer:

```
[alumno@server tmp]$ ls
sololy2.tar.bz2
[alumno@server tmp]$ tar xf sololy2.tar.bz2
[alumno@server tmp]$ ls
fich1 fich2 sololy2.tar.bz2
```



8.6 Compactado de ficheros y directorios

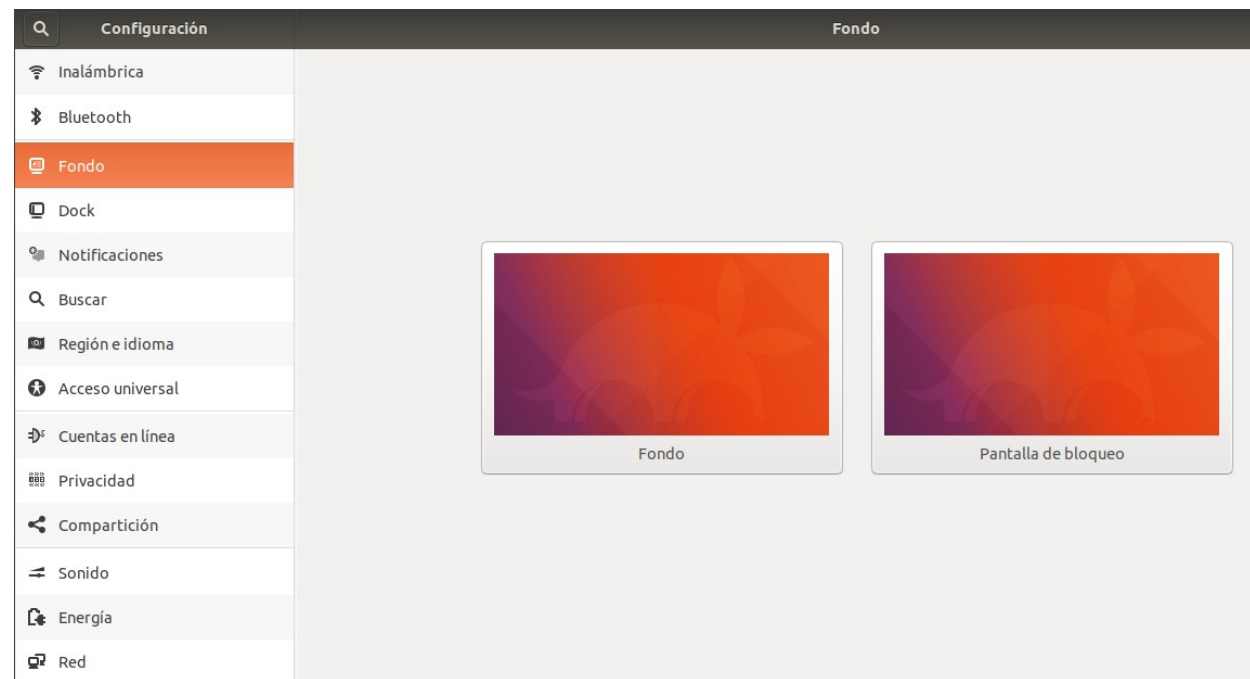
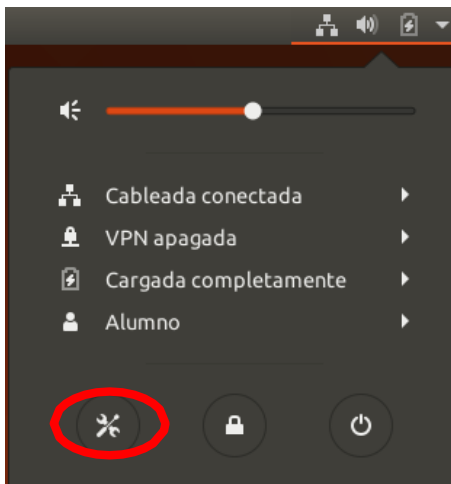
- Hay más posibilidades de compresión. Por ejemplo:
 - Con el comando **tar**, hay opciones para comprimir y descomprimir con xz, lzip, lzma, lzop,...
 - Comandos **gzip** y **gunzip**.
 - Comandos **zip** y **unzip**.



9. Configuración del sistema

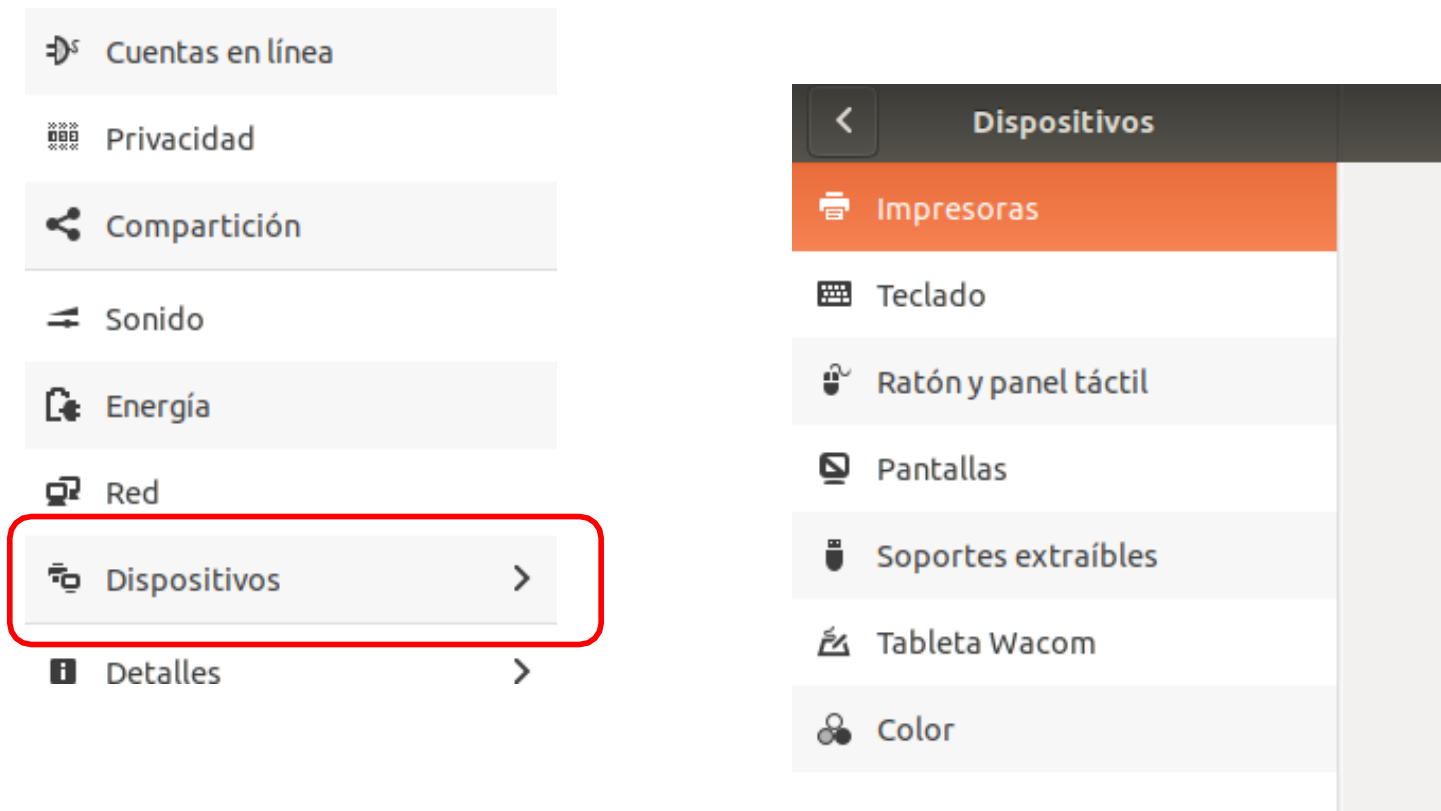
9. Configuración del sistema

- Toda la configuración en Linux se hace mediante ficheros de texto en el directorio `/etc`. Ubuntu incorpora una herramienta gráfica llamada **Configuración**.



9. Configuración del sistema

- Desde **Configuración**, se tiene acceso a la configuración de algunos dispositivos:





9. Configuración del sistema

- En el directorio `/proc`, hay una serie de ficheros que muestran información acerca del hardware.
- Algunos de los comandos que sirven para ver información del hardware son:
 - `sudo lshw` → Muestra información detallada del hardware del equipo.
 - `sudo lshw -html > ~/fichero.html` → Muestra la información en formato HTML y la guarda en un fichero.
 - `lspci` → Muestra información de los buses PCI y de los dispositivos conectados a ellos.
 - `lsusb` → Muestra información de los buses USB y de los dispositivos conectados a ellos.



10. Instalación de software



10. Instalación de software

1. Herramienta gráfica Software
2. Comandos de instalación de software
3. Software y actualizaciones



10.1 Herramienta gráfica Software



Todo			Instalado	Actualizaciones
	Actualización de software ★★★★★	This is the GNOME apt update manager. It checks for updates and lets the user choose which to install.	Desinstalar 1,6 MB	
	Aplicaciones al inicio ★★★★★	El gestor de sesión GNOME está a cargo de iniciar los componentes del núcleo del escritorio GNOME y las aplicacion...	Desinstalar 155,6 kB	
	Calendario ★★★★★	Calendario de GNOME es una aplicación de calendario bonita y sencilla diseñada para ajustarse perfectamente al escritorio GN...	Desinstalar 1,0 MB	
	Cheese ★★★★★	Cheese usa su cámara web para hacer fotos y vídeos, aplica efectos especiales divertidos y le permite compartir su diversió...	Desinstalar 446,5 kB	
	Compartición del escritorio ★★★★★ Varios	VNC es un protocolo que permite mostrar el escritorio de un usuario de forma remota. Este paquete proporciona un servido...	Desinstalar 598,0 kB	
	Compartir archivos personales ★★★★★	gnome-user-share es un pequeño paquete que permite fácilmente compartir archivos de un usuario mediante WebDAV...	Desinstalar 983,0 kB	



10.2 Comandos de instalación de software

- En Linux, las diferentes distribuciones disponen de sitios web o ftp de los cuales se puede descargar software o actualizaciones de forma gratuita → [repositorios](#).
- Para facilitar la tarea se han creado programas que instalan los paquetes directamente desde Internet, con la ventaja adicional de que son capaces de resolver las dependencias que haya.
- Cada distribución dispone de un directorio que contiene ficheros en los que se configuran los repositorios a los cuales se pueden conectar. En estos ficheros se almacena la dirección URL que se corresponde con un repositorio en particular. Además también se guarda una clave con la que se accede a este repositorio.
- Existen repositorios oficiales y no oficiales. Los oficiales son los que pone a disposición del público la propia distribución.
- Cada distribución tiene sus propias utilidades para instalar paquetes desde los repositorios de Internet.



10.2 Comandos de instalación de software

- CentOS y Red Hat utilizan la utilidad **yum**. Las principales opciones son las siguientes:
 - `yum update [paquete]` → Si no se especifica ningún paquete, actualiza todos los paquetes instalados.
 - `yum install paquete` → Descarga e instala un paquete.
 - `yum remove paquete` → Elimina un paquete del sistema.
 - `yum search cadena` → Se usa para buscar paquetes cuando no se está seguro de cuál es su nombre.
 - `yum info paquete` → Consulta información acerca de un paquete.
 - `yum list available [paquete]` → Lista los paquetes disponibles en los repositorios o si está disponible el paquete indicado.
 - `yum list installed [paquete]` → Lista todos los paquetes instalados en el sistema o si está instalado el paquete indicado.
 - `yum list updates` → Lista todos los paquetes pendientes de actualizar.
 - `yum repolist` → Muestra los repositorios configurados.
- Los ficheros de configuración de los repositorios se ubican en el directorio `/etc/yum.repos.d/`.
- El fichero de configuración de **yum** es `/etc/yum.conf`.

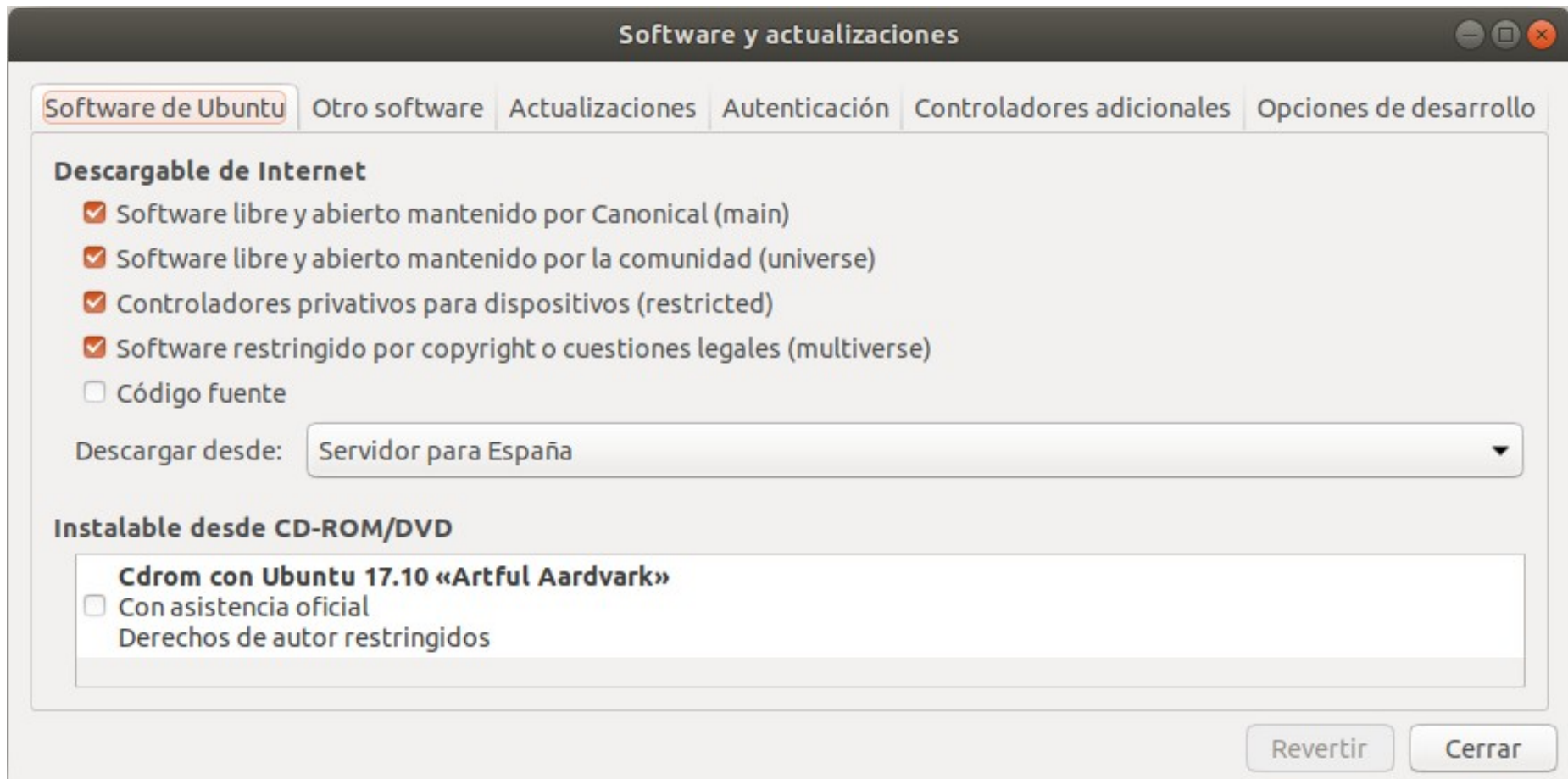


10.2 Comandos de instalación de software

- Ubuntu y Debian utilizan las utilidades `apt-get` o `aptitude`. Aptitude es una versión mejorada de apt.
- Las principales opciones de `apt-get` son:
 - `apt-get update` → Actualiza la información sobre los repositorios.
 - `apt-get install paquete` → Instala un paquete en el sistema.
 - `apt-get remove paquete` → Elimina un paquete del sistema, pero mantiene los ficheros de configuración.
 - `apt-get purge paquete` → Elimina un paquete del sistema, incluidos sus ficheros de configuración.
 - `apt-get upgrade [paquete]` → Actualiza un paquete ya instalado. Sin opciones, actualiza el SO y todas las aplicaciones, siempre que no necesiten, como dependencia, la instalación o desinstalación de otros paquetes.
 - `apt-get dist-upgrade` → Hace lo mismo que `upgrade`, pero resolviendo las dependencias que sean necesarias.
 - `apt-get clean` → Elimina los ficheros de la caché de apt.
- En Ubuntu y Debian el fichero de configuración de los repositorios está en `/etc/apt/sources.list`. El directorio con los repositorios es `/etc/apt/sources.list.d/`.

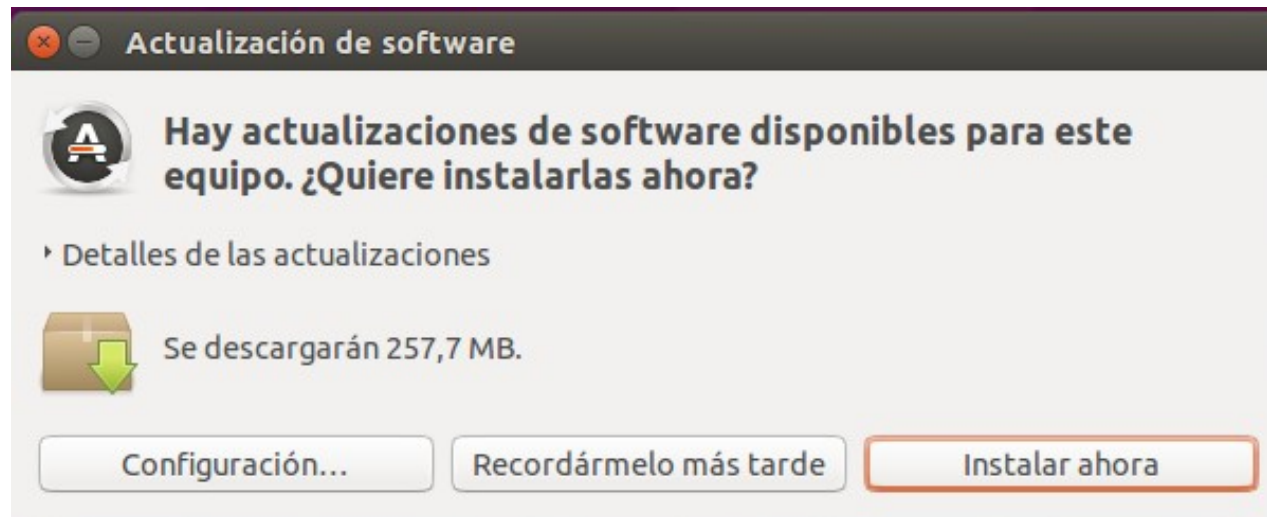
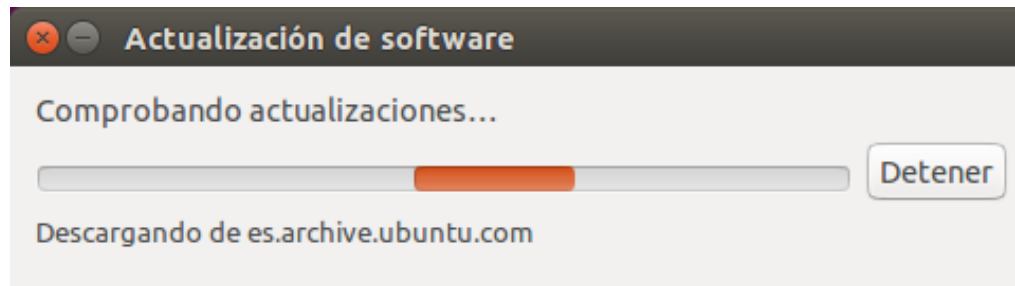
10.3 Software y actualizaciones

- La herramienta **Software y actualizaciones** permite configurar las actualizaciones.



10.3 Software y actualizaciones

- Para comprobar las actualizaciones pendientes, se puede abrir la aplicación Actualización de software.





11. Arranque y parada del sistema



11. Arranque y parada del sistema

1. Secuencia de arranque
2. Cargador de arranque. GRUB
3. Desconexión del sistema



11.1 Secuencia de arranque

- El proceso de arranque de un sistema Linux es:
 - **BIOS**. Al arrancar el equipo la **BIOS** toma el control del ordenador y realiza tareas de inicialización y comprobación de dispositivos hardware. Estas tareas reciben el nombre de **POST**. Una vez reconocido el disco duro, el control del sistema pasa de la BIOS al cargador de arranque.
 - **MBR**. El primer sector de datos de 512 bytes se carga en la RAM. El **cargador de arranque** consulta la tabla de particiones y busca cuál es la partición activa. A continuación, el cargador de arranque lanza la segunda parte del mismo, que se ubica en **/boot/grub2** y que muestra un menú con los sistemas instalados.
 - **Núcleo y initramfs**. El cargador de arranque carga en la memoria el **núcleo** y un sistema inicial de ficheros basado en RAM (**initramfs**). El initramfs contiene un pequeño ejecutable que gestiona el montaje del sistema real de ficheros.
 - **systemd**. El núcleo lanza el primer proceso llamado *systemd*, cuyo PID es el 1. systemd se encarga de iniciar el resto de servicios en función del target configurado en **/lib/systemd/system/default.target**.



11.2 Cargador de arranque. GRUB

- El cargador de arranque es un programa que se encarga de iniciar los sistemas operativos instalados en una máquina. En Linux se utiliza GRUB (GRand Unified Bootloader). Hasta el momento, hay dos versiones, GRUB Legacy y GRUB-2.
- GRUB es específico de sistemas Linux pero puede arrancar sistemas Windows. Consta de dos partes, la primera se denomina *stage1* y está en el primer sector del disco duro. La segunda se ubica en la partición de arranque de un sistema Linux (/boot) y se llama *stage2*.
- Dentro del directorio /boot se encuentra el núcleo de Linux y un directorio /boot/grub donde se almacena parte de la configuración de GRUB:
 - `/boot/grub/grub.cfg` → Contiene parámetros de configuración y los sistemas operativos que se pueden arrancar.
 - Este fichero NO se debe editar directamente. Se genera mediante el comando `grub-mkconfig` usando las plantillas que hay en el directorio `/etc/grub.d` y la configuración del fichero `/etc/default/grub`.

11.3 Desconexión del sistema

o `shutdown [opciones] [tiempo] [mensaje de aviso]`

- Detiene, reinicia o apaga el sistema de forma segura y ordenada en el momento indicado.
- Notifica a los usuarios de este hecho y, además, se bloquea el sistema para que nadie más pueda acceder.
- Si no se indica el argumento **tiempo**, se realiza la acción pasado 1 minuto. El tiempo se puede especificar de dos formas :
 - Indicando la hora en este formato → *hh:mm*.
 - Indicando los minutos que se ha de esperar → *+m*. Para cerrar inmediatamente → *now*, que equivale a *+0*.
- Opciones:
 - *-H* → Detiene el sistema.
 - *-P* → Detiene el sistema y apaga la máquina.
 - *-r* → Detiene el sistema y reinicia la máquina.
 - *-c* → Cancela un shutdown en curso.
 - Se puede indicar al final un mensaje que se enviará a todos los usuarios.
- Si no se especifica ninguna opción, apaga la máquina.
- Ejemplos:
 - `shutdown -r +10 Debido a tareas de mantenimiento, el sistema se reiniciará dentro de diez minutos`
 - `shutdown -h now`
 - `shutdown -P 22:30`

SI



11.3 Desconexión del sistema

○ **halt**

- Detiene el sistema.
- Equivale a: **shutdown -h now**

○ **reboot**

- Reinicia la máquina.
- Equivale a: **shutdown -r now**

○ **poweroff**

- Apaga la máquina.
- Equivale a: **shutdown -P now**
- **halt** detiene el sistema operativo pero no apaga la máquina, en cambio **poweroff** sí que apaga el equipo. En ordenadores antiguos, con **halt** había que apagar el equipo después desde un botón.

- Todos estos comandos se tienen que ejecutar como superusuario.



12. Particiones



12. Particiones

1. Creación de particiones
2. Formatear particiones
3. Montaje de dispositivos



12.1 Creación de particiones

- Se puede tener la necesidad de crear una partición, por ejemplo, al añadir un disco duro o si se ha dejado espacio sin particionar durante la instalación.
- Hay muchas herramientas para crear particiones y sistemas de ficheros en sistemas Linux. Desde la terminal, las principales herramientas son **fdisk**, **cfdisk** y **parted**.
- Para que funcionen bien, hay que ejecutarlas con permisos de root.
- El equivalente a **parted** en modo gráfico es la herramienta **GParted**.
- Con estas utilidades se puede:
 - **Ver** la tabla de particiones.
 - **Cambiar** el tamaño de una partición que ya existe.
 - **Crear** nuevas particiones.

12.1 Creación de particiones

○ Utilidad **parted**:

- Para usar la utilidad, hay que acceder como *root* y ejecutar el comando: **parted** [**dispositivo**]. Por ejemplo, `parted /dev/sda`. Aparece el prompt: `(parted)` y, a continuación, se pueden empezar a introducir comandos.

```
GNU Parted 2.1
Usando /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) █
```

- Si no se indica un dispositivo, usa el primer disco que encuentre.
- Comandos más usuales:
 - **help** → Muestra una lista de los comandos disponibles.
 - **print** → Muestra la tabla de particiones.
 - **quit** → Sale de parted.
 - **mklabel** *etiqueta* → Crea una etiqueta de disco para la tabla de particiones.
 - **mkpart** *tipo-partición tipo-sa start-mb end-mb* → Crea una partición
 - **rm** *número-minor* → Elimina una partición.
 - **resizepart** *número-minor tamaño* → Redimensiona una partición.

12.1 Creación de particiones

- Visualizar la tabla de particiones actual:
(parted) print

```
Model: ATA ST3160812AS (scsi)
Disk /dev/sda: 160GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
```

Number	Start	End	Size	Type	File system	Flags
1	32.3kB	107MB	107MB	primary	ext3	boot
2	107MB	105GB	105GB	primary	ext3	
3	105GB	107GB	2147MB	primary	linux-swap	
4	107GB	160GB	52.9GB	extended	root	
5	107GB	133GB	26.2GB	logical	ext3	
6	133GB	133GB	107MB	logical	ext3	
7	133GB	160GB	26.6GB	logical		lvm

- En las primeras líneas se muestran datos del disco.
- En la tabla de particiones, **el número Minor** es el número de la partición. Por ejemplo, la partición con número menor 1 corresponde a /dev/sda1.
- Se muestran los valores de **Inicio** y **Final**.
- El **Tipo** es primario, extendido o lógico.
- El **Sistema de ficheros** puede ser uno de los siguientes: ext2, ext3, ext4, fat16, fat32, hfs, jfs, linux-swap, ntfs, reiserfs, hp-ufs, sun-ufs, xfs.
- La columna **Etiquetas** enumera todas la etiquetas asignadas a la partición. Las etiquetas disponibles son boot, root, swap, hidden, raid, lvm y lba.



12.1 Creación de particiones

- Configurar una etiqueta de disco:

(parted) mklabel etiqueta

- Sólo si es un disco nuevo que no tiene ninguna partición, hay que configurar una etiqueta de disco que indique el tipo de tabla de particiones que va a tener.
- Los tipos de etiqueta que se pueden configurar son: bsd, dvh, gpt, loop, mac, msdos, pc98 o sun.
- En la mayoría de PCs se configuran las etiquetas *msdos* o *gpt*:

(parted) mklabel msdos

(parted) mklabel gpt



12.1 Creación de particiones

- Crear una nueva partición:

- Desde la tabla de particiones, hay que determinar los puntos de comienzo y fin de la nueva partición y qué tipo de partición debe ser.

Por ejemplo, para crear una partición primaria con un sistema de ficheros ext4 desde 1024 megabytes hasta 2048 megabytes:

```
(parted) mkpart primary ext4 1024 2048
```

- A continuación se puede visualizar la nueva partición creada con el comando **print**.
- Para asegurarse de que el kernel reconoce la nueva partición, se puede visualizar la salida de [/proc/partitions](#).
- La partición no tiene todavía un sistema de ficheros, para ello hay que formatearla.



12.1 Creación de particiones

○ Utilidad **fdisk**:

- Para usar la utilidad, hay que acceder como *root* y ejecutar el comando: **fdisk [dispositivo]**
- Aparece un prompt y, a continuación, se pueden empezar a introducir comandos.
- Si no se indica ningún dispositivo, se muestra una ayuda con el uso del comando.
- Comandos más usuales:
 - m → Muestra la ayuda.
 - p → Muestra la tabla de particiones.
 - q → Sale de la aplicación sin guardar los cambios.
 - w → Guarda los cambios y sale de la aplicación.
 - n → Crea una partición.
 - d → Borra una partición.
 - v → Verifica la tabla de particiones.
 - o → Crea una nueva tabla de particiones DOS vacía.
 - l → Lista los tipos de particiones conocidos.

12.1 Creación de particiones

- Para ver la tabla de particiones:

Orden (m para obtener ayuda): p

```
Disco /dev/sda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cilindros of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00086c02
```

Disposit.	Inicio	Comienzo	Fin	Bloques	Id	Sistema
/dev/sda1	*	1	32	256000	83	Linux
La partición 1 no termina en un límite de cilindro.						
/dev/sda2		32	925	7168000	83	Linux
La partición 2 no termina en un límite de cilindro.						
/dev/sda3		925	1186	2097152	82	Linux swap / Solaris
/dev/sda4		1186	1306	963584	5	Extendida
/dev/sda5		1186	1306	962560	83	Linux



12.1 Creación de particiones

- **fdisk -l**: Muestra la información para los dispositivos indicados. Si no se indica ningún dispositivo, muestra todas las particiones.

```
[root@profesor ~]# fdisk -l
```

```
Disk /dev/sda: 16.1 GB, 16106127360 bytes, 31457280 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Identificador del disco: 0x000bc4db
```

Disposit.	Inicio	Comienzo	Fin	Bloques	Id	Sistema
/dev/sda1	*	2048	2099199	1048576	83	Linux
/dev/sda2		2099200	23070719	10485760	83	Linux
/dev/sda3		23070720	27265023	2097152	82	Linux swap / Solaris
/dev/sda4		27265024	31457279	2096128	5	Extended
/dev/sda5		27269120	31457279	2094080	83	Linux

```
WARNING: fdisk GPT support is currently new, and therefore in an experimental phase.
```

```
Disk /dev/sdb: 1073 MB, 1073741824 bytes, 2097152 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: gpt
Disk identifier: D911E2A6-2835-458A-B669-5923A46B9C86
```

12.2 Formatear particiones

- Una vez se ha creado la partición, hay que formatearla con un determinado sistema de ficheros para que el sistema pueda hacer uso de este espacio.
- Para formatear una partición se utiliza el programa **mkfs**. Los datos que pueda tener la partición se borrarán.

mkfs [opciones] dispositivo

- Las principales opciones son:
 - **-t *fstype*** → Especifica el tipo de sistema de ficheros a crear.
 - **-c** → Comprueba el dispositivo en busca de bloques defectuosos antes de crear el sistema de ficheros.
 - **-v** → Produce una salida con más información. Esto es útil para comprobaciones.

- Por ejemplo:

mkfs -t ext4 /dev/sda6



12.3 Montaje de dispositivos

- En Linux los dispositivos físicos de la máquina son manipulados a través de ficheros especiales ubicados en el directorio `/dev`.
- Los discos duros, las particiones de estos, las unidades ópticas o los dispositivos de almacenamiento USB son ejemplos de dispositivos con los cuales interactuamos constantemente.
- Pero no se trabaja directamente con los ficheros de dispositivos sino que se incorporan al sistema de ficheros.
- Esta acción se conoce como **montar**, que en definitiva es asociar el dispositivo a un directorio determinado.
- Las particiones de los discos en Linux se montan en directorios como `/`, `/home` o `/usr`. Es decir, que, por una parte disponemos de los diferentes dispositivos físicos, representados por los ficheros especiales de dispositivo, y por otra su enlace o punto de montaje al sistema de ficheros, cuya raíz es `/`.
- El comando utilizado para ello es **mount**.



12.3 Montaje de dispositivos

○ Montar:

- `mount [opciones] -t type dispositivo directorio`

- `-t type` → indica el tipo de sistema de ficheros: ext3, ext4, iso9660 (CD-ROM), ntfs, vfat (FAT),...
- `-a` → monta todos los sistemas de ficheros listados en `/etc/fstab`.
- `-r` → monta el s.f. como sólo lectura.
- `-o opciones` → especifica diferentes opciones de montaje.
- Ejemplos:
 - `mount /dev/cdrom /media/cdrom`
 - `mount /dev/sdb1 /media/usb`

- Para saber qué dispositivos están montados en un momento determinado se puede ejecutar el comando `mount` sin parámetros.
- Para que los cambios sean permanentes, hay que configurar el fichero `/etc/fstab`.

○ Desmontar:

- `umount dispositivo|directorio`

○ Estas operaciones se tienen que realizar como superusuario.

12.3 Montaje de dispositivos

- El comando **df** muestra la utilización de espacio en disco de los sistemas de ficheros que están montados.

- **df -h**: muestra la información en megas y gigas.
- **df -T**: muestra también el tipo de sistema de ficheros.

```
[root@server alumno]# df -hT
```

S.ficheros	Tipo	Tamaño	Usados	Disp	Uso%	Montado en
/dev/sda2	ext4	9,8G	5,5G	3,8G	60%	/
devtmpfs	devtmpfs	907M	0	907M	0%	/dev
tmpfs	tmpfs	921M	128K	920M	1%	/dev/shm
tmpfs	tmpfs	921M	73M	848M	8%	/run
tmpfs	tmpfs	921M	0	921M	0%	/sys/fs/cgroup
/dev/sda1	ext4	477M	129M	319M	29%	/boot
/dev/sda5	ext4	1,5G	475M	923M	34%	/home
tmpfs	tmpfs	185M	16K	184M	1%	/run/user/1000
/dev/mapper/grupo-vol1	ext4	93M	1,6M	85M	2%	/media
tmpfs	tmpfs	185M	0	185M	0%	/run/user/0
/dev/sdc1	ext4	548M	876K	507M	1%	/mnt/datos

- Se puede especificar un sistema de ficheros en concreto:

- **df -h /dev/sdc1**:

```
[root@server alumno]# df -h /dev/sdc1
```

S.ficheros	Tamaño	Usados	Disp	Uso%	Montado en
/dev/sdc1	548M	876K	507M	1%	/mnt/datos




13. Usuarios y grupos




13. Usuarios y grupos

1. Introducción
2. Gestión de cuentas de usuario
3. Gestión de grupos
4. Otros comandos
5. Herramientas gráficas
6. Cambio de propietario y de grupo



13.1 Introducción

- Cada usuario dado de alta en el sistema tiene asignado un nº de usuario (**UID**). El sistema usa internamente este UID para realizar las comprobaciones de seguridad de los usuarios.
- Cada usuario pertenece al menos a un grupo, que se conoce como *grupo principal*.
- Los usuarios pueden pertenecer además a otros grupos, que se conocen como *grupos secundarios*.
- Los grupos facilitan la administración del sistema ya que se pueden otorgar privilegios a los grupos y de esta forma hacerlo indirectamente a los usuarios que conforman esos grupos.
- Al igual que sucede con los usuarios, cada grupo tiene asignado un nº (**GID**).
- **Para usar los comandos de gestión, hacen falta permisos de *root*.**



13.1 Introducción

- Cada usuario suele tener un directorio de trabajo, que por defecto es `/home/usuario`. En la mayoría de distribuciones, el único que tiene acceso a la información contenida en su directorio HOME es el propio usuario. En Ubuntu, el resto de usuarios sí que pueden acceder al HOME de los demás y ver el contenido.
- Los principales ficheros involucrados en la administración de usuarios y grupos son:
 - `/etc/passwd` → Información de las cuentas de usuario.
 - `/etc/shadow` → Información de las contraseñas de usuario. Sólo legible por el usuario *root*.
 - `/etc/group` → Información de los grupos definidos.
 - `/etc/skel` → Directorio que contiene los ficheros que se copiarán al HOME de un usuario al darlo de alta.



13.2 Gestión de cuentas de usuario

o `useradd [opciones] login`

- Añade un usuario al sistema.
- Opciones:
 - `-d homedir` → Indica cuál será el directorio *home* (si no se quiere crear en el lugar por defecto).
 - `-m` → Crea el directorio *home* del usuario.
 - `-g grupo` → Indica el nombre o GID del grupo principal.
 - `-u uid` → Indica el UID. Si no se especifica, asigna el siguiente nº.
 - `-s shell` → Indica el shell del usuario (`/bin/bash`, `/bin/sh`,...).
 - `-G grupo1,grupo2,grupo3,...` → Indica el grupo secundario o lista de grupos secundarios.
- En algunas distribuciones, como por ejemplo Suse, al crear un usuario, éste se añade a un grupo genérico llamado *users*. En otras distribuciones, como Red Hat o Ubuntu, se crea un grupo con el mismo nombre que el usuario y se le asigna como grupo primario.



13.2 Gestión de cuentas de usuario

o **passwd [login]**

- Asigna o actualiza la contraseña del usuario.
- Un usuario utilizará la orden `passwd` sin argumentos para cambiar su contraseña.

o **usermod [opciones] login**

- Modifica las características de la cuenta.
- Tiene las mismas opciones que `useradd`.
- Para añadir un grupo secundario, se usa `-G` junto con `-a`.

o **userdel [-r] login**

- Elimina la cuenta del usuario del sistema.
- Con la opción `-r`, `--remove` → borra además su directorio *home*.

13.2 Gestión de cuentas de usuario

- Fichero de usuarios: `/etc/passwd`

Campo	Descripción
1	Login del usuario
2	Contraseña. Con valor x, se encuentra oculta en <code>/etc/shadow</code>
3	Identificador del usuario (UID)
4	Identificador del grupo (GID)
5	Nombre completo del usuario
6	Directorio <i>home</i>
7	Shell predeterminado

```
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
pulse:x:497:496:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42:./var/lib/gdm:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
abrt:x:173:173:./etc/abrt:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72:./:/sbin/nologin
alumno:x:500:500:Alumno:/home/alumno:/bin/bash
```

- Fichero de contraseñas: `/etc/shadow`

- Contiene:

- El login de cada usuario.
- La contraseña encriptada.
- Datos acerca de la contraseña:
 - Días que han pasado, desde el 1 de enero de 1970, hasta que la contraseña ha sido cambiada por última vez.
 - Días que deben pasar hasta que la contraseña pueda ser cambiada.
 - Días que han de pasar hasta que la contraseña deba ser cambiada.
 - Días antes de caducar la contraseña en el que se avisará al usuario de que debe cambiarla.
 - Días que pueden pasar después de que la contraseña caduque, antes de deshabilitar la cuenta del usuario
 - Días, desde el 1 de enero de 1970, desde que la cuenta está deshabilitada.

```
gdm:!!:15659:.....
rpcuser:!!:15659:.....
nfsnobody:!!:15659:.....
abrt:!!:15659:.....
sshd:!!:15659:.....
tcpdump:!!:15659:.....
alumno:$6$s6Fp0dK33m3DiGP5$qD/7IhPmkDNw50jV9axYP0Dp6iEQ3N6mphQ4gISzEjYp6MdMkNJRv
cXVnALBDn9EE2nZ29kMRYQ8WDaLKDiri.:15659:0:99999:7:::
```

- * → Cuenta bloqueada
- ! → No tiene contraseña



13.3 Gestión de grupos

○ **groupadd [opciones] nombre_grupo**

- Crea un grupo nuevo.
- Opciones:
 - -g, --gid *GID* → Indica el GID. Si no se especifica, asigna el siguiente n°.

○ **groupdel [opciones] nombre_grupo**

- Elimina un grupo existente. No se pueden eliminar grupos primarios con usuarios asignados.

○ **gpasswd [opciones] nombre_grupo**

- Administra grupos y permite configurarles una contraseña.
- Opciones:
 - -a *usuario* → Añade el usuario al grupo indicado (como grupo secundario).
 - -d *usuario* → Elimina el usuario del grupo indicado.

13.3 Gestión de grupos

- Fichero de configuración de grupos: `/etc/group`

Campo	Descripción
1	Nombre del grupo
2	Contraseña. No se suele usar.
3	Identificador del grupo (GID)
4	Usuarios que tienen este grupo como secundario.

```
rpcuser:x:29:  
nfsnobody:x:65534:  
gdm:x:42:  
stapusr:x:156:  
stapsys:x:157:  
stapdev:x:158:  
sshd:x:74:  
tcpdump:x:72:  
slocate:x:21:  
alumno:x:500:  
vboxsf:x:501:
```

13.4 Otros comandos

o `su [-] [login]`

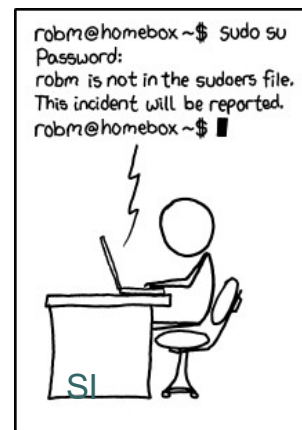
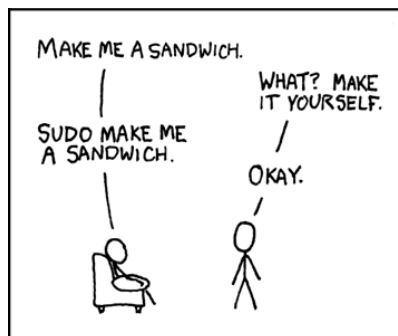
- Permite cambiar de usuario. Si no se especifica ninguno, por defecto se cambia a *root*.
- Si se usa con el guión (-), se cambia al HOME del nuevo usuario.

o `exit`

- Cierra la sesión de un usuario.

o `sudo [opciones] [comando]`

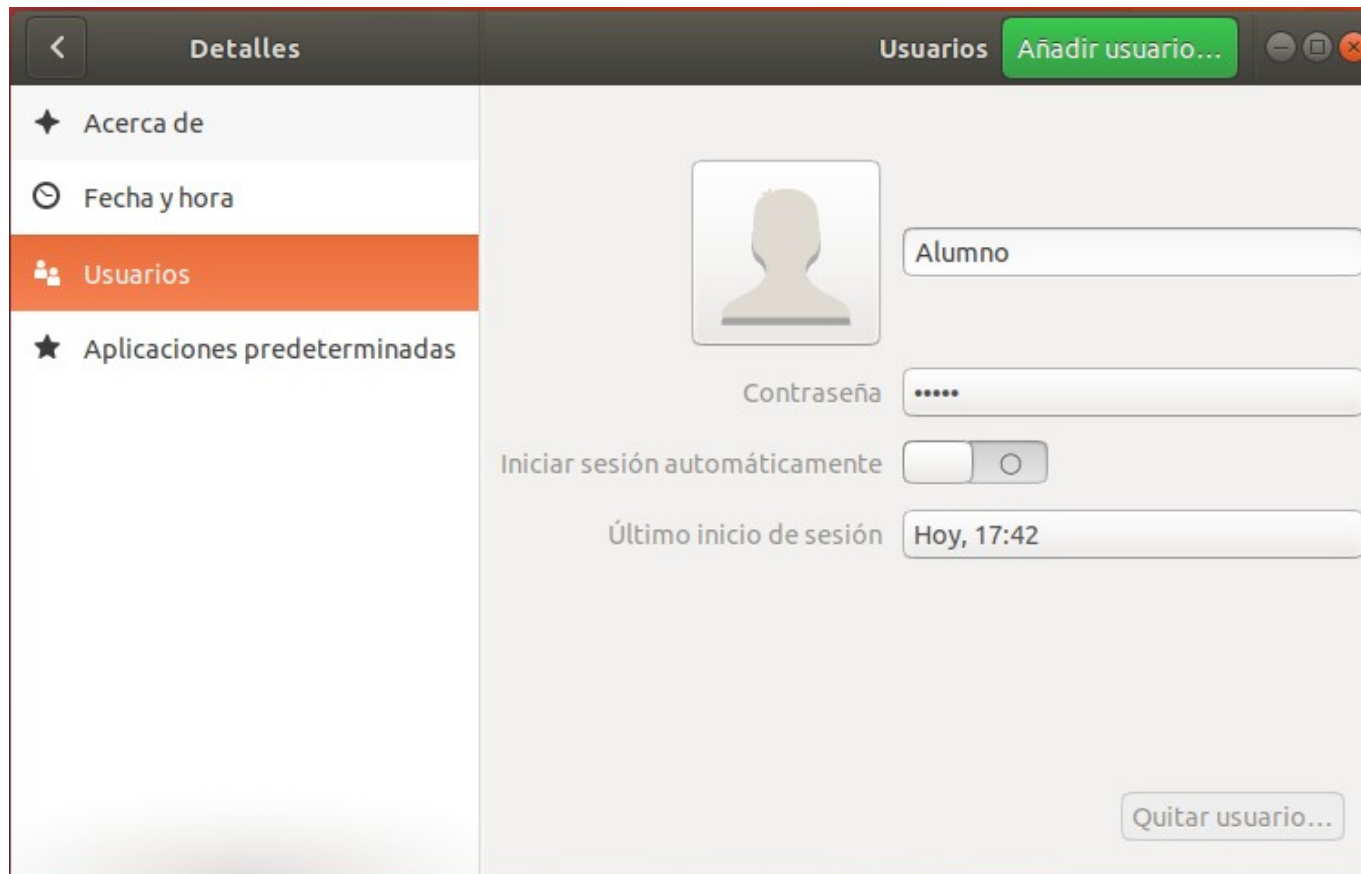
- Es una herramienta del sistema que permite a los usuarios realizar la ejecución de comandos como superusuario u otro usuario de acuerdo a como se especifique en el fichero `/etc/sudoers`, donde se determina quién está autorizado. Para editarlo → `visudo`





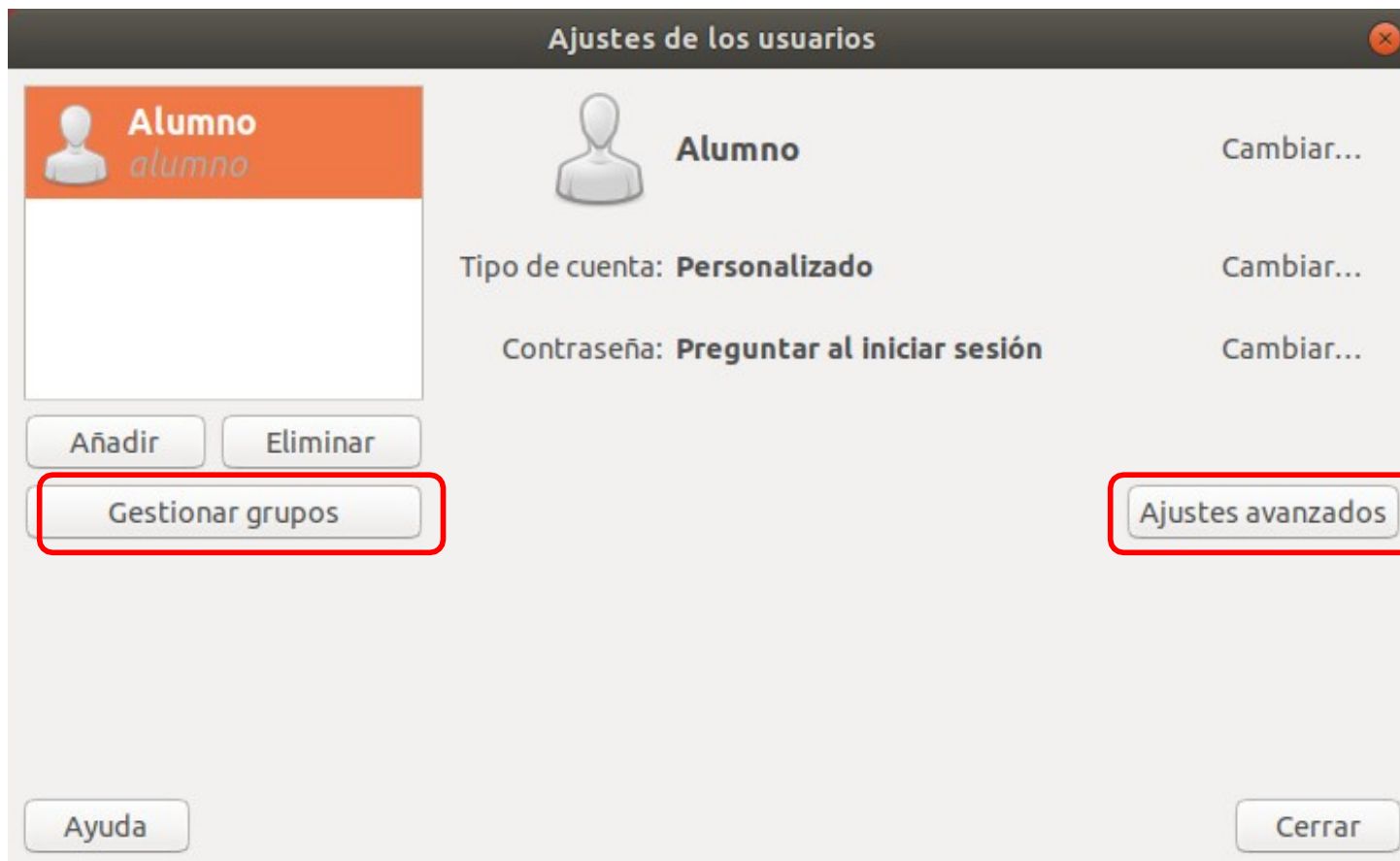
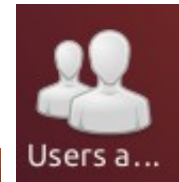
13.5 Herramientas gráficas

- La gestión de cuentas de usuario que incorpora Ubuntu está muy limitada.
- Configuración → Detalles → Usuarios
- Hay que pulsar en Desbloquear.



13.5 Herramientas gráficas

- Para una gestión más completa, hay que instalar el paquete "**gnome-system-tools**".
- Una vez instalado, se puede buscar la herramienta Users and groups.





13.6 Cambio de propietario y de grupo

○ **chown [-R] login:nombre_grupo fichero...**


- Cambia el usuario propietario y/o grupo de cada fichero.
- Ejemplos:
 - **chown user1 fich** → user1 pasa a ser el usuario propietario de fich.
 - **chown user1:group1 fich** → user1 pasa a ser el usuario propietario y group1 el grupo propietario de fich.
 - **chown :group1 fich** → group1 pasa a ser el grupo propietario de fich.

○ **chgrp [-R] nombre_grupo fichero...**

- Cambia el grupo propietario de cada fichero.



14. Permisos



14. Permisos

- Linux, es un sistema diseñado para el trabajo en red, por tanto, la seguridad de la información que se almacena en los servidores es fundamental, ya que es posible que muchos usuarios tengan acceso.
- En Linux, los permisos que los usuarios pueden tener sobre los ficheros se establecen en tres niveles:
 - Permisos del propietario.
 - Permisos del grupo.
 - Permisos del resto de usuarios.
- Cada fichero y directorio del sistema pertenecen a un *usuario* y *grupo* determinados. El usuario es el propietario, aquel que ha creado el fichero o el directorio.



14. Permisos

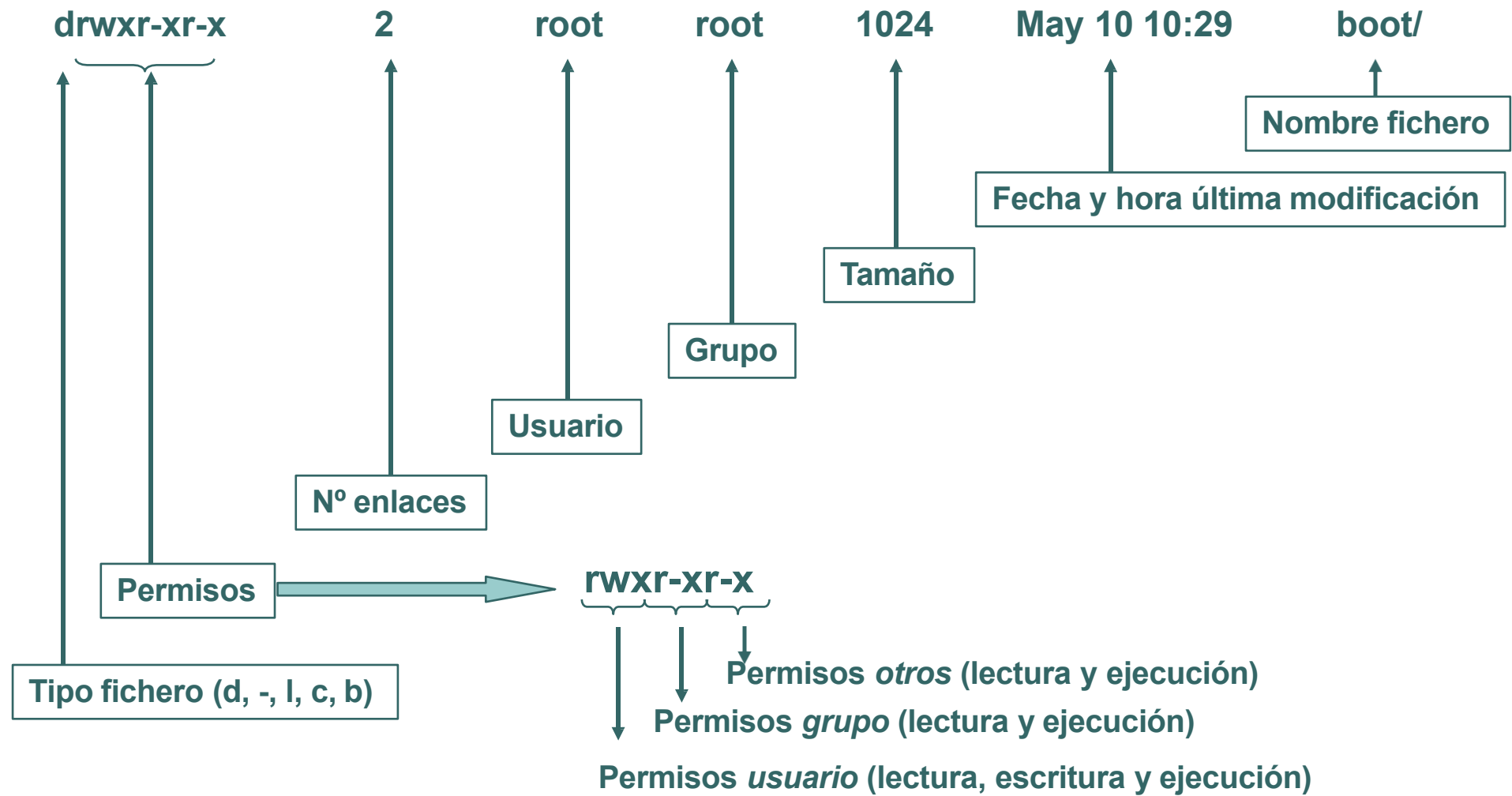
- Todos los ficheros y directorios tienen unos permisos, que son de los siguientes tipos:
 - r: lectura (*read*)
 - w: escritura (*write*)
 - x: ejecución (*execute*)
- Los permisos son diferentes en función de si se aplican a un fichero ordinario o a un directorio:
 - **Directorio:**
 - **r** → Indica que el directorio se puede leer, es decir, que se pueden ver los ficheros que contiene.
 - **w** → Indica que se pueden crear y borrar ficheros y directorios en él.
 - **x** → Indica que se puede acceder a él.
 - **Fichero:**
 - **r** → Indica que el fichero puede leerse.
 - **w** → Indica que el fichero puede modificarse.
 - **x** → Indica que el fichero puede ejecutarse.


14. Permisos

- Los permisos se representan mediante 9 bits, agrupados en tres grupos de tres bits:
 - *usuario* propietario del fichero: **u** (*user*).
 - usuarios adscritos al *grupo* propietario del fichero: **g** (*group*).
 - resto de usuarios (*otros*) registrados en el sistema: **o** (*others*).
- La representación se hace de la siguiente forma:
 - 1 → permiso otorgado
 - 0 → permiso no otorgado
- Esta combinación nos da un número en binario de 9 bits que posteriormente se transforma en octal.
- Ejemplos:
 - `rwX` → 111 → En octal es 7.
 - `r-x` → 101 → En octal es 5 .
 - `r--` → 100 → En octal es 4.



14. Permisos





14. Permisos

- ¿Qué permisos tienen los siguientes ficheros?
 - drwxrwxr-x
 - -rw-r--r--
 - -rw-r-----
 - drwxr-xr--
- ¿Cómo se expresarían estos permisos de forma numérica?
 - 11111101 → 775
 - 110100100 → 644
 - 110100000 → 640
 - 111101100 → 754



14. Permisos

o **chmod permisos fichero**

- Modifica los permisos del fichero.
- Los permisos se pueden indicar de diversas formas:
 - En octal.
 - Usando operadores que se aplican al usuario (**u**), grupo (**g**), otros (**o**) o todos (**a**, aunque no hace falta poner la a):
 - + → añade permisos.
 - - → quita permisos.
 - = → asigna permisos.
- Ejemplos:
 - `chmod 555 fich1` → `555=101101101=r-xr-xr-x` → permisos de lectura y ejecución para el *usuario*, el *grupo* y *otros*.
 - `chmod u+x fich1` → además de los permisos que tenga fich1, asigna permiso de ejecución para el *usuario*.
 - `chmod o-w fich1` → quita el permiso de escritura para *otros*.
 - `chmod +x fich1` → añade permiso de ejecución a *todos* (`=chmod a+x fich1`).
 - `chmod o=r fich1` → *otros* tienen sólo permiso de lectura.
 - `chmod o-rwx fich1` → quita todos los permisos para *otros*.

14. Permisos

- Para cambiar los permisos usando la interfaz gráfica se usa el navegador del sistema de ficheros Nautilus.
- Se ha de seleccionar el fichero o directorio con el botón derecho del ratón y pulsar *Propiedades*.
- En la pestaña Permisos, se seleccionan los permisos para el usuario, el grupo y otros:





14. Permisos

o En **resumen**:

- **r - permiso de lectura.** Un archivo debe poderse leer si se quiere examinar o copiar. Un directorio debe poderse leer si se quiere listar su contenido.
- **w - permiso de escritura.** Un archivo debe poder escribirse si desea modificarlo, eliminarlo o renombrarlo. Un directorio debe poder escribirse para agregar o eliminar archivos en él.
- **x - permiso de ejecución.** Un archivo con permisos ejecutables es aquel que el usuario. puede procesar, como por ejemplo un programa. Un directorio debe ser ejecutable si quiere tener acceso a cualquiera de sus subdirectorios.



15. Compartir recursos

15. Compartir recursos

- Para poder compartir directorios en Ubuntu, hace falta el paquete **samba**. Este paquete, además permite compartir directorios con equipos con Windows.
- Para compartir un directorio, hay que hacer clic con el botón derecho sobre el directorio y elegir entre:
 - La opción *Recurso compartido de red local*.
 - La opción *Propiedades* y a continuación la pestaña *Recurso compartido de red local*.





16. Gestión de procesos



16. Gestión de procesos

- A cada proceso, en el momento de su creación, se le asocia un número único que lo identifica → **PID**.
- Los procesos tienen además otras informaciones asociadas, como:
 - El **usuario** que lo ejecuta.
 - La **hora** en que comenzó.
 - La **línea de comandos** asociada.
 - El **estado**. Ejemplos: sleep, running, zombie, stopped, etc.
 - La **prioridad** que indica la facilidad del proceso para acceder a la CPU. Oscila entre -20 y 19, donde -20 es la mayor prioridad.
- El primer proceso que se crea en el sistema se llama **systemd** y tiene como PID = 1. Este proceso es el encargado de iniciar el resto de procesos, incluidos los *demonios* o *servicios*, y no puede eliminarse del sistema.
- Los **demonios** son procesos que el sistema operativo invoca para proporcionar servicios.
- Los procesos se pueden ejecutar en primer plano o en segundo plano.



16. Gestión de procesos

- **ps [-opciones]**: Muestra los procesos actuales (sin opciones muestra los de la terminal actual). Opciones:
 - -e: Todos los procesos.
 - -f: Formato largo.
 - -u *usuario*: Sólo los del usuario indicado.
 - -o: Muestra un formato definido por el usuario:
 - %cpu → porcentaje de uso de la CPU.
 - %mem → porcentaje de uso de la memoria.
 - comm → comando que ejecuta el proceso.
 - cputime → tiempo de CPU utilizado.
 - pid → identificador del proceso.
 - s → estado del proceso.

Ej: **ps -eo pid,ppid,s,cmd,%cpu**
- **pstree**: Muestra el árbol de procesos.
- **top**: Muestra los procesos actuales, actualizando la información periódicamente.

16. Gestión de procesos

- o **kill [-señal] pid**: Envía una señal a un proceso.

Algunas señales son:

- 9 → “Mata” el proceso.
- 15 → Termina el proceso, haciendo antes lo que necesite. Es la que se envía por defecto.
- 2 → Interrumpe un proceso (igual que pulsar **Ctrl+c**).
- 19 → Suspende un proceso (igual que pulsar **Ctrl+z**).
- Con -l se ve un listado de las señales.

Ej: **kill -2 4223**

kill -9 147

kill 5678 → terminación normal.

- o **comando &**: Ejecuta el proceso en segundo plano para que la terminal quede libre.

Ej: **firefox &**



17. Servicios



17. Servicios

- Un servicio o demonio (*daemon*) es un proceso que está en memoria esperando una señal y que da soporte a otros demonios o programas de usuario.
- Los servicios se llaman ***nombre.service***. Para realizar acciones sobre un servicio se usa el comando ***systemctl***.
 - ***systemctl start nombre.service*** → Inicia un servicio.
 - ***systemctl stop nombre.service*** → Para un servicio.
 - ***systemctl restart nombre.service*** → Reinicia un servicio.
 - ***systemctl status nombre.service*** → Comprueba el estado.
 - ***systemctl list-units --type service --all*** → Muestra el estado de todos los servicios.



18. Programación de tareas



18. Programación de tareas

- Programar una tarea es planificar que un comando, script o programa se ejecute en un instante determinado o de forma periódica.
- Hay dos mecanismos para programar tareas:
 - **at** → realiza una tarea sólo una vez.
 - **cron** → permite programar tareas que se ejecuten de forma periódica.



18. Programación de tareas

○ Programación de una tarea periódica:

- En los sistemas GNU/Linux ya se encuentran preconfiguradas ciertas tareas de forma periódica. El demonio que se encarga de gestionar las tareas programadas es el **crond**.
- La utilidad **cron** se invoca cada minuto y ejecuta las tareas que correspondan.
- Los usuarios pueden editar su fichero **crontab** y configurar la ejecución de tareas.
 - El fichero crontab del sistema es **/etc/crontab**.
 - Los ficheros crontab de los usuarios se guardan en **/var/spool/cron/crontabs**.

18. Programación de tareas

- Para modificar los crontabs se usa el comando:
`crontab opción`
- Opciones:
 - -e → Edita o crea el fichero crontab.
 - -l → Lista el contenido del fichero.
 - -r → Elimina el fichero crontab del usuario.
- En el fichero crontab, se añade una línea para cada tarea, siguiendo la siguiente sintaxis:

`* * * * * [usuario] comando o script`

- Día de la semana (1-7) (El domingo es 0 ó 7)
- Mes (1-12)
- Día del mes (1-31)
- Hora (0-23)
- Minuto (0-59)

18. Programación de tareas

- El día de la semana se puede indicar también usando las 3 primeras letras de su nombre en inglés (sun, mon, etc.).
- Y lo mismo para el mes (jan, feb, etc.).
- En cada campo se puede indicar:
 - Todos los valores (*).
 - Un valor.
 - Varios valores, separándolos mediante comas (1,4,6).
 - Un rango, usando el guión (2-5).
 - Un intervalo (* / 15 → cada 15 minutos).
- Ejemplos:
 - `0 3 * * 1 script.sh` → Los lunes a las 3 de la madrugada.
 - `0 0 * * 1-5 /home/usuario/script.sh` → Los días laborables a las 12 de la noche.
 - `*/20 * 1,15 * * script.sh` → Los días 1 y 15 de cada mes, cada 20 minutos.



19. Copias de seguridad

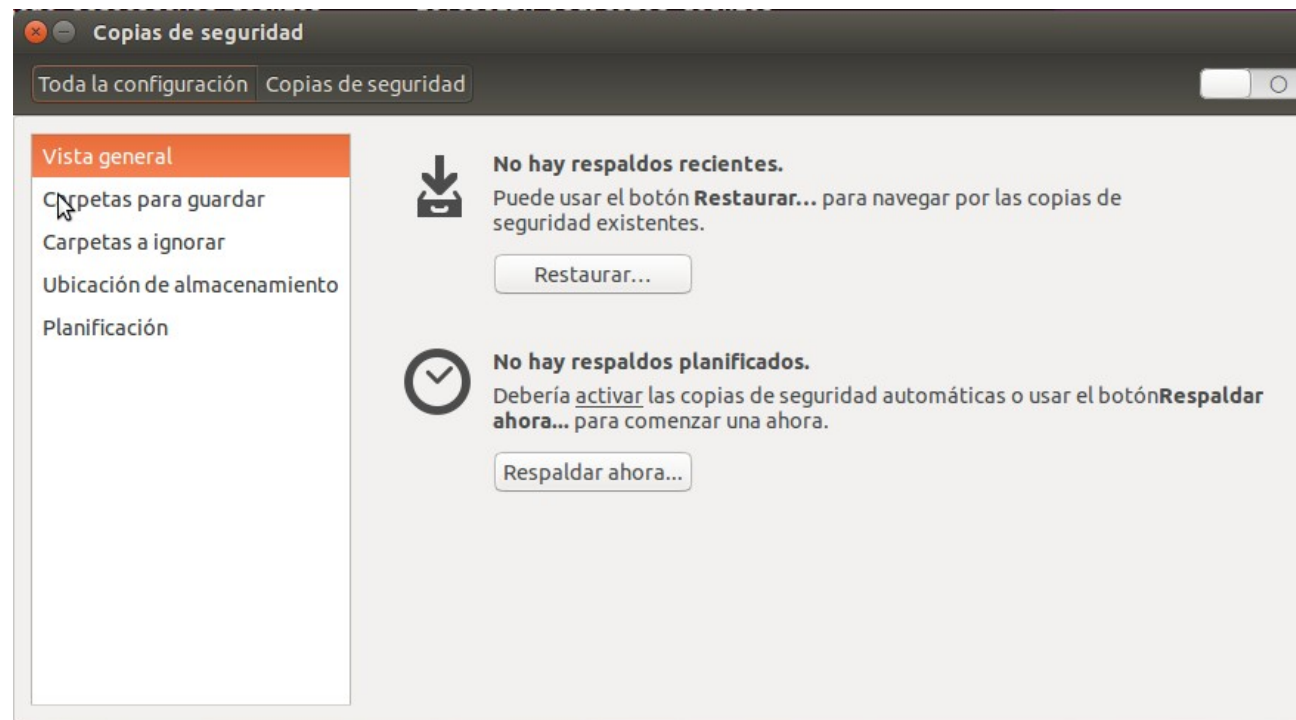


19. Copias de seguridad

- En cualquier sistema es necesario hacer copias de seguridad de forma periódica para proteger los datos de una posible pérdida, debida a un borrado accidental, a un virus, a un fallo del disco duro o a un fallo del sistema.
- De esta forma, se puede dejar el sistema como estaba cuando se hizo la última copia antes del fallo.
- Recomendaciones:
 - Hacer las copias en un dispositivo diferente al que contiene los datos.
 - Realizar copias regularmente.
 - Programar la realización de las copias de seguridad para evitar olvidos.
 - Organizar bien las copias que se van realizando.
 - Comprimir la copia para que ocupe menos espacio.
- Hay que decidir qué información se va a guardar (como mínimo: /home y /etc).

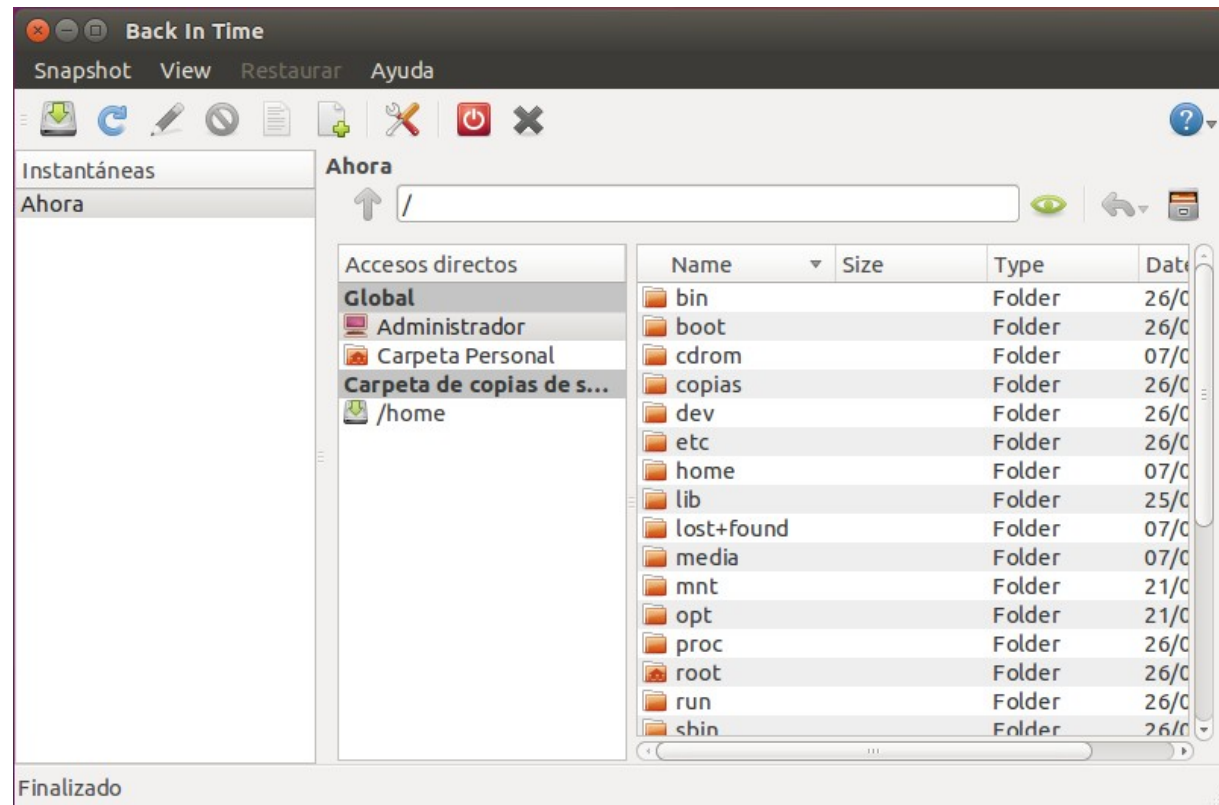
19. Copias de seguridad

- En algunas distribuciones hay herramientas para hacer copias de seguridad y en otras no.
- Ubuntu incorpora una herramienta que se llama **Copias de seguridad**.




19. Copias de seguridad

- En caso de que no disponer de una herramienta de copias de seguridad, hay varios programas disponibles, como [Back In Time](#).
- En Ubuntu, se puede instalar desde la herramienta Software.



19. Copias de seguridad

- Hay dos formas de ejecutar la aplicación: normal o como root. Si se ejecuta normal, sólo se podrá hacer copia de los ficheros y directorios en los que se tenga permiso.
- Al ejecutarla por primera vez, aparece una pantalla en la que hay que configurar algunas preferencias como:
 - General → Dónde guardar las copias y habilitar la programación.
 - Incluir → Qué directorios y ficheros incluir.
 - Excluir → Qué directorios, ficheros o patrones excluir.
 - Auto eliminar → Eliminar copias cuando se cumpla una condición.
- Estas preferencias se pueden modificar luego → 
- Para hacer una copia (*snapshot*) hay que incluir al menos un fichero o un directorio.
- Para realizar la copia → 