CURSO GUADALINEX
Primeros pasos con Debian

Juan Alonso - Fermín Rubio - Paco Villegas

15 de marzo de 2004

Índice general

Capítulo 1

Inicio del Sistema

> Al proceso de encendido de la computadora, es decir, el *arranque*, proviene del inglés *booting*, palabra que deriva del término *bootstrapping*, que es una alusión a la idea de que una computadora se levanta a sí misma jalándose de las correas de sus botas, como queriendo dar a entender que fragmentos pequeños de código simple arrancan a fragmentos más grandes y más complejos hasta hacer que todo el sistema acabe por funcionar. (*LINUX. Recursos para el usuario*. JAMES MOHR)

Los ordenadores actuales usan los mismos componentes generales y la misma secuencia general para iniciar un sistema operativo. Tras el control hardware del proceso de arranque, se carga el siguiente nivel de software a ejecutar (cargador de inicialización) y éste toma el control iniciando el proceso de carga del sistema operativo, comienza la carga de Linux.

Linux es "diferente" y esto se pone manifiesto ya desde el arranque del sistema, dado que dispone de una gran cantidad de opciones de configuración de inicio.

En este capítulo vamos a estudiar algunas cuestiones que aconsejan una primera lectura sin entrar en "profundidad" sobre todos los temas tratados. A medida que el curso avance, sí que es deseable tener una visión general clara sobre ellos.

1.1. Gestores de arranque

1.1.1. Grub

Si hemos cargado el gestor de arranque GRUB (*GRand Unified Boot loader*) en el MBR (*Master Boot Record*), es él el que se encarga de controlar el arranque del sistema. El MBR ocupa los primeros 512 bytes del disco duro. Se encuentra en el primer sector del disco (cilindro 0, cabeza 0, sector 1). Además del código necesario para iniciar la carga del sistema, contiene la tabla de particiones del disco.

<table>
<tr><td colspan="2">Código de lectura de arranque de la partición activa o GRUB</td></tr>
<tr><td rowspan="4">Tabla particiones</td><td>1ª partición</td></tr>
<tr><td>2ª partición</td></tr>
<tr><td>3ª partición</td></tr>
<tr><td>4ª partición</td></tr>
<tr><td>Firma</td><td>Valor AA55h</td></tr>
</table>

Una vez que se inicia, GRUB pasa el control del sistema al núcleo de Linux y éste a su vez al padre de todos los procesos: el proceso *init*. Pero no corramos tanto, el estudio de ese proceso corresponde al apartado siguiente.

Continuemos con el GRUB[1]: GRUB es un *boot loader*[2] que permite iniciar la máquina según unas directivas concretas, además permite pasar parámetros al kernel e iniciar otros sistemas que no sean Linux.

Tras cargarse en la memoria RAM la parte de GRUB contenida en el MBR se inicia la carga del fichero de configuración del gestor de arranque (`/boot/grub/menu.lst`). Si todo el proceso es correcto veremos en nuestro monitor[3]

y un menú con el que poder cargar el sistema operativo que deseemos.

Sea cual sea el entorno elegido, si dejamos pasar un tiempo él sólo comienza la secuencia de arranque de Linux.

Pero, STOP; congelemos el proceso de arranque y analicemos un poco qué podemos hacer:

1.1.2. Interfaces de GRUB

De Menú

Con esta "ventanita" inicial ya tenemos trabajo suficiente. Si nos fijamos en el gráfico podemos ver que en el texto inferior, además de indicarnos la posibilidad de usar las flechas de cursor nos avisan de que podemos pulsar la tecla

e para modificar los paramétros de arranque del sistema operativo resaltado

c para acceder a la interfaz de línea de comandos.

Del editor de entrada de menú

Si pulsamos **e** en la ventana anterior veremos algo similar a:

[1] Para encontrar información más amplia sobre GRUB

- En castellano: `http://www.linux.cu/manual/grub/grub.es.html`
- La documentación instalada en: `/usr/share/doc/grub` y las páginas man del programa
- La web del programa: `http://www.gnu.org/software/grub/`

[2] Gestor de arranque.

[3] Esto tiene trampa. Mi pantalla es negra y no sale el logo de guadalinex: cierto, estas capturas se han hecho después de hacer una práctica que aparece un poco más adelante, en ella se explica cómo conseguir un GRUB más atractivo (Práctica en la página 9)

Desde ella podemos:

o agregar una línea de comandos después de la línea en vídeo inverso

O agregar una línea antes de la línea actual

e modificar la línea actual

d eliminar la línea actual

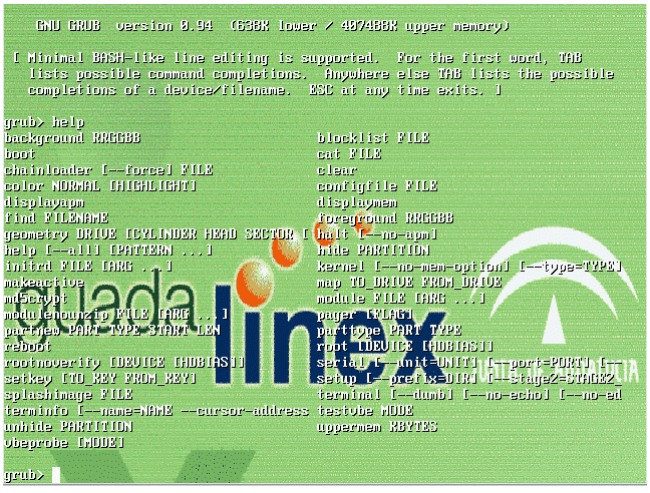
b iniciar el arranque del sistema

Esc para omitir los cambios y volver a la pantalla anterior

c para acceder a la interfaz de línea de comandos.

De línea de comandos

Si desde la pantalla inicial optamos por pulsar sobre **c**, accedemos a la ventana de línea de comandos. Para ver qué comandos tenemos disponibles podemos usar `help`:



Soporta la autocompletación de comandos (igual que las shell de Linux) mediante la tecla **[Tab]**[4]

Aclaremos cómo interpreta GRUB las particiones de nuestros discos duros:

[4] Es decir, no tenemos que conocer el nombre completo de un comando, si escribimos las letras iniciales de ese comando y después **TAB** se autocompleta el comando si hay uno sólo con esas letras iniciales, si hay varios se muestran en pantalla para ver cuál elegimos.

`hd0` primer disco duro (IDE o SCSI)

`hd1` segundo disco duro

...

En cuanto a las particiones se numeran secuencialmente desde el número 0.

Así por ejemplo `(hd1,3)` representaría la cuarta partición del segundo disco duro de nuestro sistema

➥ **Para practicar**

Para poder realizar el ejemplo que sigue hay que adecuarlo a nuestro sistema:

- la línea 3 parte de la base de que tenemos un sistema Windows 9x en la primera partición del maestro del primer canal IDE
- el cuarto comando sólo se podrá ejecutar tal cual si `/boot` está en la segunda partición del primer disco duro.
- Otra nota: cuidado con el teclado, hay que localizar las teclas adecuadas ya que todavía no puede estar en “castellano”.

Ejecutar desde la ventana de línea de comandos de GRUB[5]:

```
grub>help
grub>displaymem
grub>cat (hd0,0)/autoexec.bat
grub>cat (hd0,1)/boot/grub/menu.lst
```

Más adelante, cuando analicemos los niveles de arranque estudiaremos algunos parámetros que se le pueden pasar al núcleo. Por ahora, permitamos que GRUB continúe su labor y que pase el control al núcleo de Linux.

```
Initializing Cryptographic API
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 4096 buckets, 32Kbytes
TCP: Hash tables configured (established 32768 bind 32768)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
kjournald starting.  Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Freeing unused kernel memory: 140k freed
INIT: version 2.85 booting
Starting Bootlog daemon: bootlogd.
Loading /etc/console/boottime.kmap.gz
Activating swap.
Adding Swap: 305192k swap-space (priority -1)
Adding Swap: 262136k swap-space (priority -2)
Checking root file system...
fsck 1.35-WIP (21-Aug-2003)
/dev/hda2: clean, 92667/486720 files, 495688/971932 blocks
EXT3 FS 2.4-0.9.19, 19 August 2002 on ide0(3,2), internal journal
Real Time Clock Driver v1.10e
System time was Sat Mar  6 15:01:56 UTC 2004.
Setting the System Clock using the Hardware Clock as reference...
System Clock set. System local time is now Sat Mar  6 15:01:57 UTC 2004.
Calculating module dependencies... _
```

1.1.3. Fichero /boot/grub/menu.lst

Le ventaja que presenta GRUB sobre otros gestores de arranque (LILO por ejemplo) es que GRUB no necesita que su fichero de configuración esté en el MBR, es decir, sólo se instala el programa y él se encarga de leer la configuración desde un fichero externo localizado en algún lugar de nuestro sistema. Además, usar este método permite que:

- Si nos equivocamos en la configuración de GRUB siempre podremos arreglarlo con los interfaces anteriores.

[5] Recordar de nuevo que es posible usar la autocompletación de comandos.

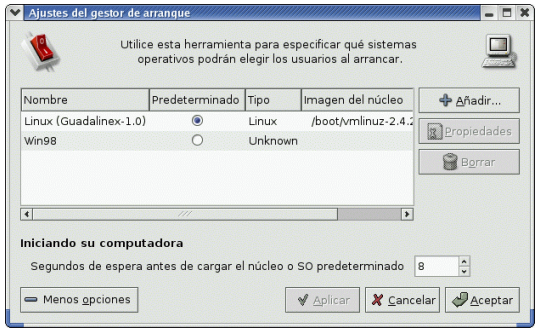
- Cada vez que modificamos el fichero de configuración de GRUB no tenemos que sobreescribir el MBR.

Disponemos de una herramienta gráfica de configuración de GRUB que podemos ejecutar bien desde una xterm

`#boot-admin`

o bien desde los menús

 **→Aplicaciones→Herramientas de Sistema→Panel de Control→Arranque**



¿Y cuál es el fichero de configuración de Grub? pues el fichero `/boot/grub/menu.lst`

Analicemos el de un sistema modelo (es distinto del que tenéis en vuestros equipos)

Las líneas que comienzan con # son comentarios (algo que ya veremos es estándar en Linux) que se han añadido para explicar el fichero[6].

```
$ cat /boot/grub/menu.lst
# sistema operativo por defecto, en mi caso el de título Linux ...
default=0
# tiempo de espera (en segundos) antes de iniciar
# la carga del sistema de forma automática
timeout=10
# Si la opción de arranque por defecto da error, en vez de esperar
# a que el usuario haga algo, empezar de nuevo pero con
# la opción 1 (0 significa la primera opción)
fallback 1
# gráfico de fondo por defecto
#(en una práctica posterior veremos cómo se pone)
splashimage=(hd0,1)/boot/grub/splash.xpm.gz
# texto que veremos en el menú
title Linux (Guadalinex 1.0)
   # Establece y monta la partición “root”: partición de arranque
   root (hd0,1)
   # núcleo a cargar y opciones de arranque: tengo una grabadora
   kernel /boot/vmlinuz-2.4.22-14 ro root=/dev/hda2 hdc=ide-scsi
# submenú de arranque del Windoze, y texto que veremos en el menú principal
title Win98
   # establece la partición de arranque pero sin montarla
   rootnoverify (hd0,0)
   # carga el fichero del primer sector de la
   # partición 0 del disco 0: partición Windoze
  chainloader +1
```

↬ ¿Cómo puede ser el fichero de configuración si en vez de dos sistemas son tres?. Un ejemplo en el que por defecto se carga el sistema DOS (`default=1`) podría ser:

[6] Con el comando `cat` mostramos por pantalla el contenido del fichero pasado como argumento.

```
$ cat /boot/grub/mnt.lst
default=1
timeout=10
password --md5 $1$6U7Tn/$Pr3691BPbD/9nD8DZMg5A/
splashimage=(hd0,2)/boot/grub/splash.xpm.gz
title Linux
    root (hd0,2)
    kernel /boot/vmlinuz-2.4.22 ro root=/dev/hda2 hdc=ide-scsi
title DOS
    rootnoverify (hd0,0)
    chainloader +1
title XP
    rootnoverify (hd0,1)
    chainloader +1
```

Si por alguna razón queremos eliminar este gestor de arranque podemos optar por reiniciar el sistema con un disco de Win 9x/ME y ejecutar el comando `fdisk /MBR`

Contraseñas

1. Tenemos la posibilidad de poner una contraseña al gestor de arranque. Para esto (como root) hay que editar el fichero `/boot/grub/menu.lst` y añadir una línea del tipo [7]:

   `password --md5 $1$6U7Tn/$Pr3691BPbD/9nD8DZMg5A/`

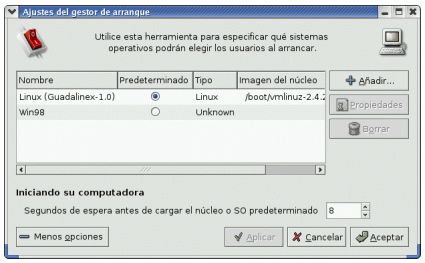
   Si usamos esta opción, es necesario introducir la contraseña (con **p**) para poder acceder a las opciones extendidas de GRUB.

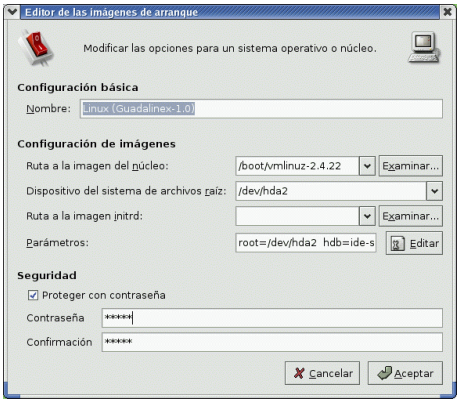
   Para desactivar esta opción sólo tenemos que comentarla con # o borrar la línea. Podemos generar la cadena encriptada asociada a una contraseña usando el comando:

   ```
   $/sbin/grub-md5-crypt
   ```

   Después de introducir la contraseña nos la devolverá encriptada con el algoritmo MD5, esa es la cadena que hay que introducir tras `md5`.

2. Con el programa `boot-admin`





podemos introducir “otra contraseña” si pulsamos sobre **Propiedades** (con Linux marcado en vídeo inverso) y en la ventana que aparece marcamos la opción de introducir contraseña. Pero no se trata de una contraseña para acceder a los menús de GRUB (como hemos visto antes), se trata de la contraseña que tendremos que introducir para acceder al sistema Linux para el que la escribamos.

[7] La contraseña utilizada es "linux".

Instalar GRUB

Si trabajamos con otro gestor de arranque y, una vez instalado Linux, deseamos cambiar a GRUB lo podemos hacer usando el comando `grub-install`. Cuando el fichero de configuración de GRUB esté ajustado a nuestro sistema, instalaremos GRUB en el MBR con:

```
#grub-install /dev/hda
```

➥ **Para practicar** Si no hicimos un disco de inicio durante la instalación de Guadalinex y GRUB no se instaló bien, podemos hacer un disco/instalar GRUB como sigue:

1. Iniciamos el ordenador desde el lector de CDs con el CD1 de la distribución.
2. Cuando se haya cargado el SO, hay que abrir una Xterm
3. En esa terminal debemos ejecutar los comandos
   a) Vamos a trabajar como administrador[8]
   ```
   su
   password
   ```
   b) Vamos a crear un directorio en el que "poner" el Guadalinex de nuestro disco duro y lo montamos[9]:
   ```
   mkdir sysimage
   mount -t ext3 /dev/hda2 sysimage
   ```
   Suponemos que el disco con el que estás trabajando es el maestro del primer canal IDE y queGuadalinex está instalado en la segunda partición primaria (hda2)
   ```
   chroot sysimage
   ```
   Nos permite trabajar como si el árbol de ficheros partiera de ese directorio, se trata del directorio en donde se monta nuestro sistema Linux.
4. Podemos optar por:

   Instalar GRUB: Con el editor `gedit` podemos modificar el fichero `/boot/grub/menu.lst` hasta ajustarlo a nuestra máquina, una vez que se adecúe a nuestro sistema ejecutaremos:
   `#grub-install /dev/hda`

   Creación de un disco de inicio: Para disponer de grub en un disco hay que realizar (como root) una serie de pasos[10] que pasamos a enumerar:
   a) Ejecutamos los comandos (pondremos # para indicar que el comando se ejecuta como root, aunque no haya que escribirlo):
   ```
   #mv /boot/vmlinuz-2.4.23 /
   #tar -czf boot.tar.gz /boot
   #/sbin/mkbimage -t 1.44 -f boot.tar.gz
   ```
   b) Introduciremos un disco en la unidad de disquetes y ejecutaremos el comando:
   ```
   #dd if=1.44.image of=/dev/fd0
   ```
   Con el primer comando (`mv`) movemos un núcleo no usado al raíz del sistema de ficheros, si no lo hacemos no podremos meter en un disco de 1.44MB el contenido del subdirectorio `/boot`. Con el segundo (`tar`), comprimimos y empaquetamos el directorio `/boot` (lugar donde se almacena el kernel de Linux); el tercero (`mkbimage`) nos permite crear una imagen de ese directorio usando el fichero comprimido y empaquetado. Por último "copiamos" ese fichero imagen (`1.44.image`) en el disco flexible (`/dev/fd0`).
   Listo, con esos comandos disponemos de un disquete de arranque por si borramos "accidentalmente" el gestor de arranque del MBR. Antes de nada deberíamos comprobar (cambiando en la BIOS la secuencia de arranque) que el disco está bien realizado.

[8] No hay que escribir nada en la contraseña del administrador cuando trabajas con el cdrom, solo pulsar **Intro**.

[9] Podemos comprobar de qué partición se trata escribiendo:
```
fdisk -l /dev/hda
```

[10] El comando clásico (en Debian) para hacer discos de inicio si usamos lilo es `mkboot` (con Guadalinex edu habría que usar `mkboot` sin más)

Password del root Veamos cómo cambiar la contraseña del root (por si se nos olvida). Para eso, antes de salir ejecuta el comando:
`#passwd`
e introduce la nueva contraseña. Contraseña cambiada. Moraleja: el único sistema medio seguro es aquel al que no tenga nadie acceso

5. Para terminar sólo tenemos que ejecutar el comando `exit` (un par de veces).

Para saber más: el capítulo del manual antes comentado y el comando:
`$info grub`

➥ **Para practicar**

1. Si tenemos dos sistemas operativos instalados, modificar el fichero `/boot/grub/menu.lst` para que cambie el sistema de arranque por defecto:
   - Bien cambiando la línea: `default`
   - Bien permutando los bloques existentes para los sistemas operativos.
2. Añadir una contraseña a GRUB que impida que se pueda acceder a las opciones de edición sin más.
3. **Imagen de fondo en el menú de GRUB**: Vamos a ver cómo poner una imagen de fondo en el menú de inicio de GRUB. La imagen de la que vamos a partir es
   `/usr/share/pixmaps/guadalinex/guadalinex-background.png`
   Manos a la obra
   a) Situémonos en la ruta en donde está la imagen
   ```
   cd /usr/share/pixmaps/guadalinex
   ```
   b) Convirtamos la imagen al formato deseado[11] (formato xpm de dimensiones 640x480 y con una profundidad de 14 colores)
   `convert -resize 640x480 -colors 14 guadalinex-background.png fondo.xpm`
   c) La imagen la comprimiremos en formato gz.
   ```
   gzip fondo.xpm
   ```
   d) Movamos el fichero obtenido a donde debe estar
   ```
   mv fondo.xpm.gz /boot/grub
   ```
   e) Modifiquemos el fichero de configuración de grub de forma que cargue la imagen en el inicio:
   ```
   ...
   timeout=10
   # gráfico de fondo por defecto
   splashimage=(hd0,1)/boot/grub/fondo.xpm.gz
   # texto que veremos en el menú
   title Linux
   ...
   ```

1.1.4. Por si se opta por LILO.

Este apartado lo hemos mantenido como información añadida. Se justifica desde la perspectiva de que LILO es el gestor de arranque por defecto de Guadalinex EDU. Por tanto, salvo que deseemos ampliar sobre este tema o porque tengamos Guadalinex EDU instalado, lo normal es que "saltemos" hasta 1.2 en la página 12.

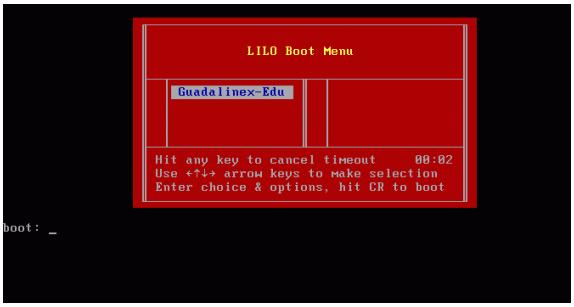
LILO es otro gestor de arranque que permite controlar qué núcleo o sistema operativo va a ser cargado e iniciado. Con él tenemos la posibilidad de pasar parámetros al kernel usando el archivo `/etc/lilo.conf`.

Para encontrar información más amplia sobre LILO, se puede consultar en:

[11] También se puede hacer con Gimp.

- `/usr/share/doc/lilo`, en este subdirectorio además de ejemplos tenemos información bastante amplia sobre él así como el manual de usuario.
- *Linux Instalación y Primeros Pasos*, de MATT WELSH.
- Los HOWTOs/mini-HOWTOS:
  - Linux-DOS-Win95-OS2.Como
  - Linux-NT-Loader
  - Discos-Grandes-Como
  - mini-HOWTO LILO
  - HOWTO BootPrompt
  - HOWTO BootDisk
- En `http://www.linux-es.com/Faq` está la FAQ sobre Linux para principiantes, en su punto 4[12] hace un estudio sobre él. En esa FAQ podemos encontrar un trabajo bastante bueno sobre LILO, de hecho, bastantes de las "ideas" que vienen a continuación están sacadas de él.

Cuando se ejecuta LILO, a la vez que se van mostrando las letras en la pantalla de nuestro ordenador se van cargando una serie de componentes de forma secuencial, si todo el proceso es correcto[13] veremos en nuestro monitor un menú con el que poder cargar el sistema operativo que deseemos.



Si pulsamos *intro* entraremos en el Sistema que hayamos definido por defecto. Sea cual sea el entorno elegido, si dejamos pasar un tiempo él sólo comienza la secuencia de arranque de Linux.

Podemos modificar los parámetros de LILO modificando el fichero `/etc/lilo.conf`. Comentemos cómo modificar algunos parámetros de forma manual[14], supongamos que nuestro fichero `/etc/lilo.conf` es:

```
#sección Global
boot=/dev/hda
vga=normal
```

[12] Sobre LILO

[13] Si surgen problemas en el arranque la presencia o ausencia de las letras de LILO nos permitirá diagnosticar el porqué del fallo. En la FAQ anteriormente mencionada se explica esto como sigue:

- **"L" error**: El primer tramo del código ha sido cargado pero no el segundo. Esto se debe generalmente a un error físico en el sector de botado o a un problema de geometría del disco.
- **"LI"**: El segundo tramo del código ha sido invocado, pero no ha podido iniciarse. Esto sucede cuando hay un error de geometría de disco o se ha movido `/boot/boot.b` sin reinstalar LILO (`/sbin/lilo`).
- **"LIL"**: El segundo tramo del código se ha iniciado pero no puede ubicar los datos que necesita en el fichero de mapeado. Esto puede deberse a un error físico del dispositivo de arranque o a un problema en la geometría del disco.
- **"LIL?"**: El segundo tramo del código se ha cargado en una dirección de memoria equivocada. Esto se debe a un error en la geometría del disco o cuando se ha movido `/boot/boot.b` sin reinstalar LILO (`/sbin/lilo`).
- **"LIL-"**: Los datos en el fichero de mapeado no son válidos. Las causas son las mismas que en el caso anterior.

[14] Como ya se ha dicho antes, siempre que queramos modificar la configuración del sistema tendremos que trabajar como root.

```
prompt
#message = /boot/message
timeout=50
linear
append = "hda=scsi hdb=scsi hdc=scsi hdd=scsi hde=scsi hdf=scsi
          hdg=scsi hdh=scsi apm=power-off nomce"
#sección del sistema Linux
image=/boot/vmlinuz-2.4.20-xfs
  label=Guadalinex-Edu
  root=/dev/hda2
  read-only
#sección del sistema Windows
# other=/dev/hda1
#   label=dos
#   optional
#   table=/dev/hda
```

este fichero está dividido en dos partes, una parte común y luego dos secciones, una para cada sistema operativo de esta máquina. Además podemos ver que si queremos poner algún comentario se usa el mismo símbolo de siempre (#)

Sección Global

- `boot=/dev/hda` indica que LILO se carga desde el MBR del maestro del primer canal ide
- `vga=normal` especifica el modo de texto VGA que debe usarse al arrancar el sistema. Al poner `normal` seleccionamos el modo de texto 80×25.
- `prompt` hace que veamos LILO boot: y que podamos seleccionar el sistema operativo
- `message=/boot/message` fichero con el gráfico que se muestra antes del indicador de arranque.
- `timeout=`*`50`* tiempo (en décimas de segundo) que LILO debe esperar antes de comenzar a arrancar el sistema operativo que tenga definido por defecto.
- `linear` permite que las referencias a los sectores se escriban como direcciones lógicas en lugar de físicas. Se emplea cuando LILO no reconoce correctamente la geometría del disco duro (debido a un remapeado por parte del BIOS).
- `default=Guadalinex-EDU` etiqueta del sistema a cargar por defecto, en este ejemplo si queremos comenzar con el otro sistema tendríamos que poner `dos`
- `append=`*`parámetro`* permite pasar parámetros y componentes de hardware al kernel como cadena de caracteres. Por ejemplo, puede que para que Linux reconozca dos tarjetas de red haya que pasarle aquí los parámetros adecuados para que reconozca la segunda tarjeta.

Sección linux

- `image=/boot/vmlinuz-2.4.20-xfs` ruta completa del fichero que contiene el kernel de Linux.
- `label=Guadalinex-EDU` etiqueta de este sistema, podemos poner lo que nos plazca pero limitado a cadenas de 15 caracteres.
- `read-only` indica al kernel que monte inicialmente la partición raíz en modo de sólo lectura.
- `root=/dev/hda2` nombre de la partición linux nativa, en este caso, es la primera partición del segundo disco duro.

Sección otros ...

Si sólo tenemos Linux esta sección no aparecería. Existen más opciones para esta sección además de las aquí listadas.

- `other= /dev/hda1` donde indicamos esta vez la partición donde está cargado el otro sistema operativo.
- `label=dos`

Otras (solo algunas) que no aparecen en el fichero de ejemplo

- `lba32` es "incompatible" con `linear`. Con esta línea "puede[15]" que podamos trabajar con LILO con discos grandes (más de 8.4 GB) en los que hayamos puesto el kernel en cualquier partición independientemente del cilindro de inicio.
- `password=`*`contraseña`* contraseña que permite cargar LILO.

Si modificamos el fichero `/etc/lilo.conf` es necesario ejecutar:

```
# lilo
```
[16]

para que lea los cambios del fichero y actualice lo que allí se le indica. La etiqueta marcada con un asterisco será la correspondiente al sistema operativo de arranque por defecto.

Si queremos desinstalar LILO de nuestra máquina tenemos varias opciones:

- desde Windows ejecutar

  ```
  fdisk /MBR
  ```
- desde Linux ejecutar

  ```
  # lilo -u
  ```

  y restauramos el MBR anterior a la instalación de LILO

El comando `lilo` admite bastantes opciones que se pueden consultar en la página man del programa.

1.2. Fichero /etc/inittab

Comentábamos en la sección anterior que el GRUB/LILO pasa el control de la máquina al núcleo y comienza la autodetección del hardware de nuestra máquina y carga los drivers de dispositivos. Toda la información que se genera la tenemos a nuestra disposición cada vez que arrancamos Linux, va apareciendo en el monitor de nuestro equipo.[17]

[15] Tanto la BIOS como el disco han de admitir transferencias de bloque EDD.

[16] `lilo` se puede ejecutar con las opciones `-t` (*test*, prueba) o `-q` (*query*, consulta) para mostrar lo que LILO haría durante un inicio real.

[17] Si después queremos acceder a esa salida por pantalla podemos usar la orden

```
$ dmesg
```

Para poder verla mejor, es usual usar (para salir `q`)

```
$ dmesg | less
```

```
Initializing Cryptographic API
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 4096 buckets, 32Kbytes
TCP: Hash tables configured (established 32768 bind 32768)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
kjournald starting.  Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Freeing unused kernel memory: 140k freed
INIT: version 2.85 booting
Starting Bootlog daemon: bootlogd.
Loading /etc/console/boottime.kmap.gz
Activating swap.
Adding Swap: 305192k swap-space (priority -1)
Adding Swap: 262136k swap-space (priority -2)
Checking root file system...
fsck 1.35-WIP (21-Aug-2003)
/dev/hda2: clean, 92667/486720 files, 495688/971932 blocks
EXT3 FS 2.4-0.9.19, 19 August 2002 on ide0(3,2), internal journal
Real Time Clock Driver v1.10e
System time was Sat Mar  6 15:01:56 UTC 2004.
Setting the System Clock using the Hardware Clock as reference...
System Clock set. System local time is now Sat Mar  6 15:01:57 UTC 2004.
Calculating module dependencies... _
```

Si la cosa va bien comienza el proceso *init*: es el padre de todos los demás procesos, su finalidad es la de generar los procesos nucleares que necesita el sistema. Para iniciar el resto del sistema, *init* utiliza el fichero de configuración[18] `/etc/inittab`[19]. Mediante este fichero podemos ajustar el inicio del sistema para que se ejecute según nuestros intereses.

Un ejemplo de fichero `/etc/inittab`[20] es:

```
# /etc/inittab: init(8) configuration.
# $Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $
# The default runlevel.
id:2:initdefault:
# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS
# What to do in single-user mode.
~~:S:wait:/sbin/sulogin
# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
# Normally not reached, but fallthrough in case of emergency.
z6:6:respawn:/sbin/sulogin
# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
# Action on special keypress (ALT-UpArrow).
#kb::kbrequest:/bin/echo "Keyboard Request--edit /etc/inittab to let this work."
# What to do when the power fails/returns.
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
```

[18] Se trata de un fichero de configuración y no de un *script* (fichero en el que hay un conjunto de órdenes del sistema operativo que se van ejecutando secuencialmente, de igual manera que si las vamos escribiendo una detrás de otra desde la línea de comandos, un script podemos entenderlo como un fichero BAT del "antiguo" MSDOS), es decir, el orden en el que se escriben las líneas no es significativo.

En primer lugar *init* ejecuta el script `/etc/init.d/rcS`. Este *script*, irá llamando a los ficheros necesarios para configurar las variables de entorno, iniciar los sistemas de ficheros, ...

[19] Quien quiera profundizar sobre el tema puede consultar la página man del fichero `inittab`

[20] Como siempre los # indican líneas de comentarios

```
# /sbin/getty invocations for the runlevels.
#
# The "id" field MUST be the same as the last
# characters of the device (after "tty").
#
# Format:
# <id>:<runlevels>:<action>:<process>
#
# Note that on most Debian systems tty7 is used by the X Window System,
# so if you want to add more getty's go ahead but skip tty7 if you run X.
#
1:2345:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6
# Example how to put a getty on a serial line (for a terminal)
#
#T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
#T1:23:respawn:/sbin/getty -L ttyS1 9600 vt100
# Example how to put a getty on a modem line.
#
#T3:23:respawn:/sbin/mgetty -x0 -s 57600 ttyS3
```

Las entradas de este fichero son de la forma:

```
etiqueta:niveldeejecución:acción:comando
```

donde:

etiqueta etiqueta única de un máximo de 4 caracteres

niveldeejecución lista de niveles de ejecución (se ve después) en los que se ejecuta el proceso

acción explicaremos sólo las más usuales, para el resto de acciones ver los documentos de ampliación ya comentados

ctrlaltdel comando a ejecutar si se pulsa la combinación de teclas[21] **[CTRL]+[ALT]+[DEL]**

initdefault nivel de ejecución por defecto

once el comando se ejecuta sólo una vez después de entrar en el nivel de ejecución seleccionado

powerfail comando a ejecutar si se recibe una señal SIGPWR (indica algún problema con la alimentación eléctrica)

powerokwait se ejecuta el comando si el SAI manda una señal indicando que se ha restablecido el nivel de energía eléctrica.

respawn se inicia una nueva instancia del proceso cada vez que termine

sysinit el *script* se ejecuta en la fase de arranque del sistema, independientemente del nivel de ejecución[22].

wait se ejecuta el comando una sola vez al iniciar el nivel de ejecución. Pero hasta que no termina el comando no se hace nada más.

comando comando a ejecutar

[21] Merece la pena fijarse en la línea:
`ca::ctrlaltdel:/sbin/shutdown -t1 -a -r now`
si la mantenemos permitimos que cualquier usuario pueda hacer un *reset* con nuestra máquina con sólo pulsar **[CTRL]+[ALT]+[DEL]**.

[22] `/etc/init.d/rcS` ⟶ A "grosso modo" podemos decir que se encarga de configurar/cargar todo el sistema (los OK del inicio).

Hay que tener mucho cuidado a la hora de modificar este fichero ya que nos arriesgamos a que Linux no pueda arrancar.

No vamos a entrar en la modificación de todos los aspectos que gestiona este fichero, tan sólo vamos a comentar cómo modificar los niveles de arranque.

1.2.1. Niveles de arranque

Al inicio de este fichero aparecen los distintos niveles de arranque de los que disponemos. Cada nivel de arranque permite configurar el sistema de forma diferente. Los niveles de arranque que tenemos a nuestro alcance son 6[23]. El objetivo de cada uno de estos niveles es:

N	carga del sistema (NONE)
S	modo Monousuario (para no ser cambiado directamente)
0	(*halt*) para apagar el sistema
1	modo monousuario (equivale a `s` →*single mode*)
2..5	modo multiusuario
6	(*reboot*) para reiniciar el sistema.

Orden de ejecución de los scripts[24] en `/etc/rc?.d`

- Todos los scripts ejecutados (o no) por el sistema `init` se encuentran en `/etc/init.d`. Los 8 directorios `/etc/rc?.d` (`?= S,0...6`) contienen enlaces simbólicos a estos scripts[25]. Estos enlaces se denominan `Sxxservicio` o `Kxxservicio`. En ambos casos, `xx` es un número de dos cifras y `servicio` es el nombre del script asociado.

  Inicialmente se ejecuta el script `/etc/init.d/rcS`, él se encarga de que se ejecuten todos los scripts `S*` de `/etc/rcS.d`. Los scripts que tengan los números más pequeños se ejecutarán primero[26].

- Después de que se han ejecutado los scripts `/etc/rcS.d`, `init` cambia al nivel de ejecución especificado en `/etc/inittab` (normalmente 2). Para eso, `init` ejecuta el script `/etc/init.d/rc` que se ocupa de arrancar los servicios en `/etc/rc?.d`, donde `?` representa el nivel de ejecución pasado como argumento.

  ↬ Por ejemplo, si el nivel de arranque es el 2, `init` ejecuta

  `/etc/rc.d/rc 2`

  Este script ejecuta entonces los ficheros que se encuentran en el subdirectorio `/etc/rc2.d`. Los ficheros de este subdirectorio[27] comienzan por S (*start*) o K (*kill*), seguidos de dos dígitos y la función que realizan. Se comienzan ejecutando los que comienzan por K y después los que comienzan por S. Los K* sirven para parar subsistemas (servicios) y los S* para arrancarlos.

  El número de dos dígitos a continuación de la K o de la S indica el orden en que se ejecutará el script. Los scripts que tengan los números más pequeños se ejecutarán primero.

- Si deseamos personalizar los servicios en el arranque, es preferible hacerlo usando los ficheros contenidos en el subdirectorio `/etc/default` y no con los script del directorio `/etc/init.d`.

[23] Debian utiliza el sistema de scripts **sys-V**.

[24] Aunque ya se ha aclarado antes su importancia obliga a esta nueva nota.

Script fichero en el que hay un conjunto de órdenes del sistema operativo que se van ejecutando secuencialmente, de igual manera que si las vamos escribiendo una detrás de otra desde la línea de comandos. Esto se llama *ejecución interpretada* de programas ya que cada línea se interpreta antes de su ejecución.

[25] En realidad en el subdirectorio lo único que hay son enlaces simbólicos a otros ficheros (podemos entender un enlace simbólico como un acceso directo de Windows, volveremos sobre ellos en un capítulo posterior) que están en `/etc/init.d`.

[26] Solo deberían arrancarse ahora las cosas que necesitan ejecutarse una vez para tener el sistema en un estado consistente. No deberíamos arrancar demonios en este nivel. El nivel de ejecución en este momento es "*N*" (*none*).

[27] Lo mismo que todos los de la forma `/etc/rc`**n**`.d` donde **n** es un número comprendido entre 0 y 6.

Cómo cambiar de nivel de ejecución

- La forma más sencilla es usando el comando[28]:

  `#init [n]`

  donde `n` es un número que indica el nivel de arranque que deseamos "activar". Cambiando niveles de ejecución:

  ↬ Si, por ejemplo, cambiamos desde el nivel 2 al nivel 5, se ejecutarán primero para el nivel de ejecución los scripts K y después todos los scripts S (ordenados de menor a mayor). Para optimizar el proceso, primero se revisa si un determinado servicio del nivel 5 se está ejecutando en el nivel de ejecución 2. Si ya lo estaba, y no hay ningún script K (stop) para él en el nivel 5 no se inicia de nuevo.

  ➥ **Para practicar**

  1. Niveles de arranque
     a) Accede a la cuenta del root
     b) Modificar el fichero `/etc/inittab` de vuestro sistema para que arranque para un nivel 3
     c) Reinicia el sistema y comprueba que es así.
     d) Cambia a nivel 1 ejecutando

     ```
     #init 1
     ```

     Para que podamos acceder así es necesario que introduzcamos la *password* del root cuando se nos pida. Deshagamos el entuerto escribiendo

     ```
     #telinit 2
     ```

     e) Dejar otra vez el fichero `/etc/inittab` como estaba al principio (nivel de ejecución 2)
     f) Reiniciar o apagar el sistema escribiendo

     ```
     init 6
     ```

     ó

     ```
     init 0
     ```

- **Sólo por si tenemos LILO.**

  Para modificar de un nivel de arranque a otro tenemos como primera opción ejecutar

  `linux número_nivel_arranque`[29]

  cuando aparece

  `LILO boot:`

  al inicio del sistema. Por ejemplo, si nosotros siempre entramos en modo gráfico y deseamos hacerlo en modo texto escribiríamos[30]:

  `LILO boot: linux 3`

[28]
- Se consigue lo mismo con el comando `telinit`.
- Podemos cambiar el nivel de ejecución en el arranque usando GRUB. Para eso, cuando aparezca el menú principal de GRUB pulsa la tecla **[e]** y en la ventana de edición escribe el número del nivel de ejecución (`[n]`) deseado,

  ```
  kernel /boot/vmlinuz-2.4.22 [n] root=....
  ```

  se escribe sólo el número tras el fichero que contiene al núcleo.

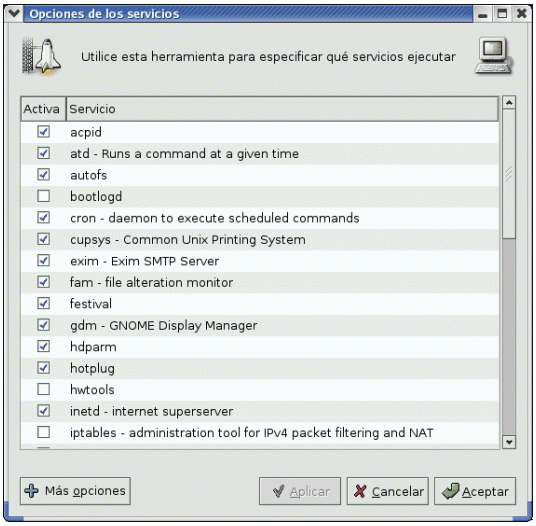
[29] En el supuesto de que la etiqueta de nuestro sistema Linux sea `linux`

[30] Merece la pena aclarar que podemos pasar de un nivel a otro ejecutando:
`# init "nuevo_nivel"`
Así, si estamos en nivel 3 y deseamos pasar el 5 escribiríamos:
`# init 5`

1.2.2. Control de acceso a servicios

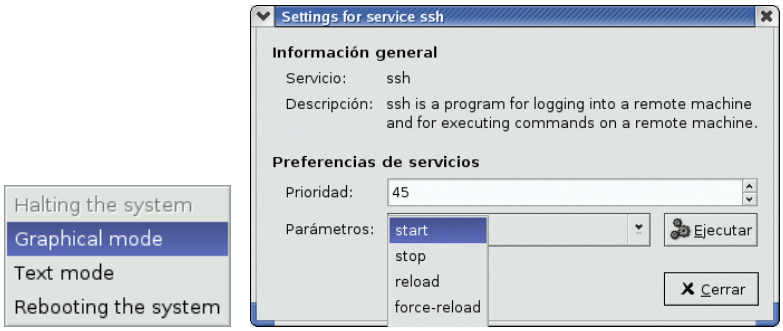
Vamos a enumerar las herramientas de que disponemos en Guadalinex para poder activar/desactivar determinados servicios según los distintos niveles de ejecución, se trata de[31]:

 # services-admin (desde Gnome:  →**Herramientas del sistema**→**Panel de control** → **Servicios**), es una utilidad gráfica que permite seleccionar qué servicios están en activo.



Si optamos por marcar Más Opciones, accederemos a funcionalidades añadidas:

- Seleccionar el nivel de arranque para el que realizamos los cambios
- Obtener más información sobre el servicio, establecer la prioridad y parámetros a pasarle.



 # /usr/sbin/update-rc.d utilidad en línea de comandos para activar/desactivar servicios. En general es más fácil trabajar con la anterior.

Usando este comando podemos configurar los enlaces simbólicos de los directorios `/etc/rc?.d` y el script situado en `/etc/init.d/`. Si por ejemplo deseamos que el servicio de nombre `service` se ejecute en el arranque

1. Se pone en el directorio `/etc/init.d/`. En general los programas que instalemos y que sean necesarios en el arranque situan sus scripts de forma automática aquí.

[31] El comando a ejecutar desde una xtermes el que aparece en negrita

2. Después creamos los enlaces simbólicos[32] mediante el comando

   ```
   update-rc.d servicio defaults 35
   ```

   Al pasarle el parámetro `defaults` forzamos a que lo cree para los niveles de ejecución que van del 2 al 5. Con el `35` obligamos a que service se arranque antes de cualquier script que contenga un número mayor de 36.

Además de los programas anteriores, si tenemos un servicio en nuestra máquina, con:

```
$ /etc/init.d/service
```

podemos comprobar qué parámetros admite. Por ejemplo, con el servidor de impresión obtendríamos[33]:

```
$ /etc/init.d/cupsys
Uso: /etc/rc.d/init.d/lpd {start|stop|restart|force-reload}
```

start arrancar **reload** volver a cargar **force-reload** forzar carga

stop detener **restart** volver a arrancar

O sea que si queremos pararlo sólo hay que ejecutar[34]:

```
# /etc/init.d/cupsys stop
Parando cupsys: cupsd
```

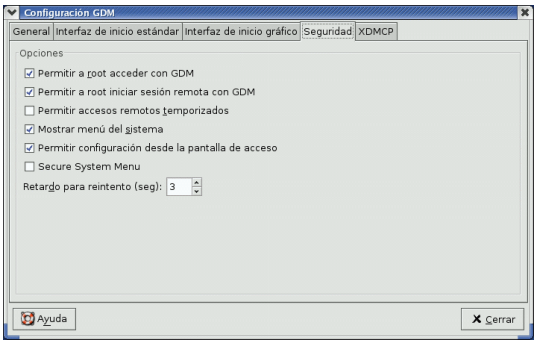
1.2.3. Gdm

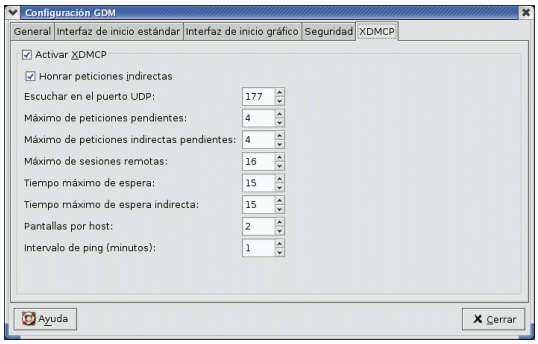
Si accedemos a nuestra máquina en modo gráfico, hemos estado utilizando el programa **gdm** (*GNOME Display Manager*):

La configuración de `gdm` se consigue pulsando sobre  →**Herramientas del sistema**→ **Panel de Control** → **Configuración de GDM**.

[32] Estos enlaces también se pueden crear a mano, pero eso mejor lo dejamos para cursos más avanzados.
[33] En general, estos parámetros funcionan para todos los scripts situados en `/etc/init.d/`.
[34] También podemos usar:

```
#invoke-rc.d cupsys stop
```





Con esta utilidad podemos personalizar `gdm` tanto para conexiones locales como remotas:

- Con la pestaña **Seguridad** y marcando la casilla **Permitir a root acceder con GDM** permitimos que el root inicie una sesión gráfica en el sistema
- Por defecto las conexiones remotas están desactivadas, si deseamos permitirlas hay que marcar la casilla **Activar XDMCP** en la pestaña **XDMCP** e iniciar el sistema en modo gráfico.
- Los cambios efectuados se almacenan en el fichero[35] `/etc/X11/gdm/gdm.conf`

➥ **Para practicar:**

Modificar la configuración de GDM para que permita el acceso como root[36].

[35] Es un enlace al directorio `/etc/gdm`
[36] Puede que ya esté hecho desde la 1ª entrega

Capítulo 2

Sistemas de ficheros en Linux

> Toda entidad física y lógica en Linux se representa como un archivo en el sistema de archivos de Linux. Las entidades físicas incluyen discos, impresoras y terminales; las entidades lógicas incluyen directorios y archivos normales, en los que se almacenan documentos y programas. *(Linux*, JACK TACKETT JR. Y DAVID GUNTER)

2.1. Introducción.

Revisemos los conceptos de formateo de un disco y de sistema de ficheros. Lo que denominamos formatear (por ejemplo, un floppy) comprende en realidad dos procesos: el formateo de bajo nivel y la creación del sistema de ficheros.

Formatear es el proceso de *escribir* marcas en el medio magnético de un disco para distinguir las pistas y sectores, que posteriormente pueden ser accesibles por su localización. Para los discos duros suele venir ya de fábrica. Sobre ese disco ya formateado se pueden establecer particiones.

Crear un *sistema de ficheros* consiste en generar las estructuras de datos que un sistema operativo (como Linux o Windows) utiliza para contener los ficheros y directorios que usa. Se crean sobre las particiones que hayamos designado para ese sistema operativo. Puede haber, por tanto, varios sistemas de ficheros en un mismo disco (en varias particiones o secciones de él). También puede ocurrir, al menos en sistemas UNIX más recientes, que un sistema de ficheros esté montado sobre varias unidades de disco, dando lugar a lo que se llama sistemas de ficheros multivolumen[1].

En Linux trabajamos básicamente con cuatro tipos de ficheros:

1. Ficheros normales.
2. Directorios.
3. Enlaces.
4. Archivos especiales[2].

Los dos primeros son de sobra conocidos y del tercero daremos algunas pinceladas en esta sección.

Un par de anotaciones antes de seguir:

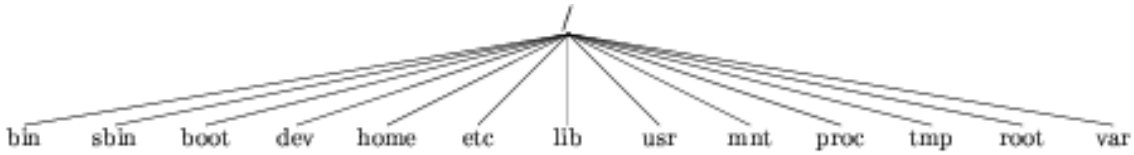
- Linux es “casesensitive”, o sea que el fichero `curso.txt` es distinto del fichero `Curso.txt` y lo mismo con los directorios.

[1] Para ampliar sobre discos duros, particiones y sistemas de ficheros visita la página `http://www.saulo.net/pub/ddypart`.

[2] Aquí englobaríamos a los ficheros de dispositivos que se encuentran en el subdirectorio `/dev` y otros ficheros especiales: *Fifo* o tuberías con nombre.

- Los nombres de fichero pueden contener los caracteres a los que estamos habituados, con un máximo de 255.

Todos los ficheros dentro del sistema de ficheros se organizan siguiendo una estructura en forma de árbol invertido, en la que el nodo superior se denomina nodo raíz.



La mayoría de los sistemas de ficheros UNIX tienen una estructura parecida, lo que varía de unos a otros son los detalles de la implementación, lo que los hará más o menos eficientes.

2.2. Tipos de Sistemas de Ficheros.

Linux soporta, además de los tipos de sistemas de ficheros nativos (Minix, Ext, Xia, Ext2 y Ext3), varios sistemas de ficheros ajenos para facilitar el intercambio de datos con otros sistemas operativos. Algunos de los sistemas de ficheros soportados por Linux son:

ext2 Sistema de archivos estándar de Linux.

ext3 Sistema de ficheros que usa esta versión de Guadalinex. Es una mejora del anterior con la ventaja de que es de tipo *journaling* (se reduce el tiempo de recuperación tras un apagado inesperado)[3].

msdos Para la compatibilidad con el sistema de ficheros FAT del MS-DOS.

vfat Para compatibilidad con sistemas Windows 9x (Fat32). Soporta nombres largos de ficheros.

ntfs Acrónimo de *new technology file system*, sistema de ficheros de Windows 2000/XP.

iso9660 Es el tipo de sistema de ficheros estándar para CDROM; la extensión Rock Ridge al CD-ROM estándar, que permite nombres de fichero más largos, está soportada de forma automática.

umsdos Extiende el sistema de ficheros msdos bajo Linux, de forma que desde Linux se pueden usar nombres de fichero largos, propiedad, permisos, enlaces y ficheros de dispositivo. Esto permite que un sistema de ficheros msdos se use como si fuera un sistema de ficheros Linux, sin necesidad de hacer una nueva partición para Linux. Tiene como contrapartida un rendimiento inferior a los sistemas de ficheros nativos.

hpfs Para la compatibilidad con el sistema de ficheros de OS/2.

nfs Es un sistema de ficheros de red que permite compartir sistemas de ficheros entre varios ordenadores.

sysv Para compatibilidad con UNIX SystemV/386, Coherent y Xenix.

minix Primer sistema de ficheros utilizado para Linux

ext Primer sucesor de minix (está en desuso).

[3] **Características**:

- Máximo tamaño de bloque 4Kb; tamaño máximo del sistema de ficheros 16.384 GB; tamaño máximo de fichero 2048 GB.
- Para pasar un sistema de ficheros **ext2** a uno **ext3** y viceversa disponemos del comando `/sbin/tune2fs`. Para saber cómo hacerlo, además de la *manpage* del programa mirar la página `http://www.debianitas.net/docbook/ext3-v1.0/ext3-v.1.0.html`.

xiafs Sucesor del ext (tampoco se usa)

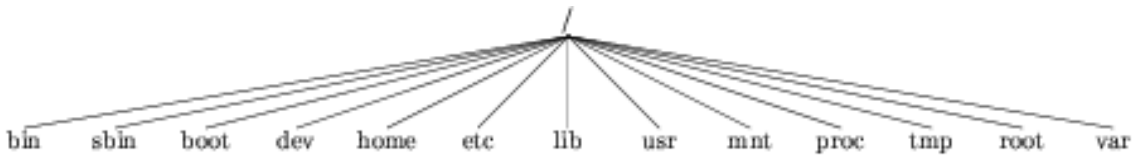
proc Sistema de archivos virtual de Linux.[4]

Existe además, generalmente, una partición o sección en el disco usada por el gestor de memoria conocida como área de **swap** (intercambio). A través ella el gestor de memoria implementa la memoria virtual. Dependiendo de los sistemas, este área puede ser tratada bien como un sistema de ficheros más, o bien directamente a través del dispositivo en bruto (*raw disk*) .

2.3. Organización de los directorios

El **FHS** (*Filesystem Hierarchy Standar* `http://www.pathname.com/fhs/`) o Estándar de Jerarquía del Sistema de Ficheros, es un documento en el que se definen los nombres y situación de muchos ficheros y directorios.

En Linux, "todo son ficheros[5]", para organizarlos tenemos una estructura de árbol de la que cuelgan todos los ficheros. El tronco principal es /, el directorio raíz.



De él cuelgan normalmente los directorios[6]:

/bin Contiene los comandos básicos del sistema operativo[7]

/sbin Comandos esenciales para la administración del sistema[8].

/boot Aquí están los binarios de arranque del sistema.

/dev Todos los ficheros de dispositivos.

/etc Ficheros de configuración del sistema.

/home En él tendremos los directorios de trabajo de los usuarios.

/root Directorio de trabajo del "jefe".

/lib Librerías básicas para trabajar en Linux.

/mnt De él cuelgan normalmente los sistemas de archivos de dispositivos extraibles, floppy, CDROM, ... Esto no es obligatorio y Guadalinex se configura montándolos directamente en /floppy, /cdrom, /cdrom1, ...

/proc Ya se ha comentado anteriormente, este directorio no existe en realidad (físicamente). En él podemos encontrar información sobre el sistema.

[4]No es en realidad un sistema de ficheros, aunque lo parece. Permite acceder a ciertas estructuras de datos del Kernel como, por ejemplo, listas de procesos (de aquí su nombre). Organiza estas estructuras como un sistema de ficheros de modo que se pueda acceder a ellas con las herramientas habituales de acceso a ficheros.

Por ejemplo, para ver los procesos que están ejecutándose en el sistema usaremos el comando:

```
$ ls -l /proc
```

O para ver los distintos tipos de sistemas de ficheros soportados por el Kernel que tenemos arrancado:

```
$ cat /proc/filesystems
```

Aunque a veces se le llame sistema de ficheros, `/proc` no está montado sobre el disco, sino que sólo existe en el kernel. Cuando intentamos acceder a alguna parte de `/proc`, el Kernel crea la ilusión de que existe en el disco aunque no es así. Si miráis el contenido de los ficheros de ese "directorio" podéis ver qué procesos están en ejecución e información de toda la máquina.

[5]Incluso los directorios son ficheros cuyos datos son los archivos que contienen. Y los dispositivos son ficheros especiales de los que se puede leer y en los que se puede escribir.

[6]Guadalinex y otras distribuciones crean además los directorios: `initrd` (si se borra no podremos acceder al sistema), `misc` , `opt, auto` y `lost+found` (zona para poner los archivos "extraviados").

[7]En `/usr/bin` también hay comandos del sistema pero son "menos básicos" que los que hay en `/bin.`

Se debe a que al iniciar el sistema, puede que todavía la estructura de `/usr` no esté disponible porque esté sobre otro sistema de ficheros. Pero `/bin` siempre lo tendremos a mano en caso de que haya problemas.

[8]En `/usr/sbin` y `/usr/local/sbin` también hay comandos de administración del sistema. De igual manera que antes son los comandos de "2ª" para esta labor.

/tmp En él se almacena información temporal.

/usr Aquí estarán los programas que no forman parte del sistema más básico.

/var ficheros de datos variables: logs del sistema, datos administrativos, etc.

2.3.1. Ficheros de configuración del sistema

Somos conscientes de que lo que sigue es un tanto árido, pero nos intriga saber dónde se configura "tal" comportamiento del sistema. Estos ficheros se almecenan en distintos lugares, fundamentalmente en:

$HOME en el directorio de trabajo de cada usuario hay una serie de ficheros[9] que rigen el comportamiento de sus cuentas de registro.

/etc En este directorio están la mayoría de los ficheros de configuración del sistema[10]:

aliases puede contener listas de distribución utilizadas por el servicio de correo de Linux.

bash.bashrc valores por defecto en todo el sistema para la shell bash.

csh.cshrc en él se establecen los valores por defecto de la shell csh (shell C).

exports lista de los directorios locales para ser compartidos utilizando NFS (Network File System).

fstab contiene información que describe los diversos sistemas de ficheros y las ubicaciones del sistema Linux en las que se montan, se estudia en 2.5.1 en la página 27.

group en él se almacenan los nombres de grupo e ID de grupo (GID) definidos en el sistema.

host.conf fichero de configuración del "resolvedor" de nombres. En él se establecen localizaciones en las que se buscan los nombres de dominio (por ejemplo, redhat.com) en las redes TCP/IP (como por ejemplo, Internet).

hosts lista de direcciones IP y nombres de máquinas.

hosts.allow ordenadores que están autorizados para utilizar servicios TCP/IP desde el ordenador local.

hosts.deny ordenadores que no están autorizados a utilizar servicios TCP/IP desde el ordenador local.

inittab mediante este fichero podemos ajustar el inicio del sistema para que se ejecute según nuestros intereses

/etc/inetd.conf es el fichero de configuración para el demonio servidor `inetd`[11]

ld.so.conf lista de directorios que contienen librerías del sistema.

mail.rc parámetros del sistema asociados con el uso del correo.

manpath.config fichero de configuración del comando `man`.

modules.conf en él se establecen los alias y opciones para los módulos cargables de kernel.

mtab lista de sistemas de archivos montados en ese momento.

passwd cada línea de este fichero guarda información relativa a cada uno de los usuarios del sistema.

printcap archivo de configuración de las impresoras configuradas en nuestro equipo.

profile fichero que configura el entorno de trabajo, tiene validez para todo el sistema, programas de entorno e inicio de todos usuarios.

protocols en él se describen los distintos protocolos para Internet que están disponibles en el subsistema TCP/IP.

resolv.conf direcciones IP de los servidores de nombres.

rpc define nombres y números para los procedimientos remotos de llamada.

services lista de servicios de red de Internet

shadow lista de contraseñas encriptadas para los usuarios que aparecen en el fichero `passwd`.

shells lista las rutas de los intérpretes de línea de comando con los que contamos en nuestro sistema.

sudoers fichero de configuración del comando `sudo`[12].

syslog.conf en él se define qué mensajes de inicio de sesión recoge el demonio `syslogd` y qué ficheros se almacenan en él.

[9] La mayoría empiezan su nombre con un punto

[10] Algunos puede que no estén (aún) en vuestro sistema y desde luego están otros que no enumeramos.

[11] Es un proceso especial que se queda a la escucha de conexiones TCP en unos puertos determinados. Cuando viene una solicitud de conexión, realiza una serie de comprobaciones y ejecuta el proceso servidor correspondiente.

[12] Con él podemos conseguir de una forma segura que determinados comandos restringidos al root puedan ser ejecutados por otros usuarios.

/etc/X11 directorio de configuración del sistema gráfico X y de los diferentes gestores de ventanas.

/etc/apt contiene, entre otros, el fichero `sources.list` con la lista de repositorios que utilizará APT para la instalación de paquetes.

/etc/cron.* distintos directorios con ficheros en los que se define la forma en que `crond` ejecuta las aplicaciones programadas con pautas diarias, horarias, mensuales o semanales.

/etc/default en este subdirectorio hay una serie de ficheros que establecen valores para diversas utilidades. Véase el fichero `useradd`, en él se definen los valores por defecto para el número de grupo, el directorio de inicio, la fecha de caducidad de la contraseña, el shell y el directorio esqueleto (`/etc/skel`) que se utiliza cuando se crea un nuevo usuario del sistema.

/etc/init.d contiene las copias permanentes de los guiones de nivel de ejecución. Estos guiones están vinculados a ficheros en los directorios `/etc/rc?.d` para que cada servicio se asocie con un guión iniciado o detenido en el nivel de ejecución en cuestión.

/etc/ppp ficheros de configuración de la conexión a internet usando un módem/RDSI.

/etc/rc?.d directorios que determinan los procesos a ejecutar en los diferentes niveles de ejecución.

/etc/security aquí se establecen una serie de condiciones de seguridad por defecto para nuestro ordenador.

/etc/skel directorio “esqueleto” para crear los directorios de usuario de las nuevas cuentas del sistema. Los ficheros de este directorio se copian al directorio de trabajo (`/home/usuario`) del usuario (la mayoría de estos ficheros son ficheros que comienzan por un punto).

2.3.2. Logs del sistema

Si en algo sobresale Linux es en la posibilidad de mantener multitud de ficheros en los que se almacenan todas las acciones que realiza el sistema (ficheros de registro). Esto se consigue gracias a los demonios:

syslogd monitoriza el registro general del sistema

klogd registro específico de la actividad del kernel

El fichero de configuración de los logs del sistema se lleva a cabo a partir del fichero `/etc/syslog.conf`, y los ficheros en donde se almecenan los resultados están en el directorio: `/var/log`. Algunos de ellos son:

mail.* mensajes generados por el demonio `sendmail`[13] .

dmesg mensajes de arranque del kernel.

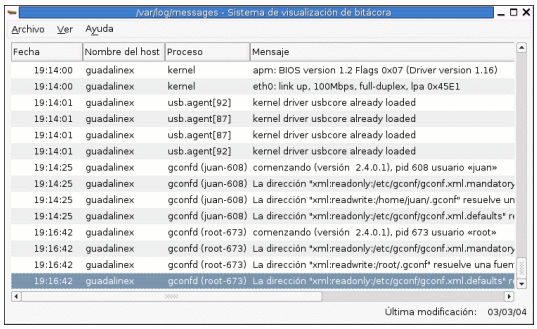
messages “cajón de sastre” de los mensajes del sistema.

XFree86.* guarda información sobre lo que pasa con el entorno gráfico (servidor X).

...

Podemos accede a los logs del sitema tecleando en un terminal `gnome-system-log` , o en modo gráfico con la siguiente secuencia de menús:  **Aplicaciones**→**Herramientas del sistema**→**Bitácora del sistema** .

[13] MTA: agente para el transporte de correo electrónico



Archivo→Abrir y elegir fichero de logs deseado.

➥ **Para practicar**

1. Mostrar por pantalla las últimas 10 líneas del fichero `/var/log/messages`
   `#tail -f /var/log/messages`
2. Solicitar la página man de `dmesg` y luego visualizar el fichero `/var/log/dmesg`
   a) Usando un editor de texto (`gedit` por ejemplo)
   b) Usando una tubería (`|`) y algo de comandos:
   `$dmesg | less`
   para salir hay que pulsar `q`
3. Utilizar el bitácora del sistema para curiosear por los logs.

2.4. Creación de un Sistema de Ficheros.

Como ya hemos comentado anteriormente, los sistemas de ficheros se crean sobre las particiones del disco duro. Una vez instalado el sistema, disponemos de varias utilidades para poder trabajar sobre las particiones. Si lo que deseamos es ver la tabla de particiones de un disco, añadir, borrar o cambiar el tipo de sistema de ficheros podemos usar `fdisk`. Pero si además de todo eso deseamos modificar el tamaño de una partición tendremos que usar `qtparted`. La sintaxis básica de uso es similar en los dos y consiste en ejecutar el programa (como root) pasándole como argumento la unidad de disco con la que se va a trabajar (`/dev/hdx`)[14].

Los sistemas de ficheros se crean con el comando **`mkfs`** (*Make Filesystem*). La sintaxis[15] de este comando es:

```
# mkfs [-t tipo_sf] sistemaficheros
```

Donde:

tipo_sf es el argumento mediante el que se pasa el tipo de sistema de ficheros a crear (ext3, ext2, hfs, Minix, etc). Si se omite este argumento, `mkfs` lo deducirá buscando en el fichero `/etc/fstab.` Si se omite y no hay entrada en `fstab` tomará por defecto **ext2**.

[14] # `fdisk -l /dev/hda` muestra la tabla de particiones del disco duro hda.
`fdisk /dev/hda` solicita teclear *m* para mostrar ayuda, ¡cuidado con las opciones!
`qtparted /dev/hdb` muestra la tabla de particiones con la posibilidad de redimensionarlas. Ya hicimos uso del programa en el proceso de instalación.

[15] No es la sintaxis “completa”, para ampliar sobre este comando mirar en las páginas man del comando.
Por si no lo hemos comentado antes, la sintaxis de los comandos se expresa como una expresión regular: opciones o argumentos entre corchetes significa que pueden o no aparecer (`[]`), la barra vertical (`|`) significa que puede aparecer un elemento u otro pero uno de ellos obligatoriamente ...

sistemaficheros es el único argumento obligatorio, y corresponde al dispositivo o a la partición del disco sobre la cual queremos crear el sistema de ficheros (`/dev/hda1`, `/dev/sda3`, `/dev/fd0`, etc), o también podría pasársele el punto de montaje[16] (`/tmp`, `/users`, etc).

Por ejemplo, la sentencia,

```
# mkfs -t ext3 /dev/sda1
```

nos creará un sistema de ficheros tipo ext3, que ocupará todo el espacio de la partición `/dev/sda1` del disco. Una vez creado el sistema de ficheros, podemos almacenar datos en el.

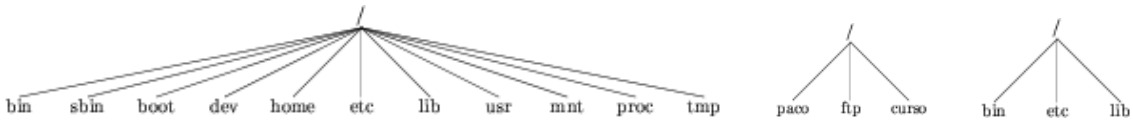
➥ **Para practicar**

1. Crear un sistema de ficheros de tipo `vfat` en un disquete (`/dev/fd0`), con la orden:
   `#mkfs -t vfat /dev/fd0`
2. Comprobar que podemos conseguir lo mismo usando el comando[17]

   `#mkfs.vfat /dev/fd0`

2.5. Montaje y Desmontaje.

El concepto de montaje tiene que ver con que en Unix/Linux todos los elementos son ficheros, incluidos los dispositivos. Para poder usar un sistema de ficheros éste tiene que estar *montado*. Para ello, cada nuevo sistema de ficheros se enlaza mediante la operación de montaje con otra estructura de directorios de la que "cuelga". El sistema de ficheros inicial se denomina el *sistema de ficheros raíz* y posee el símbolo /. Este sistema de ficheros se monta al arrancar el sistema y ocupa el lugar más alto[18]. Por ejemplo, supongamos que en tres particiones de nuestro disco (o discos diferentes) hemos creado tres sistemas de ficheros. Podemos verlos de la siguiente forma. Cada uno de ellos visto de forma independiente posee una raíz. Supongamos que son `/dev/hda1`, `/dev/hda2` y `/dev/hda3` respectivamente:

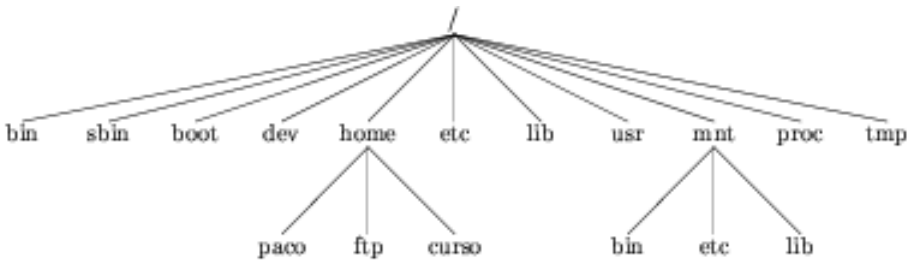


Cuando montamos un sistema de ficheros, indicamos un directorio del sistema de ficheros "padre", del que va a colgar la estructura de directorios del sistema de ficheros "hijo".

Por ejemplo, mediante las órdenes de montaje[19] siguientes:

```
# mount /dev/hda2 /home
# mount /dev/hda3 /mnt
```

obtenemos la siguiente estructura de directorios.



[16] En este caso, intentará encontrar la partición en la que crear el sistema de ficheros de la entrada en el fichero `/etc/fstab` (véase la página 27) donde encuentre el punto de montaje.

[17] De igual manera existen los scripts: `mkfs.ext2`, `mkfs.ext3`, ...

[18] En él se encuentra el "corazón" de nuestro sistema.

[19] La sintaxis la estudiaremos un poco más adelante.

Como vemos, hemos montado el sistema de ficheros que hay en `/dev/hda2` bajo el directorio `/home` del sistema de ficheros raíz. Y el sistema de ficheros de `/dev/hda3` bajo el directorio `/mnt`. Ahora ya prodríamos acceder a cualquier dirección de este árbol de directorios.

2.5.1. El fichero /etc/fstab

El fichero `/etc/fstab` contiene información descriptiva sobre los distintos sistemas de ficheros del sistema. Este fichero es de solo lectura y debe ser mantenido por el administrador del sistema. Cada sistema de ficheros ocupa una línea de este fichero, y los campos de cada línea están separados por tabuladores o espacios. El orden de los registros es importante ya que `fsck`, `mount` y `umount` actúan secuencialmente sobre este fichero.

device Es el primer campo y especifica el dispositivo de bloque o el sistema de ficheros remoto a montar.

directorio Es el segundo campo y especifica el punto de montaje para el sistema de ficheros. Para particiones de swap este campo debería estar a "none". El directorio destino de montaje tiene que existir en nuestro sistema antes de montar el sistema de ficheros.[20]

tipo Es el tercer campo y especifica el tipo de sistema de ficheros. Si el contenido de este campo es "ignore", el sistema de ficheros no se monta. Esto puede ser útil para mostrarnos las particiones de disco que están actualmente sin ser usadas.

opciones Es el cuarto campo y especifica las opciones de montaje asociadas al sistema de ficheros. Las opciones van separadas por comas. Algunas de las opciones posibles son:

`auto` → La partición se monta al arrancar
`noauto` → No se monta la partición en el arranque
`user` → Se permite a los usuarios montar la partición
`nouser` → Sólo el root puede montar esta partición.
`ro` → Partición de sólo lectura.
`rw` → Se permite la lectura y la escritura.
`exec`→ Se pueden ejecutar los binarios de esa partición.
`async` → El sistema sigue trabajando tras una petición de escritura del dispositivo, aunque todavía no haya recibido la confirmación.
`defaults` → equivale a: `rw, exec, auto, nouser, async.`

frecuencia Con él determinamos la frecuencia con que deben hacerse copias de seguridad del sistema por el comando dump. Si este campo no está presente se devuelve a *dump* el valor cero, lo que indica que el sistema de ficheros no necesita ser salvado.

secuencia Es el sexto campo (*fsckorder*) y es usado por `fsck` (se estudia en 2.6) para determinar el orden en que se realizan los chequeos de los sistemas de ficheros en tiempo de arranque. El sistema de ficheros raíz debería ser especificado con un 1, y los demás sistemas de ficheros deberían tener 2. Los sistemas de ficheros en un mismo disco deberían chequearse secuencialmente, pero los sistemas de ficheros en diferentes discos deberían de chequearse al mismo tiempo, para utilizar el paralelismo disponible en el hardware. Si el sexto campo no está presente o es cero, se devuelve el valor cero, y `fsck` asume que el sistema de ficheros no necesita ser chequeado.

Ejemplo de fichero `/etc/fstab`[21] :

```
# /etc/fstab: static file system information.
#
# The following is an example. Please see fstab(5) for further details.
# Please refer to mount(1) for a complete description of mount options.
```

[20] La orden para crear un subdirectorio es `mkdir`
[21] El vuestro no tiene por qué coincidir, pero hemos dejado las opciones que el sistema pone por defecto.

```
#
# Format:
# <file system> <mount point>  <type>  <options>         <dump><pass>
/dev/hda3     /               ext3   defaults,errors=remount-ro  0  1
/dev/hda6     none            swap     sw                  0  0
proc         /proc            proc     defaults              0  0
/swapfile    none             swap      sw                   0  0
none       /proc/bus/usb      usbdevfs  rw                   0  0
/dev/fd0   /floppy  vfat noauto,user,async,exec,rw,showexec,umask=022  0  0
/dev/hda6    swap       swap     defaults                        0  0
/dev/cdrom   /cdrom     iso9660 defaults,ro,user,noexec,noauto  0  0
/dev/cdrom1  /cdrom1    iso9660 defaults,ro,user,noexec,noauto  0  0
/dev/sda1    /mnt/USB1        vfat  defaults,rw,noauto,user     0  0
/dev/sdb1    /mnt/USB2        vfat  defaults,rw,noauto,user     0  0
/dev/hda2    /mnt/RedHat      auto  noauto,user,exec            0  0
/dev/hda3    /mnt/Linux2      auto  noauto,user,exec            0  0
/dev/hda5    /mnt/homeRedHat  auto  noauto,user,exec            0  0
/dev/hda1    /mnt/Windows9X1  auto  noauto,user,exec            0  0
```

2.5.2. El comando mount

El comando `mount` nos permite montar los sistemas de ficheros y anclarlos a puntos de montaje (directorios). A partir de este momento podremos acceder a ellos. La sintaxis más usual es:

```
mount [-t vfstype] [device] dir
```

Si montamos una unidad que está incluida en el fichero `/etc/fstab` se omiten los argumentos `[-t fstype]` y `[device]`, la información necesaria la obtiene el comando mount del fichero `/etc/fstab`. Por ejemplo, con el `/etc/fstab` anterior, para montar la unidad de CD escribiríamos

```
$ mount /cdrom
```
[22]

Si lo que queremos es montar un disquete formateado bajo Windows tendremos que escribir

```
$ mount /floppy
```

El mismo comando hubiese montado un disquete de tipo ext2, si en la entrada correspondiente a tipo de sistema de ficheros de la línea asociada a `/dev/fd0` hubiésemos puesto **auto**.

El comando `mount` soporta una gran variedad de opciones, muchas de ellas dependientes del tipo de sistema de ficheros a montar. Para una descripción detallada consultar las páginas del manual en línea :

```
man 8 mount
```

2.5.3. El comando umount

La sintaxis más usual es:

```
umount [opciones] dir
```

El comando `umount` elimina el anclaje entre el dispositivo especial y el punto de montaje.

Para desmontar la unidad de CD:

```
$ umount /cdrom
```

Si tenemos una unidad de CD montada no podremos extraer el CD hasta que no lo desmontemos. Si estamos trabajando con una unidad de disquete es fundamental desmontarla antes de extraer el disco. No olvidemos que el sistema es multitarea y puede haber operaciones sobre el dispositivo pendientes de realizarse aunque ya la hayamos ejecutado. Solo puede desmontar un dispositivo el usuario que lo montó o el root.

[22] Notad que la orden la puede ejecutar un usuario al tener como opción `user` el dispositivo `/dev/cdrom`

Para desmontar un sistema de ficheros debemos de estar situados fuera de él. Es decir, el directorio actual de la sesión desde la cual realizamos el desmontaje no debe de estar dentro del sistema de ficheros a desmontar. Es más, ningún proceso del sistema debe estar utilizando el sistema de ficheros que deseemos desmontar, en caso contrario nos diría que está ocupado. La solución será "matar" al proceso o procesos que están utilizando el sistema de ficheros que deseamos desmontar o esperar a que acaben "por las buenas".

Para saber qué procesos están utilizando un sistema de ficheros podemos utilizar la orden `fuser`. Por ejemplo, si en `/mnt/windows` tenemos montada nuestra partición con el sistema de ficheros **vfat** y no nos deja desmontarla, podemos ver de qué procesos se trata con:

```
$ /sbin/fuser -a /mnt/windows
```

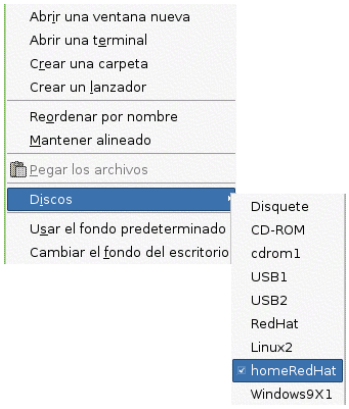
Con la orden:

```
# umount -a
```

serán desmontados todos los sistemas de ficheros contenidos en `/etc/mtab`. Con la opción `-t` le podemos decir a `umount` que desmonte solo los sistemas de ficheros de un tipo en concreto (o varios, separando los argumentos por comas).

2.5.4. Herramientas gráficas para montar dispositivos

El montaje y desmontaje de dispositivos puede hacerse en modo gráfico, para ello haga clic con el botón derecho del ratón en una zona libre del escritorio, seleccione **Discos** y elija el dispositivo que desee.



Para desmontarlo, repita la operación anterior o haga clic con el botón derecho del ratón sobre el icono del dispositivo y en el menú emergente elija **Desmontar el volumen**.

➥ Para practicar

1.
   a) Crear en un disquete un sistema de ficheros de tipo **vfat**.
   b) Montar el disquete y copiar en él el fichero `/etc/hosts`
   c) Desmontar el disquete.
   d) Comprobar que todo ha salido bien (habrá que montar el disquete de nuevo y visualizar su contenido)

2. Montar la partición Windows (partimos de la idea de que es el dispositivo `/dev/hda1`, si no se tiene o es otra partición hay que ajustarlo)

a) Crear un directorio destino de montaje:
```
# mkdir /mnt/windows
```
b) Montarla:
```
# mount -t auto /dev/hda1 /mnt/windows
```
c) Desmontar la partición y borrar el directorio creado.

3. Modificar el fichero `/etc/fstab` para que monte la partición Windows en el arranque (en el destino predefinido por Guadalinex).

4. Con el método de instalación de Guadalinex, todo, el sistema y nuestros documentos están en una misma partición (aunque en directorios bien diferenciados) ¿Qué tendríamos que hacer si necesitamos reinstalar el sistema?
Así que para los menos perezosos os proponemos crear una nueva partición del tipo ext3, que montaremos en el arranque sobre el directorio `/home` del directorio raiz. Vayamos por partes (en toda la práctica, como es lógico trabajaremos como root):

a) Redimensionar la partición windows (por ejemplo quitando 2 GB) y en el espacio liberado crear la nueva partición[23]. Para ello ejecutar:

```
# qtparted /dev/hda
```

Hemos supuesto que se va a hacer en este disco y supondremos en lo que sigue que la partición creada es la `/dev/hda4`.

b) Hacer una copia del contenido actual del directorio `/home`[24]. Para ello situaros en este directorio,
```
# cd /home
```

y

```
# tar -cvzf copiahome.tar.gz *
```

c) Crear el directorio `/copiahome` por ejemplo en `/mnt` y mover allí el fichero creado:
```
# mkdir /mnt/copiahome

# mv copiahome.tar.gz /mnt/copiahome

# cd /mnt/copiahome
```
y comprobar que allí está el fichero copia:
```
# ls -l
```

d) Ahora montamos la nueva partición, copiamos en ella el fichero `copiahome.tar.gz`, lo descomprimimos y lo desempaquetamos:
```
# mount -t ext3 /dev/hda4 /home

#cp copiahome.tar.gz /home

# cd /home

# tar -xvzf copiahome.tar.gz
```

[23] de forma similar a como se hizo en el proceso de instalación
[24] lo haremos con un `tar.gz` para mantener los permisos.

Ya hemos recuperado nuestro antiguo `/home` pero ahora físicamente está en una partición independiente. Ahora ya solo queda modificar el fichero `/etc/fstab` para que la monte en el arranque, así que editamos el fichero e incluímos una linea como esta:

e) `/dev/hda4 /home ext3 auto,user,exec,rw 0 0`

La próxima vez que arranquemos el sistema todo parecerá igual que al principio pero el directorio `/home` está en una partición independiente y si tenemos que reinstalar el sistema ya no tendremos que preocuparnos de salvar antes nuestros documentos.

2.6. Chequeo y recuperación: fsck.

El comando `fsck` nos permite chequear y, opcionalmente, reparar un sistema de ficheros (ext3 o ext2). `fsck` repara inconsistencias en los sistemas de ficheros después de un apagado incorrecto de la máquina.

Si trabajamos con el sistema de ficheros **ext3**[25], en general no surgen problemas por apagados incorrectos del ordenador y sólo es necesario chequear el sistema por problemas hardware del disco. Además, en caso de tener que recuperar un sistema de ficheros **ext3** no hay que chequear todo el sistema y el proceso es mucho más rápido (un par de segundos como máximo).

Si nuestro sistema de ficheros es de tipo **ext2**, el funcionamiento es el siguiente. Cuando un sistema de ficheros se monta, se marca como "sucio" porque el sistema en su trabajo normal tendrá datos mantenidos en memoria en vez de bajarlos directamente al disco, con el fin de mejorar las prestaciones. Si el sistema lo apagamos correctamente, una de sus tareas es descargar todo lo que se encuentre en la memoria y que corresponda a los discos a su lugar correspondiente. Tras hacer esto, se puede marcar el sistema de ficheros como "limpio". Si por cualquier razón la máquina se apagase[26] sin que se pueda realizar el proceso correcto de apagado, los discos se quedarán como "sucios" y deberán comprobarse al arrancar la máquina. Para chequear un sistema de ficheros éste debe estar desmontado.

Su sintaxis básica es:

```
fsck [-t fstype] [-fsopcion] device
```

en donde device es el fichero especial correspondiente al dispositivo y el único parámetro obligatorio. Por ejemplo:

```
# /sbin/fsck /dev/hda2
```

verificaría el volumen `/dev/hda2`.

Podemos indicarle al programa que repare los ficheros sin confirmación por nuestra parte con el parametro `-a`, por ejemplo:

```
# /sbin/fsck -a /dev/hda2
```

Los ficheros perdidos que recupere los podemos encontrar en el subdirectorio `/lost+found` del dispositivo `/dev/hda2`.

Es interesante comentar la opción

```
# /sbin/fsck -A
```

con esta orden `fsck` recorre el fichero `/etc/fstab` y verifica todas las unidadades en función del parámetro *fsckorder* del fichero `/etc/fstab`. Pero no se debería usar con sistemas de ficheros ya montados.

En los Guadalinex EDU de los centros no TIC se trabaja con el sistema de ficheros ext2[27]. Esto implica que cada vez que se apaga mal el sistema se queda "sucio" y tenemos que chequearlo de forma manual. El comando (más cómodo) a usar cuando ocurre esto es:

```
fsck -y /dev/hda1; reboot
```

[25] Para saber más sobre el sistema **ext3**: `http://linuxmobile.sourceforge.net/recursos/documentacion.html`

[26] Por ejemplo un apagón de luz.

[27] En los centros TIC esto ya no es así debido a que se ha actualizado el sistema. Inicialmente también ocurría.

En realidad se trata de dos comandos que se ejecutan de forma consecutiva, primero chequeamos el sistema diciendo que sí a todo (`-y`) y después reiniciamos el sistema (`reboot`). Si se trata de una máquina dual (con windows y linux) lo usual es que tengamos que sustituir `hda1` por `hda2`

Para más información sobre las opciones consultar el manual en línea.

➥ **Para practicar**

1. Chequear el disquete antes creado ¿qué problema surge?[28]
2. Si hemos hecho la práctica 4 anterior, chequear la partición `/home`:
   a) Si no sabemos cuál es podemos usar cualquiera de los comandos que siguen:
   ```
   $mount
   #/sbin/fdisk -l /dev/hda
   ```
3. Ya es sencillo, usar `fsck` tal cual se ha explicado.

2.7. Enlaces

El nombre de un fichero no es mas que una etiqueta que referencia a un número, **inodo**[29], que a su vez apunta al lugar físico donde se almacena la información que contiene el fichero.

Los enlaces permiten dar múltiples nombres a un fichero.

Enlaces duros (*hard link*) Un enlace duro es un nombre adicional para un fichero ya existente. Se crea con la orden **ln**. Por ejemplo, la orden :

```
ln notas.txt notashard.txt
```

Crea un "nuevo fichero" de nombre `notashard.txt` cuyo contenido es el mismo que el del fichro `notas.txt`.

Si ejecutamos la orden

```
ls -li notas*
1289347 -rw-r--r-- 2 juan users 82 2004-02-29 11:33 notashard.txt
1289347 -rw-r--r-- 2 juan users 82 2004-02-29 11:33 notas.txt
```

Podemos observar que ambos ficheros tienen los mismos permisos y el mismo número de inodo, el que aparece delante de los permisos. Son en realidad el mismo fichero con dos nombres distintos. Si hacemos cambios en uno de ellos, se reflejará en el otro. Si borramos por ejemplo `notas.txt`, `notashard.txt` seguirá existiendo, en realidad lo que hemos borrado ha sido una de las referencias al inodo, la otra sigue existendo. El 2 delante de juan indica que el fichero tiene dos enlaces.

No se pueden realizar enlaces hard entre ficheros de dos sistemas de ficheros distintos ni entre directorios.

Enlaces simbólicos Los enlaces simbólicos son ficheros que únicamente contienen el nombre de otro fichero[30]. Como un enlace simbólico apunta a un fichero (con su camino completo), es posible establecer enlaces simbólicos entre distintos sistemas de ficheros, y entre cualquier tipo de fichero, incluso con un fichero que no exista.

Se crean con la orden `ln` seguidos de la opción `-s`, por ejemplo:

```
ln -s /etc/fstab fstabsimb
```

Crea el fichero `fstabsim` que apunta al fichero `/etc/fstab`

La salida de la orden

[28] Si nos dice que no detecta el tipo es que habrá que decírselo con `-t vfat`

[29] Cada sistema de ficheros tiene su propia tabla de inodos.

[30] En cierto sentido se parecen a los "accesos directos" de Windows.

```
ls -li fstabsimb /etc/fstab
310523 -rw-r--r-- 1 root root 1049 2004-03-08 17:55 /etc/fstab
1290242 lrwxrwxrwx 1 juan users 10 2004-03-08 18:16 fstabesimb ->/etc/fstab
```

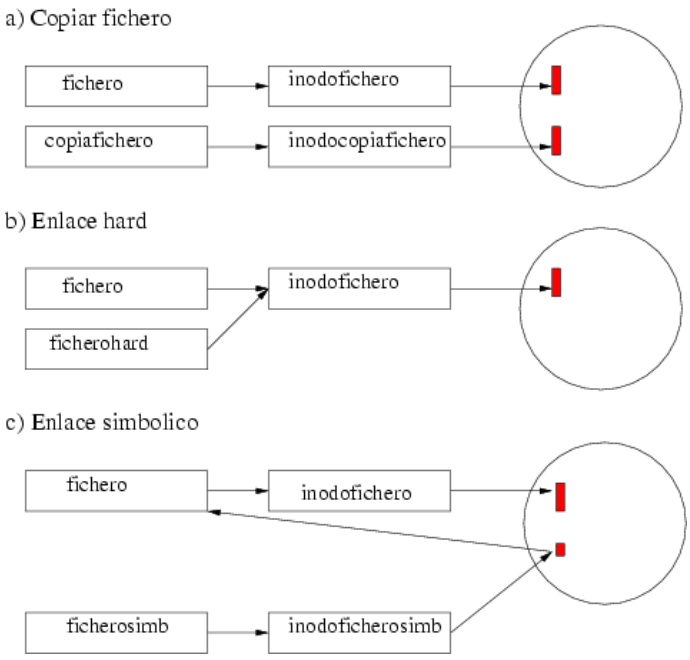
Nos muestra que:

- Los ficheros `fstabsimb` y `/etc/fstab` tienen inodos distintos.
- Los permisos del enlace simbólico estan todos activos, en realidad los permisos que imperan son los del fichero "apuntado".
- El listado nos muestra también hacia quien apunta el fichero `fstabsimb`.

Como en los enlaces duros, si modificamos uno se modifica el otro[31] y si borrásemos el "apuntado" el fichero simbólico seguiría existiendo pero sin apuntar a nadie.

Los enlaces simbólicos son muy utilizados por el sistema, fundamentalmente para enlazar con las imágenes de las librerías compartidas en `/lib`, pues este tipo de ficheros facilita la actualización del sistema.

El siguiente gráfico puede clarificar la diferencia entre copiar ficheros y los dos tipos de enlaces.



Para conocer más sobre este tema así como sobre inodos podéis consultar la página man de la orden `ln` o el libro *Linux Instalación y Primeros Pasos*, de MATT WELSH.

➥ Para practicar

1. Crear un enlace simbólico desde nuestro directorio de usuario al directorio `/var/log`:
   `$ln -s /var/log/ logs`
   a) Comprobar los efectos de ese enlace (logs apunta a un directorio).
   b) Borrar el enlace y crear uno para acceder al directorio `/tmp`
2. Crear otro que acceda al fichero `/var/log/debug`
3. Si tienes también windows en el equipo, monta la partición de windows y crea un enlace simbólico al directorio `Mis documentos` de windows.

[31] En el ejemplo, como ususario, no podremos modificar `fstabsimb` porque no tenemos permiso sobre `/etc/fstab`

Capítulo 3

Permisos. Gestión de Usuarios

> Complots apocalípticos futurísticos aparte, sin los usuarios, lor ordenadores no tienen objeto alguno.
> (*Red Hat Linux 7*, CHRISTOPHER NEGUS)

3.1. Introducción.

Linux es un sistema operativo multitarea y multiusuario. Esto posibilita que varios usuarios ejecuten distintas tareas a la vez, y en consecuencia, se hace necesario establecer algún mecanismo que proteja la información personal.

La forma que tiene el sistema de identificar a cada usuario es mediante la asignacion de *cuentas de usuario*. Cada usuario dispone de un nombre de usuario que lo identifica y además puede pertenecer a uno o varios grupos[1]. En un sistema Linux, aunque lo utilice una sola persona, al menos deberíamos tener dos cuentas, una para el root que debe utilizarse sólo cuando vayamos a efectuar labores de administración y otra como usuario de "a pie" para la realización del trabajo cotidiano.

La identidad de cada usuario junto con el grupo al que pertenece determina los derechos de acceso a ficheros y otros recursos del sistema.

3.2. Permisos de acceso a los distintos objetos.

En UNIX cada fichero tiene un propietario (inicialmente el usuario que lo creó) y pertenece a un grupo en particular. Basado en esta estructura, el sistema asigna permisos a los distintos objetos del sistema de ficheros a tres niveles:

1. a nivel del propietario,
2. a nivel del grupo al que pertenece el usuario, y
3. a nivel de todos los demás usuarios.

Para cada uno de estos tres niveles, asigna tres tipos de permisos básicos:

r de lectura

w de escritura, y

x de ejecución.

[1] al menos a un grupo con su mismo nombre de usuario que se crea al darle de alta.

Esta información está guardada en el inodo[2] del fichero. Se utilizan los bits 0 – 8 para los 9 permisos. Estos 9 permisos, así como el tipo de objeto del sistema de ficheros, pueden ser visualizados con la opción `-l` (formato largo) del comando `ls`. Supongamos que la salida de la ejecución de la sentencia:

```
$ ls -l /home/mercedes/linux2004/entrega1.lyx
```

es

```
-rwxr-xr--  1 mercedes gelinex 386 2003-9-1 22:37 /home/mercedes/linux2004/entrega1.lyx
```

analicemos la salida: `-rwxr-xr--`
para eso la vamos a dividir en cuatro bloques: **`-`** **`rwx`** **`r-x`** **`r--`**

- El primer guión por la izquierda (-) nos indica que `entrega1.lyx` es un fichero normal, si tuviera una

  d indicaría un directorio.

  l enlace simbólico

  c dispositivo de caracteres

  b dispositivo de bloques

  p canalización con nombre

- El siguiente grupo (**rwx**) indica que este fichero tiene permisos de lectura, escritura y ejecución para el propietario que en este caso es `mercedes`,
- El grupo **(r-x**) que el fichero tiene permisos de lectura y ejecución para el grupo que es `glinex`, y
- El último grupo (**r- -**) que el resto de usuarios tan sólo podrán leer el fichero pero no modificarlo ni ejecutarlo.

El significado de los permisos **rwx**, cuando el objeto es un directorio, es el siguiente:

r Permite leer el contenido del directorio, es decir los nombres de los ficheros y sus inodos, pero no la información de estos (con ese permiso puede ejcutarse el comando `ls` pero no `ls -l`).

w Permite escribir en el directorio, es decir crear y suprimir ficheros, otros subdirectorios, etc.

x Permite recorrer el directorio (podría hacer un `cd` a él-meterse dentro del directorio), y utilizar la información de los objetos del directorio, es decir, acceder a los inodos (se podría ejecutar `ls -l`).

Es importante resaltar que los permisos de un fichero están condicionados por los permisos del directorio donde reside. Por ejemplo, auque un fichero tenga los permisos `-rwxrwxrwx`, otros usuarios no podrían acceder a él a menos que tengan permiso de ejecución para el directorio en el que se encuentra el fichero. Así mismo, si un directorio tuviera permiso `w` para "otros usuarios" estos podrían borrar ficheros de ese directorio aunque los ficheros tuvieran deshabilitado ese permiso. Habitualmente los usuarios suelen dar a sus ficheros los permisos `-rx-r--r--` y a los directorios `-rwxr-xr-x`.

El propietario y grupo de un fichero lo podemos modificar con el comando `chown` (*change owner*- cambiar propietario); con `chgrp` (*change group*- cambiar grupo) podemos modificar sólo el grupo. Los permisos pueden modificarse con el comando `chmod` (*change mode*- cambiar modo). Naturalmente estas acciones sólo le están permitidas al root y al dueño del objeto. La sintaxis básica de las dos primeras es:

```
chown usuario fichero
chgrp grupo fichero
```

para cambiar el usuario o el grupo a un fichero. Después veremos cómo hacerlo con programas que nos facilitan el trabajo.

[2]Linux asigna a cada archivo un único número llamado inodo. Cuando formateamos un disco se crea la tabla de inodos. En el inodo de un archivo se almacena toda la información referente a ese archivo (propietario, permisos, tamaño...)

3.2.1. chmod

La sintaxis básica es[3]:

```
chmod modo fichero
```

Al comando `chmod` se le pasa como primer argumento los permisos que vamos a asignar al fichero que pasamos como segundo argumento. El primer argumento admite dos tipos de sintaxis: con notación octal o con notación nemónica. Para la notación octal del primer argumento del comando `chmod`, el primer dígito corresponde con el propietario, el segundo con el grupo, y el tercero con todos los demás. Cada dígito octal corresponde con tres dígitos binarios (octal codificado en binario): el primero para la lectura, el segundo para la escritura y el tercero para la ejecución. Si el dígito está a 1 el permiso está habilitado.

Octal	Binario	Permisos
0	000	(ninguno)
1	001	- - x
2	010	- w -
3	011	- w x
4	100	r - -
5	101	r - x
6	110	r w -
7	111	r w x

Por ejemplo, para colocarle al fichero `entrega1.lyx` los permisos: **`-rwxr-x--x`** utilizando la notación de la tabla, introduciríamos:

```
$ chmod 751 entrega1.lyx
```

umask

Por último, comentar que los permisos por defecto delos objetos dependen del programa que los crea y del valor de la variable `user mask`.Su valor se define en `/etc/profile`, podemos visualizarlo editando el fichero o con la orden :

```
$ umask
```

No obstante cada usuario puede modificarla de forma transitoria con la orden

```
$ umask valorumask
```

y de forma permanente en su fichero `.bash_profile`. Esta variable actúa como se explica en el siguiente ejemplo.

Ejemplo: si un programa crea ficheros con permisos 666 (`rw-rw-rw-`) y la `umask` es de 002 el fichero finalmente tendría de permisos 666-002=664, es decir, `rw-rw-r--`. Si embargo, si para otro usuario la `umask` es de 022, el mismo programa crearía los ficheros con permisos 666-022=644, o sea `rw-r--r--`

Para conocer mejor esta orden os remitimos al manual en línea.

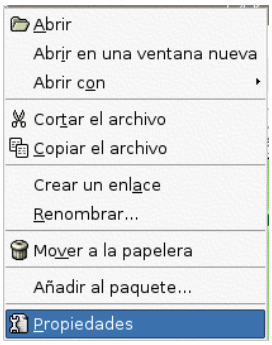
3.2.2. Permisos en modo gráfico

Además de los comandos ya comentados, podemos usar dos herramientas gráficas para modificar fácilmente los permisos de los ficheros:

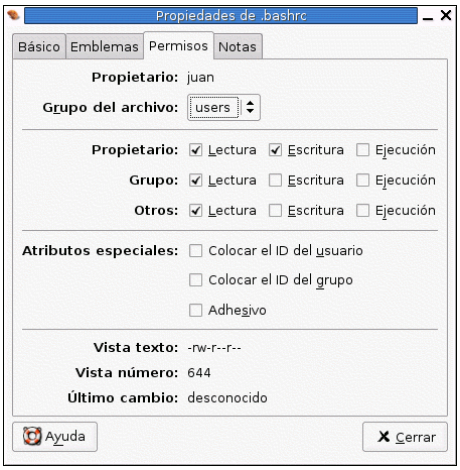
[3]Existe un parámetro opcional interesante que es `-R` (recursivamente), permite cambiar todos los permisos de los ficheros de un directorio y de todos sus subdirectorios (se puede usar también con `chown` y `chgrp`). Si queréis usarlo mirad cómo se hace en la documentación en línea.

- El gestor de ficheros **Nautilus**.

  Veamos cómo cambiar los permisos del fichero `~/.bashrc`[4]. Abrimos el navegador y pulsamos sobre el fichero con el botón derecho del ratón; aparecerá el menú de opciones:



  la opción **Propiedades** de este menú nos permite acceder a la ventana



  cambiar los permisos es ya un “juego de niños”. En este caso, el fichero en cuestión tendría de permisos: `-rw-r--r--` . Si modificamos algún permiso veremos como, automáticamente, cambia el número octal correspondiente al modo actual.

- Programa **mc**[5].

  Lanzar el programa, `$ mc`, seleccionar el archivo cuyos permisos queramos modificar, pulsar en el menú **Archivo** y del menú emergente elegir cambiar **P**ermisos...

[4] Ojo que el fichero es un fichero “punto” (oculto), o sea que hay que decirle al programa que nos muestre este tipo de ficheros: **Editar→Preferencias** y señalar la opcción.

[5] Estudiaremos mejor este programa en entregas posteriores.

3.2.3. Más sobre permisos

En las capturas gráficas anteriores podemos comprobar que aparecen tres campos más de los que no hemos hablado aún, se trata de los bits 9, 10 y 11, que corresponden con el *sticky bit*, el *SGID* (*Set Group ID*), y el *SUID* (*Set User ID*) respectivamente. Analicemos qué significado tienen:

- El *sticky bit* dice al sistema que el fichero que lo tiene activo tiene tendencia a ser ejecutado frecuentemente y debería ser retenido en el área de swap aún cuando no esté siendo ejecutado. Esto consume espacio de swap, pero reduce notablemente el tiempo de ejecución. Si el objeto del sistema de ficheros con el *sticky bit* activado es un directorio, no se permite a un usuario borrar ficheros de ese directorio (aunque sí pueda crearlos), salvo que los permisos de los ficheros lo permitan. Por último este bit activo nos lo muestra la salida del comando `ls -l` con una **t** en la posición que correspondería al permiso de ejecución para el resto de usuarios. Se puede utilizar para poner un directorio público, donde varios usuarios puedan escribir, pero no borrarse o modificar ficheros simultáneamente.

  El *sticky bit* corresponde con el valor octal 1000, de modo que si quisiéramos activarlo, utilizaríamos la sintaxis:

  ```
  $ chmod 1000 nombre_objeto
  ```

  La sintaxis anterior activaría el modo *sticky bit* a `nombre_objeto`, pero eliminaría todos los demás si los tuviera. Por tanto, si los permisos de `nombre_objeto` fueran: `-rwxr-x--` para mantener esos permisos y además activar el *sticky bit*, introduciríamos[6]:

  ```
  $ chmod 1750 nombre_objeto
  ```

- El *SUID*[7] le indica al kernel que el usuario que ejecute el fichero con este bit activo tome la personalidad, durante la ejecución, del usuario propietario del fichero. De esta forma, con ficheros SUID root, se soluciona, por ejemplo, el problema de escribir en el fichero `/usr/bin/passwd` por cualquier usuario, ya que cogería momentáneamente la personalidad del root, al ejecutar el comando passwd que es SUID root. Este bit activo nos lo muestra la salida del comando `ls -l` con una **s** en la posición que correspondería al permiso de ejecución para el propietario. Por ejemplo, podemos comprobarlo con

  ```
  $ls -l /usr/bin/passwd
  ```

- El *SGID* tiene un significado parecido, pero referido al grupo de usuarios al que pertenece el fichero. Así, cuando ejecutamos un programa que tiene activo este bit, nuestro GID toma el valor del GID

[6] La notación nemónica de `chmod` nos permitiría activar el *sticky bit*, o cualquier otro modo, sin tener que conocer cuáles son los modos actuales del fichero.

[7] Los modos con valores octales 4000 y 2000 son para activar el SUID y el SGID respectivamente.

del propietario del programa durante el tiempo de ejecución. Algunos sistemas (SVR4) utilizan este bit con los directorios: si el directorio tiene el SGID activado, los nuevos ficheros heredan el GID del directorio y los subdirectorios heredan el GID y el bit SGID; en otro caso tanto los ficheros como los directorios son creados con el GID primario del proceso que los crea. Este bit activo nos lo muestra la salida del comando `ls -l` con una **s** en la posicion que correspondería al permiso de ejecución para el grupo. Si bien estos bits aportan funcionalidades importantes al sistema, no están exentos de riesgos en cuanto a lo que seguridad se refiere.

La notacion nemónica de chmod

Mediante la notación nemónica (o simbólica) también podemos asignar permisos a los ficheros. Es equivalente a la octal y podemos utilizar cualquiera de las dos.

La sintaxis básica es:

```
chmod [usuario][operador][permiso] fichero
```

En el lugar de usuario podemos poner:

u propietario (*user*)
g grupo (*group*)
o el resto (*other*)
a todos (*all*)

El operador puede ser:

+ añade el permiso

- quita el permiso

= fija el permiso

Los permisos pueden ser:

r lectura

w escritura

x ejecución

s set uid o gid

t sticky bit

Veamos su significado con ejemplos:

```
# chmod o=r fichero
```

Con esta orden fijamos los permisos para el resto de la gente (los que no son ni el propietario del fichero, ni del grupo al que pretenece el fichero) como de sólo lectura. El equivalente octal sería un 4 en la posición de otros.

```
# chmod u+x fichero
```

Con este comando añadimos (+) al propietario (u) el permiso de ejecución (x)
Podíamos haber hecho las dos modificaciones de una vez con la orden:

```
# chmod o=r,u+x fichero
```

Es importante destacar que con la adición no modificamos el resto de los permisos. Si este fichero tuviera permiso de lectura, éste no se vería modificado.

3.3. Gestión de usuarios en modo texto.

El sistema, para cada usuario, mantiene cierta cantidad de información que resumimos a continuación:

nombre de usuario El nombre de usuario es el identificador único dado a cada usuario del sistema.

user ID El **UID** es un número único que el sistema asigna a cada usuario y por el que lo identifica.

group ID El **GID** es el número del grupo por defecto del usuario (su grupo principal).

clave La clave de usuario, que se almacena de forma encriptada. El comando `passwd` se utiliza para poner y cambiar la clave de los usuarios.

nombre completo El nombre real del usuario.

directorio inicial El directorio inicial de trabajo del usuario. Normalmente cuelga de `/home` con el mismo nombre que el usuario.

intérprete de inicio El intérprete de comandos que es arrancado para el usuario en tiempo de conexión, suele ser `/bin/bash`.

Toda esta información se almacena en el fichero `/etc/passwd`.

LINUX proporciona herramientas muy potentes para la administración de usuarios y de grupos, tanto en modo comando como en modo gráfico. Las órdenes[8] que usaremos para gestionar los usuarios del sistema son:

adduser para añadir un usuario[9].

passwd para asignarle contraseña a un usuario o modificarla[10].

deluser permite eliminar un usuario.

Supongamos que queremos añadir el valiente usuario THALES CICA a nuestro sistema, y que la contraseña con la que va a poder entrar en el sistema va a ser Averroes, para esto escribiremos[11]:

```
# adduser thales
```

El sistema solicita la password (contraseña) del usuario. Como es obvio al escribir la contraseña de nuestro usuario no veremos los caracteres que introducimos y habrá que volverla a teclear para confirmarla.

A continuación nos solicita información adicional del usuario; esta información es opcional y podemos dejar campos en blanco

```
Full Name []: Thales Cica
Room Number []:
Work Phone []:
Home Phone []:
Other []:
```

y confirmamos que la información es correcta.[12]

[8] Como siempre no estamos usando todas las opciones de estos scripts, para conocerlas en profundidad os remitimos a las páginas `man` de cada uno de ellos.

[9] También se puede usar `useradd` pero `adduser` es mejor como podéis comprobar.

[10] sólo el root o el propio usuario puede modificar su contraseña.

[11] Los valores por defecto se establecen en `/etc/adduser.conf`

[12] Observa que el sistema ha incluído al usuario en los grupos audio, dip (para poder conectar a internet), disk, floppy, cdrom. La razón es clara: para que el nuevo usuario pueda acceder a ellos.

Los grupos a los que se añadirán los nuevos usuarios los podemos establecer desde el fichero `/usr/local/sbin/adduser.local`

Para ampliar, consultar:

`http://www.debian.org/doc/manuals/securing-debian-howto/ch11.es.html`

`http://www.augcyl.org/glol/old/N_1/admbasica.html`

El sistema habrá leído el fichero `/etc/login.defs`, en él se establecen los valores por defecto a la hora de crear cuentas de usuario. También se creará un directorio de inicio para el usuario en el subdirectorio `/home`, de nombre `thales` cuyo contenido se basa en el directorio de inicio de la plantilla configurada en `/etc/skel`. Además, se añadirán entradas en los ficheros: `/etc/passwd`, `/etc/shadow` y `/etc/group`[13].

Cada vez que añadimos un usuario se modifica el archivo de contraseñas de usuarios, este archivo es `/etc/passwd`. Si lo mirásemos ahora veríamos que la última línea es:

```
thales:x:1003:1003:Thales Cica„,:/home/thales:/bin/bash
```

En esa línea hay distintos campos separados por ":", el significado de ella es que el usuario thales tiene de contraseña "x[14]", que el número que biunívocamente utiliza el sistema operativo para él es 1003 (UID[15]), igual que su número primario de grupo (GID), después vemos el nombre completo[16] y por último el shell[17] que va a ser utilizado por este usuario al arrancar. ¿Fácil no? **Arj**, ¡me salté un campo! el penúltimo campo es el subdirectorio inicial de este famoso usuario.

Pero después de un tiempo nuestro usuario ya no va a utilizar más nuestro sistema así que le daremos de baja, para ello basta con ejecutar:

```
# deluser thales
```

Con esta orden se ha eliminado a nuestro usuario de los ficheros `/etc/passwd` y `/etc/shadow` pero no se ha borrado su directorio de trabajo ni algún otro asociado con sus aplicaciones. Para que también se hubiesen borrado deberíamos haber añadido la opcción -r al comando `userdel`, así

```
# userdel -r thales
```

no habría dejado rastro de nuestro usuario. Naturalmente hay otras formas de borrar esos directorios.

Si por alguna razón lo que quisiéramos fuera deshabilitar temporalmente la cuenta de un usuario, bastaría con anteponer un asterisco "*" en el campo de la clave en `/etc/passwd`. Por ejemplo,

```
thales:*x:1003:1003:Thales Cica„,:/home/thales:/bin/bash
```

impedirá que THALES entre en el sistema.

Cada usuario es miembro de al menos un grupo. El sentido de trabajar con grupos es que se pueden dar privilegios de acceso a determinados ficheros a todos los usuarios pertenecientes a un grupo. La información sobre los grupos se guarda en `/etc/group`.[18]


Algunas órdenes para trabajar con grupos son:

groups ver los grupos a los que pertenece el usuario pasado como argumento.

addgroup crear un nuevo grupo

delgroup borra un grupo[19]

3.4. Gestión de usuarios en modo gráfico.

El sistema proporciona una herramienta gráfica para la gestión de usuarios y grupos que no puede ser más fácil de utilizar. Se accede a ella con:


Aplicaciones→Herramientas del sistema→Panel de contros→Usuarios y grupos
o desde un terminal con la orden `# users-admin`.

[13] Cada vez que se añade un usuario al sistema se crea un grupo del que él sólo forma parte, se trata del grupo privado de ese usuario.

[14] No puede ser tan fácil ver la contraseña ¿verdad?

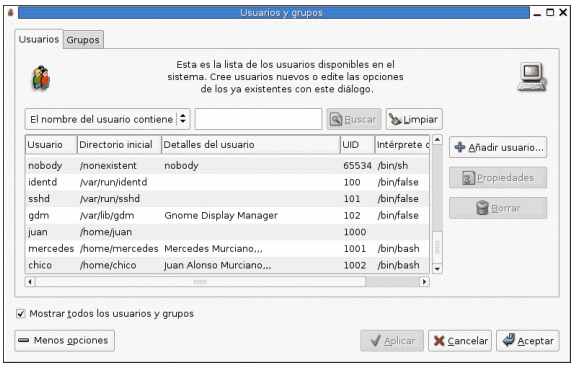
[15] Los *uids* identifican a los propietarios de los archivos, directorios y procesos.

[16] Si hemos añadido más datos sobre él, irían aquí separados por comas.

[17] Si ponemos `/bin/false` ese usuario no podrá ejecutar comandos de nuestra máquina aunque sí podrá acceder a ella.

[18] De nuevo si se desea ampliar, consultar las Web ya comentadas:
`http://www.debian.org/doc/manuals/securing-debian-howto/ch11.es.html`
`http://www.augcyl.org/glol/old/N_1/admbasica.html`

[19] es una buena idea mirar las páginas man tanto de los comandos de grupo como de los de usuarios.



3.5. ➥ Para practicar: Varias sesiones abiertas

Hemos dicho muchas veces que Linux es multiusuario pero hasta ahora no hemos utilizado esta posibilidad para nada.[20]En esta práctica haremos ver que podemos estar trabajando varios usuarios a la vez. Para hacer uso de esta posibilidad en modo gráfico es preciso previamente borrar el fichero `.xsession` del directorio `$HOME` de cada usuario que vaya a abrir una sesión gráfica.[21]

Supongamos que hasta ahora sólo están registrados en el sistema el root y el usuario de "a pie", **nano**. Así que vamos a dar de alta a un nuevo usuario de nombre thales. Para ello,

```
$ su
$ Pasword:
```

y tras introducir la clave de root,

```
# adduser thales
```

Tras introducir la pasword del usuario thales y otros datos adicionales,

```
# exit
```

Habíamos entrado en el sistema en modo gráfico como usuario NANO pero THALES necesita hacer uso del él con todos sus privilegios (no de "prestado" `$ su thales`). Para ello pulsamos **[CTRL]+[ALT]+[F1]** y aparece

```
login:
```

Tras introducir su nombre y contraseña, thales ha entrado en modo texto. Pero quiere trabajar en modo gráfico así que teclea:

```
$ startx -- :1
```

y ¡observa!, aparece un nuevo display[22] gráfico para él.

Para pasar de nuevo al display de NANO, pulsamos **[CTRL]+[ALT]+[F7]** y para volver al de thales **[CTRL]+[ALT]+[F8]**. Ambos pueden estar ejecutando aplicaciones, mientras uno descarga un fichero de tamaño considerable el otro puede mirar su correo.

[20]Hemos utilizado el comando **su** para cambiar momentáneamente de "personalidad" pero sigue habiendo solamente un usuario haciendo uso del sistema.

[21]`$ rm /home/nano/.xsession` para borrar el del usuario **nano**. Si como root borramos el fichero `/etc/skel/.xsession` habremos resuelto el problema para todos los usuarios que demos de alta.

[22]Se llama así al conjunto formado por un monitor, un teclado y un ratón. El display 0 pertenece al usuario NANO, de ahí que la llamada `startx` deba llevar el modificador **:1** (los espacios son necesarios)

Pero queremos trabajar como **root** para hacer ciertas labores de administración:

Pulsamos **[CTRL]+[ALT]+[F2]** y de nuevo, tras introducir el login y la password de root, hemos entrado en modo texto (pueden efectuarse tareas en modo texto), pero nos gusta más el modo gráfico, así que ahora:

```
# startx -- :2
```

y un nuevo display para el root. Con **[CTRL]+[ALT]+[F7]** volvemos al de nano y con **[CTRL]+[ALT]+[F9]** al del root. Con

[CTRL]+[ALT]+[F1] al modo texto de thales y con **[CTRL]+[ALT]+[F2]** al modo texto de root.

Tenemos a nuestra disposición 6 terminales de texto **[CTRL]+[ALT]+[Fi]**, i=1 ... 6 y otros 6 gráficos **[CTRL]+[ALT]+[Fj]**, j=7 ... 12 y ya hemos visto cómo movernos de uno al otro.

3.6. ➥ Para practicar

En esta práctica, se propone crear un entorno de trabajo que ofrezca una posible solución al problema de la compartición de ficheros por parte de una clase de alumnos/departamentos didácticos de un centro.

El problema se va a plantear proponiendo crear un entorno de trabajo para realizar prácticas:

- Habrá un profesor (profesor),
- alumnos (alumno1 ... alumno4) y
- grupos de prácticas (grupo1 y grupo2).

Por extensión se podría aplicar a un grupo más amplio.

Se trata de conseguir:

1. Para las prácticas individuales, cada alumno tendrá un directorio `/home/alumnoi`. Dentro de él, deben crear el fichero `practica_individuali`. Ese fichero podrá ser visto por el profesor pero no podrá modificarlo. Cada alumno sólo tendrá acceso a sus ficheros.
2. Para las prácticas en grupo cada grupo dispondrá de su directorio, `/home/grupo1`, `/home/grupo2` en el que podrán escribir los miembros del grupo. Todos los alumnos pertenecientes a un grupo (alumno1 y alumno2 pertenecen al grupo1 y alumno3 y alumno4 al grupo2) pueden escribir en el directorio del grupo pero no pueden leer ni escribir en el directorio de otros grupos. El profesor podrá ver los ficheros de este directorio pero no modificarlos.
3. Habrá un directorio `/home/clase_linux` en el que todos los alumnos tendrán un fichero de nombre `alumnoi`, que sólo podrá modificar dicho alumno, pero podrá ser leído por todos los demás.

Tenemos que ver qué permisos deben tener los ficheros y directorios para conseguir las condiciones anteriores.

El problema se puede resumir en determinar:

1. Permisos del directorio `/home/alumnoi` y del fichero `/home/alumnoi/practica_individuali` para llevar a cabo la política de accesos adecuadamente. El profesor, sin ser root (superusuario), deberá poder acceder al contenido de todas las prácticas individuales.
2. Permisos del directorio `/home/grupoi` y del fichero `/home/grupoi/practica_grupoi` para llevar a cabo la política de accesos a prácticas de grupo. Por supuesto, el profesor podrá consultar las prácticas del grupo sin ser superusuario.
3. Permisos del directorio `/home/clase_linux` y de los ficheros `alumnoi` dentro de ese directorio.

3.6.1. SOLUCIÓN

Prácticas individuales.

1. **Creación de los usuarios:**
   Creamos el usuario profesor mediante el método que prefiramos. Por ejemplo:
   `#adduser profesor`[23]

   Después creamos los distintos usuarios-alumnos[24]:
   `#adduser alumnoi (i:1..4)`

   Cada usuario que creamos pertenece a un grupo propio, es decir, el propio nombre de usuario. Así al crear el usuario alumno1, el identificador de usuario será alumno1 y el grupo al que pertenece será alumno1.
   Los permisos que tiene el directorio de cada usuario por defecto son **drwx r-x r-x**
   Hacemos que el profesor pertenezca al grupo alumnoi, editando y añadiéndolo en la entrada correspondiente al grupo alumnoi del fichero `/etc/group`[25]
   ```
   alumno1:x:1001:profesor
   ...
   alumno4:x:1004:profesor
   ```

2. **Modificación de los permisos:** Para ver los permisos de los distintos directorios de usuario lo haremos con la orden
   `#ls -l /home`
   Si los permisos del directorio cuando se crea son:
   ```
   drwxr-xr-x    5 alumnoi  alumnoi      4096 oct  9 20:33 alumnoi
   ```

   Los modificaremos con
   `#chmod o-rx /home/alumnoi`

   quedando:
   ```
   drwxr-x---    5 alumnoi  alumnoi      4096 oct  9 20:33 alumnoi
   ```

   Los permisos del usuario no necesitan comentario. Para el grupo alumnoi (al cual el profesor pertenece) se permite `r` (ver qué ficheros hay en el directorio) y `x` (poderse meter dentro del directorio. Hacer `cd /home/alumnoi`). Con esto conseguimos que el profesor pueda entrar en el directorio del alumno.
   Es importante destacar lo que no se puede hacer. El profesor, aunque podrá entrar en el directorio, no podrá crear nuevos ficheros, porque no tiene el permiso de escritura en el directorio.
   El resto de usuarios del sistema, no podrán entrar, ver, ni modificar nada del directorio del alumno.

3. **Crear los ficheros `practica_individuali` para cada uno de los alumnos**
   Podemos optar por crearlos con un editor de textos (cuidado con quién los crea, si es el root[26], él será su dueño y después habrá que cambiar propietario y grupo), o bien con el comando:
   `$ touch practica_individual1`

   que crea un fichero vacío.

[23] La creación de usuarios y grupos que en la práctica hacemos en modo comando, puede hacerse en modo gráfico.

[24] Para cada usuario podemos asignar o cambiar las contraseñas de acceso con el comando

```
#passwd usuario
```

[25] Se puede en modo texto, como acabamos de decir o con la herramienta gráfica que hemos visto. Los UID de los usuarios no tienen por qué corresponder con los que aparecen a continuación.

[26] Recordar que podemos pasar de ser un usuario a otro usando el comando:
`$su usuario`
e introduciendo después la contraseña. Para salir
`$exit`.

Los permisos del fichero `practica_individual1` pueden ser:

```
-rw-r--r--    1 alumno1  alumno1         4 oct  9 20:34 practica_individual1
```

Así, el profesor podrá revisar la práctica del alumno, pero no la podrá modificar. Si el fichero se ha creado con permisos 664, tendremos que cambiarlos, bien como root, bien como alumno1

```
#chmod 644 /home/alumno1/practica_individual1
```

Práctica de grupo

Creamos los grupos

```
#addgroup grupo1
#addgroup grupo2
```

y añadimos en el fichero `/etc/group` a alumno1, alumno2 como usuarios pertenecientes al grupo1 (lo mismo haremos con los alumnos 3 y 4 para el grupo 2)[27].

```
grupo1:x:1006:alumno1,alumno2
```

Podemos comprobar cómo va el proyecto ejecutando:

```
$id usuario
```

y/o

```
$groups usuario
```

Después crearemos el directorio `/home/grupo1:`

```
#mkdir /home/grupo1
#chgrp grupo1 /home/grupo1
#chown profesor /home/grupo1
```

Nos interesa que el directorio (lo mismo para el grupo2) tenga los permisos

```
drwxrwx---    2 profesor grupo1       4096 oct  9 22:43 grupo1
```

Con esto, el profesor podrá acceder al directorio del grupo, ya que es el propietario, y los alumnos, por el hecho de pertenecer al grupo (grupo1) podrán `r` (ver el contenido del directorio), `x` (meterse en el directorio) y `w` (escribir en el directorio, que equivale a poder crear ficheros dentro del directorio).

Pero los permisos por defecto son:

```
drwxr-xr-x
```

Para cambiarlos y conseguir el efecto deseado, ejecutaremos la orden:

```
#chmod 770 /home/grupo1
```

El resto de alumnos que no pertenecen al grupo1, no podrán hacer nada (ni por supuesto copiar la práctica). Con sólo esto, si el alumno1 hace

```
$cd /home/grupo1
```

y crea el fichero `practica_grupo1`, éste se crearía con los permisos:

```
-rw-rw-r--    1 alumno1  alumno1         2 oct  9 22:42 practica_grupo1
```

y el resto de los alumnos del grupo, podrían ver la práctica, pero no modificarla.

Para solucionar esto, hacemos

```
#chmod g+s /home/grupo1
```

quedando el directorio

```
drwxrws---    2 profesor grupo1       4096 oct  9 22:43 grupo1
```

[27] El GID de ejemplo no tiene por qué ser el que aparezca en vuestro fichero.

El bit `s` en directorios hace que el fichero que se cree dentro de él, posea como grupo al mismo grupo al que pertenece el directorio. Así, el grupo del fichero se mantendrá como grupo1 y el resto de alumnos que pertenecen al grupo, podrán modificar (trabajar sobre) la práctica, aunque el propietario haya sido el alumno que lo ha creado.

```
-rw-rw-r--    1 alumno1  grupo1          2 oct  9 22:43 practica_grupo1
```

/home/clase_linux

Creemos el directorio `/home/clase_linux` con el profesor como propietario y perteneciente al grupo profesor.

```
#mkdir /home/clase_linux
#chgrp profesor /home/clase_linux
#chown profesor /home/clase_linux
#chmod a+rwx /home/clase_linux
```

el resultado sería:

```
$ls -l /home
...
drwxrwxrwx    2 profesor profesor     4096 oct  9 23:12 clase_linux
...
```

Si fueran estos permisos, los alumnos (mediante el `rwx` del resto de usuarios, puesto que no son ni propietarios ni pertenecen al grupo) pueden crear los ficheros `alumno`i, pero otros alumnos podrán borrar los ficheros de otros alumnos, con el consiguiente peligro. Una forma de evitar esto es mediente el *sticky bit*. Hacemos

```
#chmod +t /home/clase_linux
```

El efecto es que aunque el directorio sea de escritura pública (`rwx` para todos) solamente el propietario del fichero podrá borrarlo.

```
$ls -l /home/clase_linux
drwxrwxrwt    2 profesor profesor     4096 oct  9 23:21 clase_linux
```

Supongamos que alumno1 crea el fichero `/home/clase_linux/fichero1` de permisos

```
-rw-rw-r--    1 alumno1  alumno1          0 oct  9 23:21 fichero1
```

Aunque por los permisos del directorio pudieran borrarlo otros alumnos, el bit `t` hace que sólo el propietario (alumno1) pueda borrarlo[28]. Además, los permisos de este fichero impiden que pueda modificarlo otro alumno.

[28] Es lo que pasa en el directorio `/tmp` de uso público.

Capítulo 4

Instalación, desinstalación de paquetes y actualización del sistema

> ¿Desea instalar o eliminar una aplicación? No hay problema. ¿Desea actualizar un programa que ya ha instalado? Muy fácil. Con un par de simples comandos o pulsando algunos botones este proceso lo podrá realizar usted mismo. *Red Hat Linux 7.0: The Official Red Hat Linux Getting Started Guide*

4.1. apt - Introducción

En este capítulo vamos a estudiar cómo instalar y desinstalar programas. En GuadaLinex existen diferentes métodos de instalación y desinstalación de paquetes

Estudiamos el comando `apt` aparte de los otros comandos debido a la "importancia" merecida que tiene. Este sistema es muy avanzado y de gran flexibilidad y potencia para entornos de red. Con respecto a los programas gráficos para instalar y desinstalar programas tan sólo comentaremos los aspectos más básicos para trabajar con ellos ya que, si se sabe qué se puede hacer con el programa `apt`, su manejo es casi inmediato.

Se puede ampliar sobre el tema en:

- La página man del programa[1].
- Y sobre todo la web: `http://www.debian.org/doc/`

4.1.1. ¿Qué es apt?

APT son las siglas de *Advanced Packaging Tool*, es decir, *herramienta avanzada de empaquetamiento*. El sistema APT es un sistema abierto, basado en la licencia GNU y desarrolado por el ATP Team[2]. Este sistema fue adaptado por la distrubucion Conectiva para poder usarse con `rpm`[3] y ha sido adoptado por otras distribuciones como:

- Conectiva `http://www.conectiva.com.br`
- Fedora `http://fedora.redhat.com/`
- Mandrake `http://www.mandrake.com`
- PLD `http://www.pld.org.pl`

[1] Recuerda: `$ man apt`

[2] Si bien el sistema RPM es el usado por RedHat, Caldera, SUSE, etc, la realidad es que un paquete creado para una determinada distribución no siempre se instala bien en las otras.

[3] RPM son las siglas de RedHat Package Management, es decir, sistema de paquetes de RedHat

- Vine `http://www.vinelinux.org`
- APT4RPM `http://apt4rpm.sf.net`
- Alt Linux `http://www.altlinux.ru`
- Red Hat `http://www.redhat.com`
- Sun Solaris `http://www.sun.com`
- SuSE `http://www.suse.de`
- Yellow Dog Linux `http://www.yellowdoglinux.com`

La idea de paquetes es específica del mundo Linux/UNIX y se justifica en la filosofía imperante en el mundo UNIX: la de disponer de pequeñas utilidades que aunque hagan una sola tarea la hagan muy bien. Mezclando estas "pequeñas" utilidades podremos resolver cualquier problema por grande y complejo que sea. Además, estos programas usan librerías compartidas con el objetivo de minimizar el código duplicado y optimizar el uso del disco y de la memoria. Esta forma de organizar el sistema obliga a mantener un control estricto sobre los programas y librerías instaladas en nuestro equipo.

Aclaremos qué es un **paquete**: es un archivo que contiene todos los ficheros de un determinado componente instalable y que además almacena información de control y *scripts* que se ejecutan al instalar o borrar el paquete.

Con un sistema de paquetes se pretende mantener un control efectivo sobre las aplicaciones (programas, librerías, etc) que instalemos en nuestra máquina y las modificaciones realizadas sobre ellas.

Las características fundamentales de los sistemas de paquetes son[4]:

- Mantienen una base de datos en la que se almacena la información de todos los paquetes, tanto los paquetes instalados como los ficheros que contiene cada paquete.[5]
- Control sobre dependencias: controlando las dependencias antes de instalar o desinstalar un paquete sabemos si ese paquete es usado por otros paquetes o librerías para funcionar correctamente. Si no tenemos instalado ese programa/librería, antes de instalar se nos avisará. Lo mismo sucederá si queremos desinstalar un paquete que es necesario para otras aplicaciones.
- Control sobre las incompatibilidades: si intentamos instalar un paquete que va a impedir que otro que ya tenemos instalado funcione correctamente surgirá una incompatibilidad.
- Podemos instalar paquetes sin tener que reiniciar el equipo.[6]

El sistema de paquetes APT sigue una "nomenclatura" que permite identificarlo, se basa en dar de cada paquete los campos: nombre, versión, revisión, plataforma y extensión. Por ejemplo, consideremos el paquete

`xpdf-reader_2.03-2_i386.deb`

veamos qué significa cada uno de esos campos:

xpdf-reader es el nombre del paquete.

2.03 es el número de versión.

-2 número de la revisión, en este caso indica que es la segunda modificación realizada.

i386 nos indica la plataforma para la que está construido. En plataformas Intel disponemos de: i386, i586, athlon, i686. Si bien un paquete para la plataforma i386 podremos instalarlo en cualquier máquina Intel o compatible, uno con "extensión" i686 será sólo para micros Pentium II (o compatibles) y superiores.

deb La extensión común a todos los paquetes *Debian.*

[4] RPM o deb

[5] Podremos saber en cualquier momento si hemos modificado los ficheros de un determinado paquete, comprobar la integridad de un paquete o saber a qué paquete pertenece un determinado fichero.

[6] Como en otros sistemas operativos, ¿verdad?

4.1.2. El archivo /etc/apt/sources.list

Parte fundamental del funcionamiento de `apt`, es el archivo en que están localizadas las "fuentes"[7] en donde se encuentran los paquetes. Este archivo es:

`/etc/apt/sources.list`

Si miramos en nuestro GuadaLinex nos aparecerá:

```
# Junta de Andalucía (Repositorio raiz)
deb http://http.guadalinex.org/repositorio liron main contrib non-free guada
deb http://http.guadalinex.org/repositorio liron/non-US main contrib non-free

# DEBIAN OFICIAL
#deb http://ftp.fi.debian.org/debian sid main contrib non-free
#deb http://non-us.debian.org/debian-non-US sid/non-US main contrib non-free
#deb-src http://ftp.fi.debian.org/debian sid main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US sid/non-US main contrib non-free
#deb http://ftp.fi.debian.org/debian ../project/experimental main contrib non-free
```

Todas las líneas de este fichero están comentadas[8] (están encabezadas por el símbolo "#") salvo dos. Ambas líneas comienzan con `deb`, esto quiere decir que será en los "*sitios*" que figuran a continuación donde buscará los paquetes binarios (deb) a descargar. Estos paquetes ya están pre-compilados y son los que normalmente se usan.

Aparecen otras líneas comentadas encabezdas con `deb-scr`, no debemos quitar el "#" salvo que deseemos descargar los paquetes fuente. Estos paquetes son los códigos originales con los que está hecho un programa[9].

El fichero`/etc/apt/sources.list` puede contener varios tipos de líneas. APT sabe perfectamente cómo interpretar si son `http, ftp, ssh, file` (archivos locales).

 Si realizamos algún cambio en este fichero **siempre** deberemos ejecutar el comando:

```
# apt-get update
```

4.1.3. Agregar un CD-ROM al archivo sources.list

Puede ser que los paquetes que deseamos instalar los tengamos en un cd-rom y no necesitemos conectarnos a internet para descargarlos. Lo más sencillo es incorporar el cd-rom al archivo `sources.list`, para lo que ejecutaremos:

`# apt-cdrom add`

Nos solicitará un nombre para ese cd-rom en concreto, por lo que debemos indicárselo.

↳ Veamos un ejemplo:

```
guadalinex:/home/fermin# apt-cdrom add
Using CD-ROM mount point /cdrom/
Unmounting CD-ROM
```

[7] Máquinas de internet a las que nos conectaremos para bajar los programas.

[8] El que una línea de un fichero aparezca comentada, #, quiere decir que lo que figure a continuación será ignorado, no se leerá ni ejecutará.

[9] Cuando decimos que Linux es "Código abierto" (en inglés, *Open Source*) nos referimos precisamente a esta característica. Cualquier persona es libre de ver, modificar y optimizar su código. De esta manera sabemos cómo funciona realmente una aplicación, sus virtudes y tambien sus defectos y fallos. Al tener acceso al código fuente podemos corregirlo en favor de toda la comunidad y perfeccionarlo. El código fuente es lo que Microsoft, tan celosamente, guarda y no libera. Solamente ellos pueden modificarlo.

El método de desarrollo de la comunidad Linux es mucho mejor que el de Microsoft. Cuando se detecta un fallo, sobre todo en lo referente a la seguridad, inmediatamente es corregido y puesto en conocimiento. No en vano hablamos de programadores y desarrolladores de todo el planeta. Mientras tanto, cuando se detecta un fallo en una aplicación de Microsoft, ésta se corrige o bien en la siguiente versión (previo pago, por supuesto) o bien mediante algún parche, que en algunos casos graves de seguridad se ha resuelto hasta 6 meses después de haberse conocido dicho "agujero".

```
Please insert a Disc in the drive and press enter
Mounting CD-ROM
Identifying.. [4e60a51092b5d4ac3ea845878f993177-2]
Scanning Disc for index files.. Found 3 package indexes and 0 source indexes.
Please provide a name for this Disc, such as 'Debian 2.1r1 Disk 1':
cursolinux
This Disc is called:
'cursolinux'
Reading Package Indexes... Hecho
Wrote 111 records.
Writing new source list
Source List entries for this Disc are:
deb cdrom:[cursolinux]/ stable contrib main non-free
Repeat this process for the rest of the CDs in your set.
```

La última línea nos indica que este proceso deberemos repetirlo para el resto de cd-rom's que queramos añadir.

Veamos cómo queda el fichero `/etc/sources.list`

```
# Junta de Andalucía (Repositorio raiz)
deb cdrom:[cursolinux]/ stable contrib main non-free
deb http://http.guadalinex.org/repositorio liron main contrib non-free guada
deb http://http.guadalinex.org/repositorio liron/non-US main contrib non-free

# DEBIAN OFICIAL
#deb http://ftp.fi.debian.org/debian sid main contrib non-free
#deb http://non-us.debian.org/debian-non-US sid/non-US main contrib non-free
#deb-src http://ftp.fi.debian.org/debian sid main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US sid/non-US main contrib non-free
#deb http://ftp.fi.debian.org/debian ../project/experimental main contrib non-free
```

Debemos tener en cuenta que esto sólo funcionará si el cd-rom está debidamente configurado en el archivo `/etc/fstab` del sistema.

Si no fuese esta la ruta de montaje del dispositivo, deberíamos configurarlo a mano, como por ejemplo:

```
# apt-cdrom -d /home/usuario/midisco add
```

Las opciones que debemos tener en cuenta son:

-d directory punto de montaje del cdrom

-r renombrar a un cdrom ya identificado

-m no montarlo

-f modo rápido sin comprobar los paquetes que hay en él

-a modo de escaneo

También tenemos la posibilidad de identificar un cd-rom sin agregarlo al archivo `sources.list` mediante la orden:

```
# apt-cdrom ident
```

4.1.4. Instalar paquetes

El sistema de paquetes utiliza una base de datos para monitorizar los paquetes instalados, los no instalados y los que están disponibles por si deseamos instalarlos.

A la hora de instalar paquetes utilizaremos el programa `apt-get` el cual utilizará la base de datos anteriormente comentada y averiguará cómo instalar los paquetes que le indicamos, así como aquellos otros paquetes adicionales, si son necesarios, que serán requeridos por el paquete que deseamos instalar para que funcione correctamente.

Para actualizar la lista, se utiliza el comando:

```
# apt-get update
```

Este comando busca el paquete solicitado en los archivos listados en `/etc/sources.list`

Si tenemos conexión a internet buscará las fuentes en los repositorios indicados en `/etc/sources.list`

No está de más realizar esta operación con cierta frecuencia para mantenerse informado sobre las posibilidades de actualización del sistema, sobre todo en lo referente a las actualizaciones de seguridad.

Una vez que tenemos el archivo `sources.list` preparado y la base de datos actualizada llega el momento esperado ¡vamos a instalar paquetes!

Nada mejor para explicarlo que con un ejemplo. Vamos a instalar un visor de ficheros PDF llamado *xpdf*, así que vamos a indicar los pasos a seguir:

1. Actualizamos la base de datos ejecutando desde una **xterm** y como **root**:

   ```
   # apt-get update
   ```

   Nos debe aparecer algo similar a:

   ```
   guadalinex:/home/fermin# apt-get update
   Des:1 http://http.guadalinex.org liron/main Packages [2835kB]
   Obj http://http.guadalinex.org liron/main Release
   Obj http://http.guadalinex.org liron/contrib Packages
   Obj http://http.guadalinex.org liron/contrib Release
   Obj http://http.guadalinex.org liron/non-free Packages
   Obj http://http.guadalinex.org liron/non-free Release
   Des:2 http://http.guadalinex.org liron/guada Packages [1179B]
   Des:3 http://http.guadalinex.org liron/guada Release [82B]
   Obj http://http.guadalinex.org liron/non-US/main Packages
   Obj http://http.guadalinex.org liron/non-US/main Release
   Obj http://http.guadalinex.org liron/non-US/contrib Packages
   Obj http://http.guadalinex.org liron/non-US/contrib Release
   Obj http://http.guadalinex.org liron/non-US/non-free Packages
   Obj http://http.guadalinex.org liron/non-US/non-free Release
   Descargados 2837kB en 1m48s (26,1kB/s)
   Leyendo lista de paquetes... Hecho
   ```

2. Procedemos a instalar ejecutando:

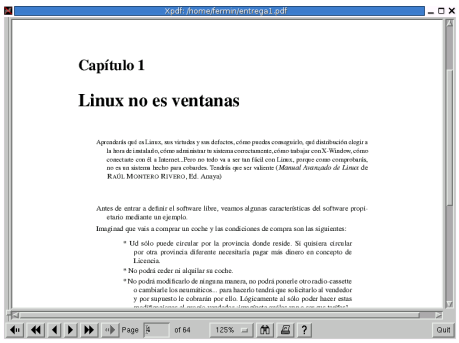
   ```
   # apt-get install xpdf
   ```

   Y nos aparecerá:

   ```
   guadalinex:/home/fermin# apt-get install xpdf
   Leyendo lista de paquetes... Hecho
   Creando árbol de dependencias... Hecho
   Se instalarán los siguientes paquetes extras:
   lesstif2 t1lib1 xpdf-common xpdf-reader xpdf-utils
   Se instalarán los siguientes paquetes NUEVOS:
   lesstif2 t1lib1 xpdf xpdf-common xpdf-reader xpdf-utils
   0 actualizados, 6 se instalarán, 0 para eliminar y 54 no actualizados.
   Necesito descargar 2602kB de archivos.
   Se utilizarán 6746kB de espacio de disco adicional después de desempaquetar.
   ¿Desea continuar? [S/n] s
   Des:1 http://http.guadalinex.org liron/main lesstif2 1:0.93.94-1 [691kB]
   ...
   Configurando xpdf (2.03-2) ...
   ```

3. Si todo ha funcionado correctamente, vamos a comprobarlo ejecutando la aplicación[10]:

   ```
   $ xpdf
   ```



Vamos a detallar lo que ha hecho `apt-get`:

1. Ha comprobado que tenía el paquete **xpdf** en la base de datos.
2. A continuación ha detectado que necesitaba varios paquetes extra (nos los ha detallado incluso) y nos ha informado que se instalarán también dichos paquetes.
3. Nos indica el tamaño de los archivos que necesita descargar y el espacio en disco duro que necesitará despues de desempaquetarlos (por si acaso no tenemos espacio suficiente en nuestro disco duro, aunque en este caso se trata sólo de 6,7 Mb, poca cosa)
4. Nos pide conformidad para proceder a la descarga e instalación de los paquetes.
5. Finalmente tras haberle dado la conformidad, se conecta a los "*sites*" indicados en el `sources.list`, descarga los paquetes, los desempaqueta y configura. Ya sólo nos queda, en este caso, ejecutar la aplicación.

[10] Ya no es necesario actuar como root (#) ahora lo podemos hacer como un usuario normal del sistema ($)

Los paquetes descargados son almacenados en el directorio `/var/cache/apt/archives` por si los necesitamos en algún otro momento o sencillamente deseamos instalarlo en otro ordenador; ya no tenemos por qué descargarlos de nuevo.

Puede ocurrir que se haya dañado el paquete instalado o sencillamente deseamos reinstalar una nueva versión disponible del mismo, entonces deberemos añadir la opción `--reinstall`

```
# apt-get --reinstall install xpdf
```

4.1.5. Eliminando paquetes

Si ya no necesitamos utilizar un determinado paquete, bien sea porque no lo utilizamos o porque no nos gusta, podemos eliminarlo del sistema utilizando apt. Para realizar esta tarea escribiremos, recuerda, en una xterm y como root: `#apt-get remove nombre_paquete`

Vamos a ponerlo en práctica con el ejemplo del visor **xpdf**. Si sencillamente ejecutamos:

```
# apt-get remove xpdf
```

Este sería el proceso normal para desinstalar un paquete, pero si probamos a ejecutar `$ xpdf` nos damos cuenta que la aplicación sigue estando instalada. ¿Qué ha sucedido? Sencillamente que no hemos desintalado el paquete que contiene el binario que ejecuta la aplicación. Vamos a localizar el binario y eliminar la aplicación completamente.

Si escribimos

```
# which xpdf
```

nos dirá exactamente dónde se encuentra el binario:

```
/usr/bin/xpdf
```

Ahora, para averiguar el nombre del paquete que realmente debemos indicar en `#apt-get remove nombre_paquete,` ejecutamos

```
# dpkg -S /usr/bin/xpdf
xpdf-reader: /usr/bin/xpdf
```

Por tanto lo que debemos hacer para eliminar el **xpdf** de nuestro ordenador sería:

```
# apt-get remove xpdf-reader
guadalinex:/home/fermin# apt-get remove xpdf-reader
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Los siguientes paquetes se ELIMINARÁN:
xpdf xpdf-reader
0 actualizados, 0 se instalarán, 2 para eliminar y 54 no actualizados.
Necesito descargar 0B de archivos.
Se liberarán 1586kB después de desempaquetar.
¿Desea continuar? [S/n] s
(Leyendo la base de datos ...
82692 ficheros y directorios instalados actualmente.)
Desinstalando xpdf ...
Desinstalando xpdf-reader ...
```

Si ahora intentamos ejecutar:

```
$ xpdf
```

Veremos que ya no funciona, lo hemos eliminado completamente; aunque ojo, no hemos borrado los paquetes de instalación. Estos siguen estando en `/var/cache/apt/archives` por si deseamos volver a instalar de nuevo.

Ejecutando `apt-get` como en el ejemplo eliminaremos los paquetes, pero sus archivos de configuración, si es que existían, permanecerán intactos en el sistema. Para una eliminación completa del paquete debríamos ejecutar:

```
# apt-get --purge remove xpdf-reader
```

Para eliminar paquetes también lo podemos hacer añadiendo un "`-`" a continuación del nombre del paquete a eliminar `apt-get install nombre_paquete-`

Al añadir "`-`" le estamos indicando que deseamos eliminarla

```
# apt-get install xpdf-reader-
guadalinex:/home/fermin# apt-get install xpdf-reader-
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Los siguientes paquetes se ELIMINARÁN:
xpdf xpdf-reader
0 actualizados, 0 se instalarán, 2 para eliminar y 54 no actualizados.
Necesito descargar 0B de archivos. Se liberarán 1586kB después de desempaquetar.
¿Desea continuar? [S/n] s
(Leyendo la base de datos ... 82692 ficheros y directorios instalados actualmente.)
Desinstalando xpdf ... Desinstalando xpdf-reader ...
```

4.1.6. Actualizando paquetes

Imaginemos que el paquete xpdf, que ya tenemos instalado en nuestro ordenador, está dañado (no funciona) o simplemente deseamos instalar una nueva versión de este paquete; utlizaremos la opción `--reinstall`

```
# apt-get --reinstall install xpdf
```

4.1.7. Actualizando a una nueva versión

Para actualizar todos los paquetes que tenemos instalados en nuestro sistema tenemos dos opciones:

1. `# apt-get upgrade`
2. `# apt-get dist-upgrade`

La diferencia entre ambos es que con `upgrade` se actualizará el sistema pero no se instalará un paquete nuevo, ni se eliminará uno ya instalado ni se actualizará un paquete que presente conflictos con otro ya instalado. Sin embargo, si usamos `dist-upgrade` realizamos una actualización completa, es decir, una vez determinado el mejor conjunto de paquetes para actualizar el sistema lo máximo posible, se instalan, actualizan y eliminan todos los que sean necesarios.

Resumiendo, si deseamos tener nuestro ordenador a la "última" deberemos optar por la segunda opción (`dist-upgrade`) ya que nos lo va actualizar todo y nos va a instalar lo último que haya disponible en este momento. Si deseamos actualizar el sistema de forma más "conservadora" optaremos por la 1ª.

Por ejemplo:

```
guadalinex:/home/fermin# apt-get dist-upgrade
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Calculando la actualización... Listo
Se instalarán los siguientes paquetes NUEVOS:
libiw27 libxslt1.1
Se actualizarán los siguientes paquetes
abcde calctool cdda2wav cdrecord cpp-3.3 cupsys cupsys-bsd cupsys-client debconf
debconf-i18n dia dia-common g++-3.3 gaim gcc-3.3 gcc-3.3-base gimp1.3 gimp1.3-data
gnome-control-center gnome-terminal hdparm iptables libbrlapi libcupsimage2 libcupsys2
libdb4.1 libgcc1 libgimp1.3 libgucharmap3 libieee1284-3 libnewt0.51 libpam-modules
libpam-runtime libpam0g libpcre3 libreadline4 libscrollkeeper0 libsdl-ttf2.0-0 libstdc++5
libstdc++5-3.3-dev libxslt1 mkisofs mozilla-locale-es-es python-newt scrollkeeper sox
synaptic-debtags timidity vim vim-gtk whiptail wireless-tools xml-core yelp
54 actualizados, 2 se instalarán, 0 para eliminar y 0 no actualizados.
Necesito descargar 39,0MB de archivos.
Se utilizarán 1483kB de espacio de disco adicional después de desempaquetar.
¿Desea continuar? [S/n]
```

Como podemos ver de la salida de ejemplo del comando, con esta opción realizamos una actualización completa ya que se consultan todas las posibles dependencias y conflictos que puedan surgir resolviéndolas de la manera más adecuada.

Imaginemos que estamos actualizando nuestro sistema con un cd-rom. APT siempre buscará la versión más reciente de los paquetes y, por tanto, si en nuestro archivo `/etc/apt/sources.list` se encontrase con alguna otra fuente que tuviese una versión del paquete más reciente que la del cd-rom, se descargaría esta nueva versión.

4.1.8. Eliminando archivos de paquete no utilizados

Los paquetes que se instalan en nuestro sistema se bajan previamente a un repositorio[11] de paquetes desde el que son instalados automáticamente por APT, sin que nosotros debamos hacer nada especial para que esto ocurra. Sin embargo, con el paso del tiempo, este proceso hace que el repositorio empiece a crecer y vaya ocupando mucho espacio en nuestro disco duro.

Para borrar los paquetes después de haber actualizado por completo nuestro sistema podemos ejecutar:

```
# apt-get clean
```

De esta forma se elimina la totalidad de paquetes de la caché.

Pero también podemos optar por:

```
# apt-get autoclean
```

De este modo, sólo se eliminarían los paquetes “inútiles”, es decir, los que ya no sirven porque existe una nueva versión de los mismos.

Ojo, tanto la opción `clean` como `autoclean` no dan opción a elegir `[s/n]`, directamente se ejecutan y eliminan.

4.2. dpkg - Introducción

El programa `dpkg` es la base del sistema de gestión de paquetes de Debian. Fue creado por Ian Jackson en 1993; es similar a `rpm`[12]. Se utiliza para instalar y desinstalar paquetes “*.deb*” pero también resulta muy útil para obtener información sobre los paquetes instalados.

Utilizaremos `dpkg`, sobre todo, cuando trabajemos con archivos locales[13]. No es imprescindible ya que `apt` suple la gran mayoría de las tareas con gran facilidad, como hemos visto en el capítulo anterior. Indicaremos brevemente su uso más habitual, aunque si deseamos conocer más podemos ver las páginas man del programa[14]. Trabajaremos desde la carpeta donde tengamos los paquetes .deb previamente descargados, o copiados, o el cd-rom donde se encuentren los mismos.

4.2.1. Instalar paquetes

Para instalar paquetes sólo debemos ejecutar[15]:

```
# dpkg -i paquete.deb
```

Siendo el “`paquete.deb`” el que previamente tenemos en nuestro ordenador (HD, cd-rom, USB disk...)

[11] Recuerda que los paquetes descargados son almacenados en el directorio `/var/cache/apt/archives`

[12] *RPM Package Manager* (o RPM, originalmente llamado *Red Hat Package Manager*) es un sistema de administración de paquetes pensado para Linux. Es capaz de instalar, actualizar, desinstalar, verificar y solicitar programas. RPM es el formato de paquete de partida del Linux Standard Base.

Originalmente fue desarrollado por Red Hat para Red Hat Linux.

[13] Por ejemplo si no tenemos acceso a internet en la máquina en la que estamos trabajando.

[14] Recuerda: `$ man dpkg`

[15] “-i” del inglés *install* (instalar)

```
guadalinex:/var/cache/apt/archives# dpkg -i xpdf-reader_2.03-2_i386.deb
Seleccionando el paquete xpdf-reader previamente no seleccionado.
(Leyendo la base de datos ... 82680 ficheros y directorios instalados actualmente.)
Desempaquetando xpdf-reader (de xpdf-reader_2.03-2_i386.deb) ...
Configurando xpdf-reader (2.03-2) ...
guadalinex:/var/cache/apt/archives#
```

Si probamos con `$ xpdf` veremos que funciona correctamente.

4.2.2. Desinstalar paquetes

Para proceder a la desintalación de paquetes ejecutamos[16]:

```
# dpkg -r paquete.deb
guadalinex:/var/cache/apt/archives# dpkg -r xpdf-reader
(Leyendo la base de datos ... 82691 ficheros y directorios instalados actualmente.)
Desinstalando xpdf-reader ...
```

Si intentamos `$ xpdf` veremos que ya no funciona.

Si deseamos que, además de desinstalar, borrre también los ficheros de configuración

```
# dpkg -purge paquete.deb
```

4.2.3. Opciones útiles

Podemos utilizar a `dpkg` para:

dpkg l lista los paquetes instalados en nuestro sistema.

dpkg -l nombre lista la información que tiene de ese paquete.

```
guadalinex:~# dpkg -l xpdf
Desired=Unknown/Install/Remove/Purge/Hold
| Estado=No/Instalado/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: mayúsc.=malo)
||/ Nombre Versión Descripción
+++-==============================================
ii  xpdf  2.03-2  Portable Document Format (PDF) suite
```

dpkg -L nombre lista los directorios donde ha instalado archivos el paquete indicado.

```
guadalinex:~# dpkg -L xpdf
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/xpdf
```

dkpg -s nombre lista información de los paquetes instalados, estado y dependencias.

```
guadalinex:~# dpkg -s xpdf
Package: xpdf
Status: install ok installed
Priority: optional Section: text Installed-Size: 6
Maintainer: Hamish Moffatt <hamish@debian.org>
Version: 2.03-2 Replaces: xpdf-i (<= 0.90-8)
Depends: xpdf-reader (= 2.03-2), xpdf-utils (= 2.03-2), xpdf-common (= 2.03-2)
Conflicts: xpdf-i (<= 0.90-8)
Description: Portable Document Format (PDF) suite xpdf is a suite of tools for Portable Document
Format (PDF) files. (These are sometimes called 'Acrobat' files after the name of Adobe's PDF software.)
. The tools include xpdf, a PDF viewer (in the package xpdf-reader), and PDF converters (including
to/from PostScript) (in the package xpdf-utils). . This package is intended for compatibility with
previous versions of this package only. You can safely remove it from your system.
```

[16] "-r" del inglés: *remove* (suprimir, eliminar)

dpkg -S nombre busca un nombre de fichero, correspondiente al patrón de búsqueda que le indicamos, en los paquetes instalados.

```
guadalinex:~# dpkg -S xpdf
xpdf xpdf-common: /usr/share/xpdf/turkish
xpdf-common: /etc/xpdf/xpdfrc

...
xpdf-reader: /usr/bin/xpdf.bin
```

4.3. synaptic

¡Muy bonito! ¿verdad? Dejamos para el final el entorno gráfico ¡eh!

Cierto, pero es preferible saber cómo funciona algo en profundidad para que, más tarde, cuando aparecen las "*ventanitas*" ya todo nos suene y sólo entonces sí que es realmente sencillo.

Aunque cuando finalice el capítulo os daréis cuenta que las herramientas gráficas no tienen la potencia de los comandos.

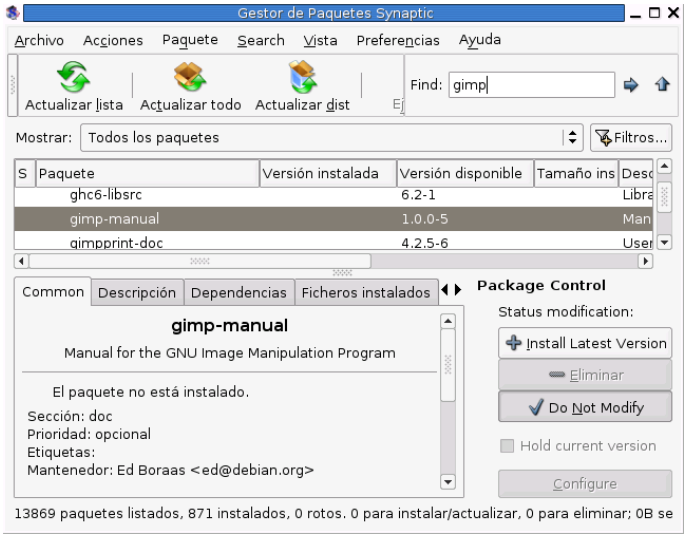
Empecemos pués, `synaptic` es una herramienta en modo gráfico para la instalación, desinstalación de paquetes y mantenimiento en general de nuestra distribución.

Para trabajar con ella podemos dirigirnos directamente a:

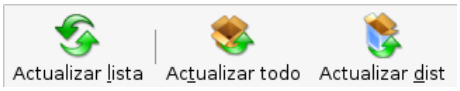
→ **Aplicaciones → Herramientas de sistema → Panel de Control → Synaptic (gestor de paquetes)**

O bien desde una **xterm** y como **root** ejecutando:

```
# synaptic &
```



Todo lo que se ha explicado en los apartados anteriores de este capítulo tiene su reflejo en este entorno gráfico. Demos un repaso a las principales equivalencias:

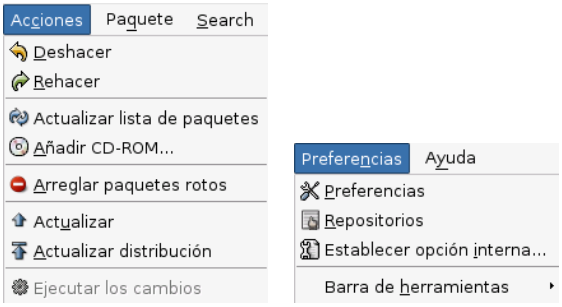


Actualizar lista: `# apt-get update`

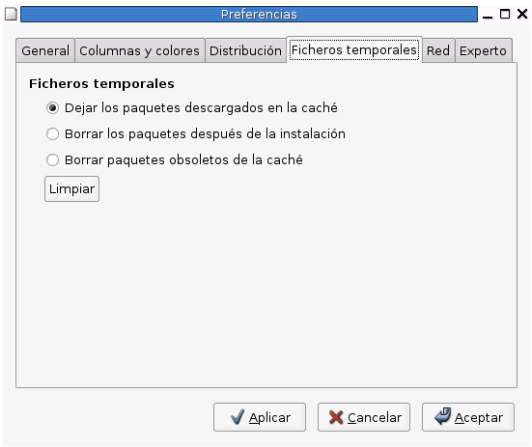
Actualizarlo todo: `# apt-get upgrade`

Actualizar dist: `# apt-get dist-upgrade`

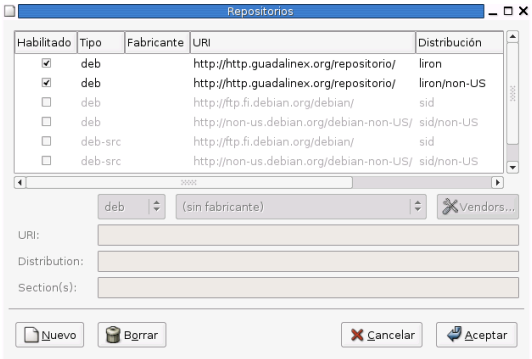
El menú [**Acciones**] no presenta dificultad. Desde [**Preferencias**] podemos:



- Ajustar las preferencias de `synaptic` tanto en aspectos visuales (colores, orgranización gráfica ...) como organizativos (proxy, acciones con los paquetes una vez descargados ...)

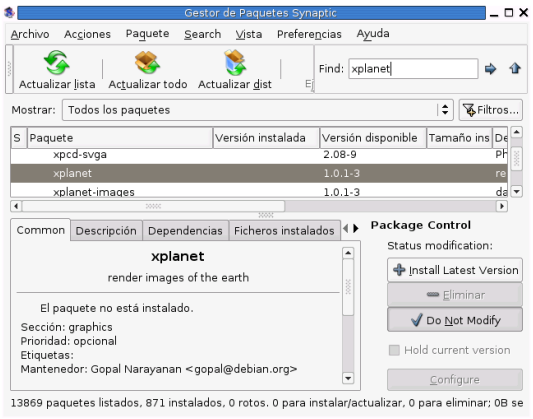


- Configurar los repositorios, es decir, configurar el archivo `/etc/sources.list`



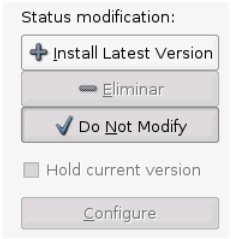
➥**Para practicar:** Vamos a ver cómo se instalaría un programa con `synaptic`:

1. En la casilla **Find**[17] le indicaremos el nombre de la aplicación que deseamos instalar; en este caso vamos a elegir un programa de astronomía **xplanet**.
2. Una vez que lo ha encontrado,



observamos en las pestañas de la parte inferior, que el paquete no está instalado; nos muestra una descripción del mismo; las depedencias y los ficheros que va a instalar y dónde.

3. Para proceder a la instalación del paquete, en la parte inferior derecha, disponemos:

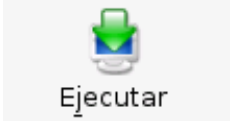


y con ellos podemos:

- instalar la última versión disponible del paquete (si es que ya lo tuviésemos instalado y sólo deseásemos actualizarlo)
- eliminar el paquete (sólo si ya está instalado, obviamente)

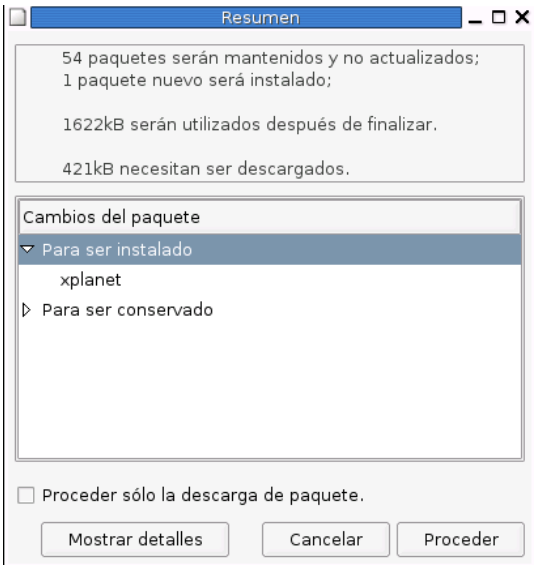
Si optamos por instalar la aplicación, [**Install Lastest Version**] podemos seguir eligiendo paquetes a instalar utilizando el mismo procedimiento. No es necesario marcar un paquete e instalarlo. Se pueden marcar diferentes y más tarde instalarlos todos juntos.

4. Una vez que hayamos decidido que ya no vamos a instalar más paquetes haremos “clic” en:



5. Y nos aparecerá una ventana informándonos de los cambios que va a realizar:

[17] del inglés “buscar”



Podemos marcar una casilla de verificación para que sólo descargue el paquete y que no lo instale.

Si se trata de desinstalar paquetes se realizaría el mismo proceso pero, en lugar de marcar [**+ Install Lastest Version**], procederíamos a marcar [**- Eliminar**] ya que el resto de pasos a seguir serían idénticos.

4.4. Otros

Las herramientas más comunes para el mantenimiento de nuestro GuadaLinex son las ya comentadas con anterioridad: `apt-get`, `dpkg` y `synaptic`.

Pero además de éstas, existen otras como son:

- `dselect`
- `aptitude`

Ambas se ejecutan en modo comando desde una **xterm** y, como es lógico, actuando como **root**.

Para ampliar información sobre estos comandos podéis ver las páginas man de ambos:

```
$ man dselect
$ man aptitude
```