Software Libre y Educación:

Guadalinex (Debian) y aplicaciones didácticas

La Shell Bash. Redes (Introducción)

Juan Alonso - Fermín Rubio - Paco Villegas

26 de abril de 2005

Índice general

1. La Shell Bash **3**
- 1.1. La Shell Bash . . . 3
  - 1.1.1. ¿Qué es una shell? . . . 3
  - 1.1.2. Características básicas de la Shell. . . . 3
  - 1.1.3. Variables de entorno de la Bash . . . 4
  - 1.1.4. Ficheros de inicio de la bash . . . 5
  - 1.1.5. Personalizando el Prompt . . . 6
  - 1.1.6. Los Alias . . . 7
  - 1.1.7. Historia de órdenes. . . . 8
  - 1.1.8. Los Builtins (Órdenes internas) . . . 9
- 1.2. Redirección . . . 9
  - 1.2.1. Redirección de la salida (>) . . . 10
  - 1.2.2. Redirección de la entrada (<) . . . 10
  - 1.2.3. Tuberías . . . 11
- 1.3. Comandos de la Shell . . . 11
  - 1.3.1. Comandos simples . . . 11
  - 1.3.2. Listas de comandos . . . 11

2. Comandos básicos de Unix/Linux **13**
- 2.1. Introducción . . . 13
  - 2.1.1. Convenciones en cuanto a la sintaxis . . . 15
  - 2.1.2. Comodines . . . 15
- 2.2. Resumen de comandos . . . 16
  - 2.2.1. Ayuda . . . 16
  - 2.2.2. "Construir" comandos . . . 16
  - 2.2.3. Gestión de usuarios y grupos . . . 16
  - 2.2.4. Manipulación de archivos y directorios . . . 16
  - 2.2.5. Localización de archivos . . . 17
  - 2.2.6. Procesamiento de archivos . . . 17
  - 2.2.7. Guardar y comprimir ficheros . . . 17
  - 2.2.8. Procesos de control . . . 18
  - 2.2.9. Control de usuarios . . . 18
  - 2.2.10. Administrar ficheros . . . 18
  - 2.2.11. Comunicaciones y redes . . . 18
  - 2.2.12. Comandos de Impresión . . . 19
  - 2.2.13. Módulos del kernel . . . 19
  - 2.2.14. Varios . . . 19
- 2.3. Algunos ejemplos . . . 19
  - 2.3.1. "Construir" comandos . . . 19
  - 2.3.2. Manipulación de archivos y directorios . . . 20
  - 2.3.3. Localización de archivos . . . 23
  - 2.3.4. Procesamiento de archivos . . . 25

2.3.5. Empaquetar y comprimir ficheros. 26
2.3.6. Control de tareas 29
2.3.7. Administrar ficheros 35
2.3.8. Comunicaciones y redes. 36

3. Introducción a las redes **39**
3.1. Introducción 39
3.2. Redes TCP/IP: conceptos básicos 40
3.2.1. Protocolos de Red 40
3.2.2. Introducción a las direcciones IP. 41
3.3. Guadalinex en una red IP 43
3.3.1. Configuración del interfaz de red 44
3.3.2. Configuración gráfica. 49
3.3.3. Configuración: servidores y servicios de red 52
3.3.4. Gnome-netinfo 54

4. En Red-ando con Guadalinex **56**
4.1. Servicio http 56
4.1.1. Como cliente 57
4.1.2. Como servidor: Apache 58
4.2. Telnet y ssh 60
4.2.1. Acceso remoto: telnet 60
4.2.2. SSH 62
4.3. FTP y SFTP 64
4.3.1. ftp 64
4.3.2. sftp 66
4.3.3. gFTP 66
4.4. Cortafuegos 71
4.5. Samba 72
4.5.1. Configuración 73
4.5.2. Swat 77
4.5.3. A “bailar” la Samba 79

Capítulo 1

La Shell Bash

> Tras la instalación de todos los programas educativos preguntaremos una y otra vez " TODO ESTÁ EN EL MENÚ, ¿NO?". Haremos eso cada vez que el pringao nos intente explicar qué son ficheros, carpetas y chorradas de esas. Si nos intenta enseñar una ventana negra en la que hay que ¡¡ESCRIBIR!! (sí, amigos, en pleno siglo 21 hay que escribir cosas) y además ¡¡EN INGLÉS!! le diremos que no lo entendemos y que nos ponga eso en el menú. Si nos dice que no se puede poner eso en el menú haremos referencia a que creíamos que él sabía más de informática... (*Pringao Howto (o Windows-es-fácil-Howto)*, Santiago Romero aka NoP/Compiler)

1.1. La Shell Bash

1.1.1. ¿Qué es una shell?

El nombre de bash viene de *Bourne Again Shell*[1]. Bash está pensado con la intención de ser una implementación conforme con la especificación POSIX de Shell y Herramientas, de la IEEE (Grupo de Trabajo 1003.2 de la IEEE).

Ya que hemos mencionado POSIX, debemos decir que es un estándar[2] (normas escritas en papel, para que lo veamos de una forma más práctica) que pretende definir un Sistema Operativo Abierto[3] definido por la IEEE (que es una organización internacional de estándares). Linux intenta cumplir con los estándares POSIX.

Para poder estudiar más sobre la shell bash:

- Los howtos traducidos en el INSFLUG.
- La completa página man de la shell Bash[4]
- Bash Reference Manual `http://www.gnu.org/software/bash/manual/bashref.html`

1.1.2. Características básicas de la Shell.

El funcionamiento de la shell es el siguiente:

1. Lee la entrada desde teclado o desde un fichero.

[1] Algo así como la shell Bourne viene de nuevo, aunque una traducción más libre quedaría como "la Bourne Shell Contraataca".

[2] Los estándares de derecho (de iure), emitidos por organismos independientes y reconocidos, son importantes porque permiten la independencia de un determinado fabricante y fomentan la interoperabilidad de distintos sistemas. Un "estándar" de hecho (de facto), simplemente puede reconocer el monopolio de un fabricante.

[3] Abierto en el sentido de no perteneciente a ninguna empresa o grupo y con unas reglas claras para poder operar con él.

[4] $ man bash

2. Divide la entrada en palabras y operadores, obteniendo los comandos.
3. Realiza las expansiones correspondientes y las redirecciones de salida.
4. Ejecuta la o las órdenes.
5. Espera (opcionalmente) a que terminen las órdenes y devuelve un valor de estado de finalización. El valor de estado 0 (cero) significa finalización sin errores y un valor distinto de cero indica el código de error producido.

1.1.3. Variables de entorno de la Bash

La shell utiliza las variables de entorno para *afinar* ciertos detalles del comportamiento del sistema. Algunas de estas variables de entorno, ya predefinidas, que utiliza bash son:

HOME El directorio de comienzo del usuario.

PATH Una lista de directorios separados cada uno de ellos por el carácter dos puntos (:) que nos indica en qué directorios busca la shell para encontrar los comandos. Escoge el comando que primero encuentre, en caso de que pueda encontrarse en varios sitios. Si no lo encuentra dentro de esta lista de directorios, nos devolverá un error con el mensaje "Comando no encontrado" o "command not found".

PS1 El prompt (o indicador de inicio) que presenta la bash al usuario.

PWD El directorio de trabajo actual.

Para ver el contenido de una variable basta con teclear:

```
$ echo $nombre_var
```

➥ **Para practicar:**

Comprobar el valor de cada una de las variables anteriores. Por ejemplo, el valor de la variable `PATH` en mi máquina y para el usuario que ejecuta el comando es:

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
```

■

Podemos también definir nuestras propias variables mediante la orden:

```
nombre_var=valor
```

Por ejemplo:

```
$ miedad=14
$ minombre="Harry Potter"
```

define dos variables cuyos contenidos son explícitos y podemos visualizarlo con el comando `echo`.

Si pusiéramos como valor de la variable un comando, por ejemplo `ls`

```
$ listado=ls
```

podríamos invocarlo de la siguiente forma:

```
$ $listado
```

[5]

[5] El primer símbolo de $ es el prompt y el segundo sirve para obtener el valor de la variable. Observar la diferencia entre la orden anterior y esta otra $ echo $listado.

En cualquier momento podemos ver el valor de todas las variables de entorno definidas en nuestra shell con el comando `set`.

Con `export nombre_var` (exportamos la variable para que sea visible en esta shell y todos los procesos hijos[6] de esta shell).

➥ **Para practicar:**

Si has definido las variables anteriores, ejecuta

```
$echo "me llamo " $minombre " y tengo " $miedad " años"
$set|less
```
■

1.1.4. Ficheros de inicio de la bash

En el inicio, dependiendo de la shell con que entre el usuario al sistema, se ejecutan una serie de ficheros que le configuran su entorno de trabajo. Existen unos ficheros generales que se ejecutan para todos los usuarios que entran al sistema con una misma shell (como por ejemplo el `/etc/profile` para las shell Bourne y Korn), y otros específicos para cada usuario y que se encuentran en su directorio HOME. Estos ficheros de inicialización son utilizados para establecer el camino de búsqueda de ficheros ejecutables, establecer protección por defecto de los ficheros que se creen, tipo de terminal desde el que se trabaja y otras variables de entorno. Algunos de estos ficheros son:

`/etc/profile` en él se configuran algunas variables de entorno y otros parámetros para todos los usuarios del sistema. Es del root. Se lee una sola vez cuando se inicia el sistema y, dependiendo de la distribución, en él se establecen:

- el prompt por defecto
- el path por defecto
- el tamaño máximo de los ficheros que podemos crear
- los permisos por defecto para los ficheros que creemos.
- tamaño de los ficheros de historial
- ...

`~/.bash_profile` permite introducir información específica para cada usuario[7]. Se lee sólo una vez cuando el usuario accede en el sistema. En él hay una llamada que hace que se ejecute `.bashrc`. En la configuración por defecto de Guadalinex2004 esta llamada está comentada. Sugerimos que se edite el fichero .bash_profile y se descomente esa llamada (como aquí).

```
# include .bashrc if it exists
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi
```

`~/.bashrc` información/configuración específica de un usuario para la shell bash. Puede modificar los valores que se cargaron para el conjunto de usuarios. Su contenido se lee cada vez que se entra en el sistema y cada vez que se abre una nueva shell bash[8].

[6] Ya hablaremos sobre los procesos. Por ahora, sepamos que la shell ejecuta los comandos que le introducimos como procesos hijos. La shell se encarga de que nazcan, realicen su tarea y mueran cuando finalicen.

[7] El símbolo ~ hace referencia al home de usuario. Por ejemplo, si se trata del usuario Thales ~ se sustituye por `/home/Thales`

[8] las modificaciones de las *variables de entorno* y los *alias* (hablaremos de ellos un poco mas alante) deben establecerse en este fichero.

Cuando la bash es llamada como una shell interactiva[9] de comienzo, lo primero que hace es leer y ejecutar los comandos que se encuentran en el fichero `/etc/profile`. Después pasa al fichero `~/.bash_profile`. Cuando se trata de una shell interactiva pero que no es de comienzo[10], el fichero que ejecuta es `~/.bashrc`[11].

El fichero `/etc/profile`, como hemos comentado antes, se encarga de que tengamos el entorno listo para trabajar, se ejecuta al entrar cualquier usuario del sistema y es modificable sólo por el superusuario, mientras que los que se encuentran bajo el directorio `HOME` (~) de cada usuario son configurables y personalizables por éstos. El contenido del fichero /etc/profile en Guadalinex2004 es el siguiente:

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

#PATH="/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games"

if [ "$PS1" ]; then
  if [ "$BASH" ]; then
    PS1='\u@\h:\w\$ '
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

#export PATH


# Añadido por /usr/bin/eurocastellanizar
if [ -f /etc/language-euro-es ]; then source /etc/language-euro-es; fi
# fin cambios
umask 022
```

Listado 1.1: /etc/profile

1.1.5. Personalizando el Prompt

Como vimos antes, el valor de la variable PS1 determina lo que se nos presenta en el prompt del sistema. Si se asigna un valor a la variable PS1 en el fichero `/etc/bash.bashrc` , éste será establecido para todos los usuarios y sobreescribe entonces el valor definido en `/etc/profile`. Cada usuario puede modificar su prompt el el fichero `.bashrc`.

El valor predeterminado es `PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$`[12] ', que podemos ver ejecutando el comando

```
$ echo $PS1
```

[9] La que vemos normalmente y le introducimos comandos. Cuando termina el comando, nos presenta otra vez el símbolo del sistema para continuar.

[10] por ejemplo al abrir una `xterm` en X-Windows

[11] el proceso de inicio es como se ha descrito si se entra en modo consola. Cuando se inicia una sesión gráfica el proceso es algo más complejo y depende del gestor de escritorio y del administrador de ventanas. En el caso de Gnome, gdm configura el administrador de escritorio y el idioma, además la mayoría de las características se pueden configurar desde el Centro de control de Gnome. Para ampliar sobre esto podéis mirar en: `http://lucas.ok.cl/Manuales-LuCAS/AA_Linux_colegio-1.1/AA_Linux_colegio-1.1-html/x4475.htm`

[12] Sustituye `${variable+palabra}` por el valor de `palabra` si `variable` está configurado, si no se ignora.

Existen algunos valores predeterminados que podemos utilizar para modificar el prompt a nuestro antojo.

\d la fecha en el formato "Día-Semana Mes Día" (ejemplo, "Tue May 26") en inglés

\e un carácter de escape (ESC) ASCII (033)

\h el nombre del computador hasta el primer '.'

\H el nombre del computador con dominio completo

\n salto de línea

\r retorno de carro

\s el nombre del shell. El nombre base del ejecutable de la shell (la porción que sigue a la última barra inclinada)

\t la hora actual en el formato de 24 horas `HH:MM:SS`

\T la hora actual en el formato de 12 horas `HH:MM:SS`

\@ la hora actual en el formato de 12 horas con indicador AM/PM

\u el nombre de usuario del usuario actual

\v la versión de bash (e.g., 2.00)

\V la distribución de bash, versión + nivel de parches (e.g., 2.00.0)

\w el directorio de trabajo en curso

\W el nombre base del directorio de trabajo

\! el número de historia de esta orden

\# el número de orden de este comando en la shell actual

\$ si el UID[13] efectivo es 0 (el super-usuario root), un `#`. Si no lo es, un `$`

\nnn el carácter correspondiente al número octal `nnn`

\\ una barra inclinada invertida

\[empieza una secuencia de caracteres no imprimibles, que pueden emplearse para insertar una secuencia de control del terminal en el indicador, por ejemplo para cambiar el color del prompt

\] termina una secuencia de caracteres no imprimibles

➥ **Para practicar:**

Entra al sistema como usuario de a pié, abre un terminal gráfico y ejecuta

- `$ PS1="[\t] \u@\h: \w\$ "`
  Observa que el prompt ha cambiado
- `$ echo $PS1`
- haz un `su` a root y observa como el prompt es distinto
- la razón está aquí, ejecuta `# echo $PS1`[14]
- Comenta la linea `PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '` del fichero `/etc/bash.bashrc` y reinicia el sistema
  Abre un terminal gráfico y observa la diferencia en el prompt al hacer un `su` a otro usuario y `su` a root.[15] ■

Para ampliar sobre este tema se puede consultar el HOWTO: *Bash Prompt COMO*.

1.1.6. Los Alias

Un *alias* es "un nombre corto" para un comando "largo" (generalmente un comando complejo). Con ello conseguimos economía de escritura, pues cuando el "nombre corto" se utiliza como primera palabra de un comando simple, en la ejecución es sustituido por el "largo".

[13] Identificador de usuario.
[14] el prompt para todos está definido en `/etc/bash.bashrc`
[15] Ahora para el root el prompt por defecto es el definido en `/etc/profile` y para el otro usuario el definido en su `.bashrc`.

Los alias se crean y muestran con la orden `alias`, y se quitan con la orden `unalias`.
La sintaxis para definirlos es[16]:

```
alias [-p] [nombre[=valor] ...]
```

Por ejemplo, con

```
$ alias ll="ls -laF"
```

definimos el alias `ll` y conseguiremos con sólo dos caracteres ("ll", mnemónico de Listado Largo) realizar la misma función que con siete (`ls -laF`).

Para eliminar un alias utilizamos

```
unalias [-a] [name ... ]
```

así,

```
$ unalias ll
```

elimina el alias creado anteriormente.

La shell mantiene en memoria una lista de los alias definidos que podemos visualizar con la orden `alias`.

Cuando se ejecuta una orden la shell mira si la primera palabra, si no está entrecomillada, tiene un alias. Si es así, la palabra se reemplaza con el texto del alias. El nombre del alias y el texto por el que se reemplaza, pueden contener cualquier entrada válida para el shell, incluyendo metacaracteres, con la excepción de que el nombre del alias no puede contener un `=`. La primera palabra del texto de reemplazo se comprueba también para ver si es un alias, pero si es un alias idéntico al que se está expandiendo, no se expande una segunda vez; esto significa que uno puede poner un alias "`ls`" a "`ls -F`", por ejemplo, y bash no intenta expandir recursivamente el texto de reemplazo. Si el último carácter del valor del alias es un blanco, entonces la siguiente palabra de la orden que sigue al alias también se mira para la expansión de alias. No hay ningún mecanismo para poder usar argumentos en el texto de reemplazo[17].

Si queremos definir un alias de forma permanente tendremos que hacerlo en el fichero `~/.bashrc`, de lo contrario se borrará de la memoria cuando salgamos del sistema.

➥ **Para practicar:**

Como ya habremos comprobado, para movernos al directorio padre del actual se utiliza el comando `cd ..` (el espacio es necesario), mientras que los "mayores" recordamos que en DOS el comando equivalente era `cd..`
Crear un alias que permita que funcione el comando `cd..`

```
$ alias cd..="cd .."
```

Probar que funciona y eliminarlo después con

```
$ unalias cd..
```

■

1.1.7. Historia de órdenes.

Cuando se habilita la opción `-o history` (opción que ya está normalmente por defecto[18]), el shell da acceso a la historia de órdenes: lista de órdenes tecleadas con anterioridad.

El texto de los últimos mandatos se guarda en una lista de historia. La shell almacena cada orden en la lista de historia antes de la expansión de parámetros y variables (el número de órdenes almacenadas en la lista se define en la variable `HISTSIZE`, por omisión 500) . En el arranque, la historia se inicia a partir del fichero nombrado en la variable `HISTFILE` (por omisión `~/.bash_history`[19]). `HISTFILE` se trunca, si es necesario, para contener no más de `HISTFILESIZE` líneas.

[16] Los corchetes indican que los parámetros son opcionales y no se tienen que escribir.
[17] Si se necesitan, debería emplearse mejor una función del shell.
[18] Podemos ponerla con `set -o history`
[19] Podemos ver con un editor el contenido de este fichero

➥ **Para practicar:**

- Comprobar los valores (por defecto) de las variables anteriores con

  `$ echo $HISTSIZE`

  `$ echo $HISTFILE`

- Para visualizar la lista:

  `$ history`

  o mejor

  `$ history | less`

  Podemos recorrer las órdenes anteriores con las flechas del cursor; `q` para salir

- Tras la salida del comando

  `$ history`

  ejecutar
  `$ !comando_número_línea`

- Ejecutar el último comando echo con

  `$ !e` ■

1.1.8. Los Builtins (Órdenes internas)

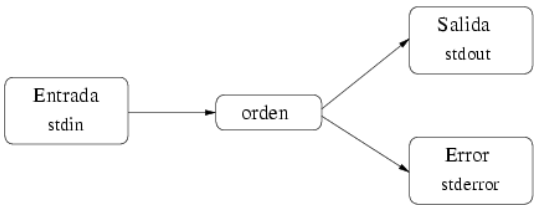
Los *builtins* (u órdenes internas) son comandos que ya vienen implementados dentro de la propia bash. No hay que buscar un ejecutable externo porque la propia bash lo lleva incorporado. Por ello, se ejecutan mucho más rápido. Algunos de ellos son:

cd que como ya sabemos nos cambia de directorio de trabajo.

pwd que nos indica en qué directorio estamos situados.

1.2. Redirección

La ejecución de un comando generalmente responde al siguiente esquema:



En la figura se observa que el comando, si necesita algún dato de entrada, lo habitual es que lo reciba a través del teclado, que es la entrada por defecto (*stdin*). Si la ejecución del comando conlleva la devolución de alguna información, esta se envía a la pantalla, que es el dispositivo de salida por defecto (*stdout*). Si se produce un error en la ejecución del comando, el mensaje correspondiente se envía por el dispositivo de errrores por defecto (*stderr*), que es también la pantalla.

Este comportamiento puede modificarse con lo que denominamos **redirección**.

1.2.1. Redirección de la salida (>)

Supongamos que deseamos guardar la salida del comandeo `dmesg` para posteriormente analizarla con tranquilidad. Para ello basta con ejecutar:

```
$ dmesg > mensajes.txt
```

Con ello, la salida que hubiera aparecido por pantalla, se ha guardado en el fichero `mensajes.txt`

Si el fichero especificado existe, se trunca a longitud cero, es decir, se borra previamente su contenido. Si no existe, se crea.

Posteriormente podremos ver el contenido del fichero.

Añadir a la salida redirigida (>>)

Como hemos comentado más arriba, la redirección de salida (>) borra previamente el contenido del fichero especificado. Si queremos añadir la salida conservando el contenido anterior del fichero, debemos utilizar el signo (>>).

Por ejemplo, el comando `df -h` devuelve información del espacio de disco ocupado en el sistema. Para hacer un seguimiento del consumo de disco podemos ejecutar periódicamente el comando que sigue y no perderemos los valores que vamos almacenando, sino que se irán acumulando en el fichero.

```
$ df -h >> consumo_disco.txt
```

1.2.2. Redirección de la entrada (<)

La redirección de la entrada hace que el comando tome como argumento de entrada el fichero especificado.

Por ejemplo, con la orden

```
$ cat < /etc/passwd
```

el comando `cat` (que muestra por salida estándar lo que recibe por la entrada estándar) recibe como entrada el fichero `/etc/passwd`[20]

➥ **Para practicar:**

Al final de cada apartado, comprobar el resultado, por ejemplo con

```
$ more prueba
```

- El comando que sigue crea un fichero de texto con ese universal saludo

  ```
  $ echo "Hola Mundo" >prueba
  ```

- Añadir al final "cruel"

  ```
  $ echo "cruel" >>prueba
  ```

- Ahora almacenamos en `prueba` los ficheros de nuestro `$HOME`, en una columna y ordenados por tiempo de creación[21]

  ```
  $ ls -t >prueba
  ```

- ¿Y si lo ordenamos alfabéticamente?

  ```
  $ sort <prueba >prueba_o
  ```

- ¿Qué pasa ahora?

  ```
  $ more prueba >>prueba
  ```

■

[20] la orden `$ cat /etc/passwd` haría el mismo efecto

[21] `$ man ls` para ver las opciones de este comando

1.2.3. Tuberías

Una tubería es una secuencia de una o más órdenes separadas por el carácter "|" (barra vertical). El formato de una tubería es:

```
orden1 [ | orden2 ... ]
```

La salida estándar de `orden1` se conecta a la entrada estándar de `orden2`. Esta conexión se realiza antes que cualquier redirección especificada por la orden.

Cada orden en una tubería se ejecuta como un proceso separado (esto es, en una subshell).

Por ejemplo, para contar el número de líneas de un fichero, ejecutaríamos:

```
$ cat fichero | wc -l
```

Su explicación es que con el comando `cat` visualizamos el contenido del fichero, pero esta salida, en vez de ir a la pantalla, se mete en la tubería que va hacia la entrada de la orden `wc` (de *word count*, contador de palabras) que con su opción `-l` nos dice el número de líneas que ha leído[22].

En esta característica se apoya gran parte de la elegancia de los sistemas Unix/Linux. Con comandos simples podemos llegar a realizar acciones verdaderamente complejas.

➥ **Para practicar**

1. La orden `sort` ordena alfabéticamente lineas de ficheros de texto.
   Crea, por ejemplo con `gedit`, un fichero de texto `alumnos.txt`, con los apellidos y nombres de un grupo de alumnos (no los introduzca ya ordenados)
   Ejecuta la orden
   `cat alumnos.txt | sort`
   Ahora ejecuta
   `cat alumnos.txt | sort >alumnos_ordenados.txt`
   y visualiza el fichero `alumnos_ordenados.txt`
2. El comando `grep` envía a la salida estándar (o a la especificada) las lineas que concuerden con un patrón.
   Ejecuta las ordenes
   `ls /etc >dir_etc`
   `cat dir_etc | grep conf` ■

1.3. Comandos de la Shell

1.3.1. Comandos simples

Un comando simple es la clase de comandos que nos encontramos más frecuentemente. Consiste en una secuencia de palabras separadas por blancos. La primera palabra especifica el comando a ejecutar, seguido por unas opciones (como por ejemplo "`ls -l`"[23]) o unos argumentos (`cat /etc/profile`[24]).

1.3.2. Listas de comandos

Una lista de comandos es una secuencia de comandos simples o tuberías separados por uno de los operadores ';', '`&`', '`&&`', or '`||`', y terminada por '`;`', '`&`', o retorno de carro.

Si un comando se termina con el operador de control '`&`', la shell ejecuta el comando de forma asíncrona en una subshell. Esto se conoce como ejecutar el comando en segundo plano (*background*). En este caso, la shell no espera a que el comando termine sino que inmediatamente

[22] la orden `$ wc -l fichero` daría el mismo resultado

[23] En este caso la opción es `l`, y el guión sirve para indicar que lo que viene detrás es una opción. Una opción normalmente modifica el comportamiento de un comando.

[24] El fichero `/etc/profile` es un argumento. Los argumentos normalmente indican sobre qué actúa el comando.

aparece otra vez el indicador de inicio (`prompt`), mientras el comando se ejecuta de manera "oculta" (en segundo plano). Por ejemplo si en un xterm ejecutamos

```
$ mozilla &
```

veremos que el programa se ejecuta en segundo plano, quedando el terminal libre por si necesitamos introducir más comandos.

Los comandos separados por '`;`' se ejecutan secuencialmente, uno detrás de otro.

```
$ comando1; comando 2
```

La shell espera a que terminen los comandos en su turno correspondiente. Por ejemplo:

```
$ cd /home/Thales; ls
```

primero se posiciona en el subdirectorio `/home/Thales` y después lista los ficheros de ese directorio.

Los operadores de control permiten ejecuciones condicionales.

El efecto de

```
$ comando1 && comando2
```

es que `comando2` se ejecutará si y sólo si `comando1` termina de forma satisfactoria (devuelve un código de cero).

En cambio, en la lista

```
$ comando1 || comando2
```

el `comando2` se ejecutará si y sólo si `comando1` falla (devuelve un código distinto de cero).

➥ **Para practicar: comando `tee`**

Podemos conseguir guardar la salida de un comando en un fichero y dirigirla también a la salida estándar usando el comando `tee`. El nombre del comando viene de que se comporta como una T de fontanería. El caudal que llega por una rama, pasa por la T y sale por los otros dos orificios.

Por ejemplo, supongamos que deseamos ver los usuarios de nuestra máquina y guardarlos en un fichero ordenados, escribiremos:

```
$ cut -f1 -d: /etc/passwd | sort | tee usuarios.txt
```

Expliquemos un poco el comando:

- `cut -f1 -d: /etc/password` →Obtiene del fichero `/etc/passwd` el primer campo (`f1` de *field1*), especificando como separador de campo (`-d` de delimitador) el carácter `:`
- `sort` → ordena alfabéticamente los nombres de usuario
- `tee usuarios.txt` →guarda el resultado en el fichero `usuarios.txt` y además lo dirige a la salida estándar. Son las dos salidas de la T ■

Capítulo 2

Comandos básicos de Unix/Linux

> Como regla general, se podría decir lo siguiente: "Todo lo que se puede hacer en modo gráfico, se puede hacer también en modo texto, a base de comandos. Pero no todo lo que se puede hacer en modo texto, se puede hacer en modo gráfico". (*FAQ sobre Linux para principiantes* - es.comp.os.linux)

2.1. Introducción

En este apartado veremos los comandos más usuales de Linux. Ni están todos ni tiene sentido ver todas y cada una de las opciones de los que expongamos. Para ampliar información os remitimos a las páginas de ayuda de cada comando, a las infopages, así como a los manuales comentados en la primera entrega.

A la pregunta ¿es necesario conocer los comandos? la respuesta es clara: sí, **al menos los más usuales**. Creemos que es necesario saber qué se puede hacer con ellos aunque a veces necesitemos la chuleta con la orden apropiada. Si sólo nos dedicásemos a usar Linux como un entorno de oficina es posible que el número de comandos necesarios sea mínimo, pero si deseamos administrar nuestro sistema Linux no queda más remedio que ampliar el conocimiento sobre ellos.

El tema sobre comandos se ha dividido en dos partes: por un lado tenéis una referencia rápida de qué hace cada uno. Por otro, se han analizado con más detalle aquellos que tienen más utilidad.

Recordar de nuevo la facilidad de uso que representa la autocompletación de comandos. Cuando queramos ejecutar un comando, no tenemos que conocer su nombre exacto ni el del fichero que le pasamos como parámetro para poder trabajar con él. Así, por ejemplo, si deseamos saber qué comandos comienzan por las letras `wh` escribiremos

```
$ wh
```

y tras pulsar la tecla [**Tab**] dos veces, nos aparecerán las concordancias encontradas en nuestro `path`.

```
whatis        which         whiptail         whoami
whereis       while         who               whois
```

Si la concordancia es única, se autocompletará el comando pulsando una sola vez la tecla.

Para "abrir" boca un mini resumen de la equivalencia entre los comandos más usuales del DOS y los de Linux

Cuadro 2.1: Del DOS a Linux

Descripción	**DOS/Windows**	**Linux**
Ayuda	`help`	`man`
Copiar ficheros	`copy`	`cp`
Contenido de un fichero	`type`	`cat`
Renombra un fichero	`ren`	`mv`
Mover ficheros/directorios	`move`	`mv`
Lista archivos	`dir`	`ls`
Borra archivos	`del`	`rm`
Borra la pantalla	`cls`	`clear`
Terminar una sesión	`exit`	`exit`
Crea un directorio	`mkdir`	`mkdir`
Borra un directorio	`rmdir`	`rmdir`
Cambiar de directorio	`cd`	`cd`
Cambiar atributos de ficheros	`attrib`	`chmod`
Cambiar la fecha	`date`	`date`
Compara ficheros	`fc`	`diff`
Memoria libre	`mem`	`free`
Imprimir un fichero	`print`	`lpr`
Editar un fichero	`edit`	`mcedit`
Mandar paquetes	`ping`	`ping`
Configuración interfaz de red	`ipconfig`	`ifconfig`
Configuración interfaz de red	`winipcfg`	`ifconfig`

➥ **Para practicar:**

El paquete `mtools` se instala por defecto, trae los comandos: `mcopy, mdir`, ... similares a los de MS-DOS, la única diferencia es que hemos de anteponer una `m` al comando. Por ejemplo:

```
$ mcopy a:* /home/thales
```

copia el contenido del floppy en el subdirectorio indicado. Es interesante resaltar que **para usarlos no es necesario montar el floppy**.

El fichero de configuración de este paquete es `/etc/mtools.conf`. En general no hay que modificarlo nunca pero si algo no funciona bien puede que tengamos que ajustarlo a nuestro sistema.

1. Comprobar qué comandos componen el paquete usando
   ```
   $ info mtools
   ```
2. Formatear un disquete con la orden
   ```
   $ mformat a:
   ```
3. Listar el contenido del disquete con `mdir`
   ```
   $ mdir a:
   ```
4. Copiar el fichero[1] `~/.bashrc` al disquete usando las `mtools`:
   ```
   $ mcopy ~/.bashrc a:
   ```
5. Montar el disquette creado con la orden `mount` y comprobar que la copia del fichero se ha realizado correctamente.
6. El sistema mantiene una lista de los sistemas de ficheros montados actualmente, en el fichero `/etc/mtab`. Se puede ver el contenido del fichero utilizando el comando `mount` sin argumentos:
   ```
   $ mount
   ```
   ■

[1] El caracter `~` referencia el `home` de ese usuario.

2.1.1. Convenciones en cuanto a la sintaxis

La sintaxis común a todos los comandos es:

```
comando [opciones][parámetro_1] parametro_2 ...
```

donde las opciones y los parámetros son opcionales si van entre corchetes e imprescindibles cuando van solos[2]. Si además algún parámetro va seguido de tres puntos suspensivos es para indicar que pueden incluirse cuantos parámetros de ese tipo se quieran.

Las opciones, en general se le pasan al comando como una serie de valores precedidos por un guión, por ejemplo:

```
$ def -h -l
S.ficheros          Tamaño Usado  Disp Uso% Montado en
/dev/hda2              49G  4,8G   41G  11% /
tmpfs                 252M     0  252M   0% /dev/shm
```

nos informa de la utilización del espacio en disco del sistema de ficheros. Al pasarle como opciones:

-h (`--human-readable`) añade una letra indicativa del tamaño, como M para megabytes, a cada tamaño

-l hace que se limite el listado a los sistemas de ficheros locales, no en máquinas remotas que pudieran estar montados por NFS (*Network File System*)[3], por ejemplo.

En general, esta forma de poner las opciones es equivalente a poner un solo guión y los valores de las opciones a partir del guión como una cadena de caracteres. Así la orden anterior es equivalente a escribir:

```
$ df -hl
```

2.1.2. Comodines

De igual manera que en sistemas DOS[4], en Linux se puede hacer uso de comodines para hacer referencia a nombres de archivos, las posibilidades son:

***** igual que en sistemas DOS, el comodín se sustituye por cualquier cadena de caracteres

? la interrogación también tiene el uso habitual, se sustituye por cualquier carácter, pero sólo uno.

[..] El uso de corchetes permite hacer referencia a un solo carácter, las posibilidades son:

- hacer referencia a un solo carácter pero con la obligatoriedad de ser uno de los valores listados entre corchetes:

  ```
  $ ls ed[89]linux
  ```

  en este caso se mostrarían los ficheros cuyo nombre sea de la forma `ed9linux` o `ed8linux`

- hacer referencia a un rango de valores separados por un guión:

[2]

- Ejecutar `$ man free` para comprobar que todas las opciones y parámetros son opcionales.
- El comando `write`, que sirve para enviar un mensaje a otro usuario conectado al sitema, necesita al menos el argumento `user`.

[3]Sistema de archivos de red, para compartir sistemas de archivos entre equipos que funcionan en red

[4]Más bien DOS lo tomó de Unix. Recordemos que Unix es de finales de los años 60.

```
$ ls ed[7-9]linux
```

en esta caso se mostrarían todos los ficheros cuyo nombre fuese de la forma `ed7linux`, `ed8linux` o `ed9linux`.

- invertir el rango anteponiendo el signo `!`

```
$ ls ed[!1-8]linux
```

en este caso se mostrarán todos los ficheros con tercer carácter arbitrario y distinto de los números 1 al 8 (ambos inclusive)

Se pueden mezclar entre ellos, así:

```
$ ls ed?[7-9]*
```

mostrará todos los ficheros cuyo nombre verifique que:

1. Sus dos primeros caracteres sean "ed"
2. El tercer carácter puede ser cualquiera
3. El cuarto carácter sea un número comprendido entre 7 y 9
4. El resto de caracteres pueden ser cualesquiera

2.2. Resumen de comandos

2.2.1. Ayuda

apropos Busca las páginas de ayuda que contienen la clave que especifiquemos

info Permite el acceso a la ayuda *online* de un comando

man Para visualizar las páginas man

whatis Busca palabras completas en la base de datos `whatis`

2.2.2. "Construir" comandos

alias Se usa para definir abreviaturas para comandos largos. También nos muestra una lista con los alias ya definidos

type Indica cómo interpretaría la shell el comando pasado como argumento

unalias Para eliminar las abreviaturas que previamente hemos definido con `alias`

2.2.3. Gestión de usuarios y grupos

chgrp Cambia el grupo de un archivo

chmod Cambia los permisos de acceso de ficheros

chown Cambia el usuario y grupo propietarios de ficheros

groups Muestra los grupos en los que está un usuario

addgroup Crea un nuevo grupo

delgroup Borra un grupo

newgrp Para hacer que el grupo que especifiquemos sea, desde ese momento, nuestro grupo activo.

passwd Para asignarle la contraseña a un usuario

umask Establece la máscara de creación de ficheros

adduser Para añadir un usuario

userdel Permite eliminar un usuario

2.2.4. Manipulación de archivos y directorios

cd Cambia el directorio de trabajo

cp Copia ficheros y directorios

file Determina el tipo de un fichero

ls Nos muestra el contenido de un directorio (`dir`, `vdir` son versiones de `ls`)

ln Permite crear enlaces entre ficheros

mkdir Crea directorios

mv Mueve (renombra) ficheros

rm Borra ficheros o directorios

rmdir Borra directorios vacíos

pwd Muestra el nombre del directorio de trabajo actual

touch Actualiza la fecha de un archivo a la actual

2.2.5. Localización de archivos

find Busca ficheros en un árbol de directorios

locate Permite localizar archivos basándose en una base de datos que se va actualizando periódicamente

whereis Localiza los ficheros binarios, fuentes y páginas del manual correspondientes a un programa

which Muestra el `path` del archivo de comandos pasado como argumento

2.2.6. Procesamiento de archivos

cat Concatena archivos y también muestra su contenido usando la salida estándar

cmp Compara dos archivos

csplit Divide un archivo en secciones determinadas por líneas de contexto

cut Imprime secciones de líneas de un archivo de entrada

dd Convierte y copia un fichero

diff Busca diferencias entre dos archivos o directorios

expand Convierte las tabulaciones en espacios

fold Permite ajustar las líneas de texto al ancho que especifiquemos

grep, egrep, fgrep Muestran líneas de ficheros que concuerdan con un patrón

head Muestra la parte inicial de un archivo (por defecto 10 primeras líneas)

less Muestra archivos en pantalla de una vez paginando la salida, permite volver atrás

more Filtro que muestra un archivo pantalla a pantalla (es mejor less)

nl Numera las líneas de un archivo que no estén en blanco

paste Combina líneas de ficheros

patch Aplica el comando `diff` actualizando el archivo original. Aplica un "parche"

sed Editor de texto no interactivo

sort Ordena las líneas de archivos de texto

split Divide un archivo en varias partes (por defecto de 1000 líneas en 1000 líneas)

tac Invierte el orden de las líneas de un archivo. Cat al revés.

tail Muestra las últimas líneas (10 por defecto) de un documento

tr Cambia unos caracteres por otros

uniq Borra las líneas duplicadas de un archivo ordenado

wc Muestra el número de bytes, palabras y líneas de un archivo

xargs Construye y ejecuta órdenes desde la entrada estándar

zcat Igual que `cat` pero sobre ficheros comprimidos

zless Actúa como `less` pero sobre archivos comprimidos

zmore Igual que `more` pero sobre ficheros comprimidos

2.2.7. Guardar y comprimir ficheros

compress Comprime (o expande) archivos

gunzip Expande ficheros

gzip Comprime/expande ficheros

tar Para empaquetar y desempaquetar archivos y directorios

uncompress Expande archivos

bzip2 Comprime ficheros con una ratio mejor que los anteriores

bunzip2 Descomprime ficheros comprimidos con bzip2

2.2.8. Procesos de control

at Permite planificar la ejecución de tareas

bg Permite ejecutar un proceso interrumpido que está en segundo plano

cron Para planificar órdenes o procesos de forma periódica en el tiempo

fg Sigue con un proceso interrumpido anteriormente, pero en primer plano

free Muestra la cantidad de memoria libre y usada en el sistema

halt Cierra el sistema

jobs Lista la tabla de trabajos en ejecución

kill Termina un proceso

ldd Nos muestra las librerías compartidas que necesitamos para ejecutar un programa

nice Ejecuta un programa con la prioridad de planificación modificada

ps Informa del estado de los procesos

printenv Imprime parte o todo el entorno

pstree Proporciona un árbol de los procesos en ejecución

reboot Reinicia el sistema

shutdown Cierra el sistema

sync Vuelca a disco los *buffers* del sistema de archivos

uname Imprime información del sistema

2.2.9. Control de usuarios

chfn Cambia los datos de un usuario

chsh Cambia el shell

groups Imprime los grupos en los que está un usuario

id Muestra los identificadores de usuario y de grupo

last Muestra los últimos accesos al sistema

passwd Cambia contraseñas

su Ejecuta una shell con identificadores de grupo y de usuario distintos

whoami Muestra el usuario con el que estamos trabajando

2.2.10. Administrar ficheros

df Informa de la utilización del espacio de disco en sistemas de ficheros

du Lista el espacio ocupado por los archivos o directorios

fdformat Formatea un disquete

fdisk Manipulador de tablas de particiones para Linux

fsck Chequea y repara un sistema de archivos de Linux

mkfs Construye un sistema de ficheros de Linux

mknod Crea ficheros especiales de bloques o caracteres

mkswap Construye un área de intercambio para Linux

mount Monta un sistema de ficheros

swapoff Deshabilita dispositivos o ficheros de intercambio

swapon Habilita dispositivos o ficheros de intercambio

tty Imprime el nombre del fichero del terminal conectado a la entrada estándar

umount Desmonta sistemas de ficheros

2.2.11. Comunicaciones y redes

finger Proporciona información sobre los usuarios conectados al sistema

mail Programa destinado al envío y recepción de correo

mesg Permite permutar la posibilidad de recibir mensajes de otros usuarios

talk Permite establecer una "charla" con otro usuario

wall Manda un mensaje o un archivo a todos los usuarios que admitan mensajes con `write`

w Muestra qué usuarios están conectados y qué están haciendo

who Muestra información de los usuarios conectados al sistema

write Manda un mensaje a la pantalla de un usuario

2.2.12. Comandos de Impresión

lpq Muestra los trabajos en la cola de impresión

lpr Envía un trabajo a la impresora o pone en cola un trabajo de impresión

lprm Elimina un trabajo de la cola

lpstat Permite comprobar el estado de los trabajos de impresión

2.2.13. Módulos del kernel

depmod computa las dependencias entre módulos

lsmod lista los módulos activos

insmod carga un módulo en el kernel

rmmod descarga un módulo cargable

2.2.14. Varios

cal Calendario

clear Borra la pantalla

date Proporciona o ajusta la fecha y hora del sistema

dmesg Permite ver los mensajes de inicio del sistema

echo Muestra el texto/contenido de la variable

env Muestra el entorno actual de trabajo con todas sus variables

exit Cierra el shell actual

nohup Permite que un comando se ejecute aunque se cierre la sesión, y sin salida a un `tty`

time Tiempo que tarda en ejecutarse un comando

2.3. Algunos ejemplos

En esta sección vamos a ver varios ejemplos de cómo se utilizan algunos de los comandos anteriores[5]. Hay grupos de comandos del resumen anterior que ya se han visto, y, por tanto, no se ponen de nuevo aquí; entre ellos estarían los comandos de impresión, los de gestión de usuarios, ayuda, etc.

Hemos seguido el convenio de poner:

comando

```
sintaxis_usual
```

2.3.1. "Construir" comandos

En el capítulo sobre la Shell Bash ya se ha visto y comentado el funcionamiento de estos comandos. Retomemos algunos aspectos más sobre ellos.

alias

```
alias [-p] [nombre[=valor] ...]
```

➥Para practicar

Como ya hemos visto, un alias nos permite invocar a un comando con otro nombre distinto. Uno de los usos más "típicos" del comando `alias` consiste en definir en el fichero `~/.bashrc` la serie de "alias"

```
alias ls='ls --color=auto '
alias cd..='cd ..'
.....
```

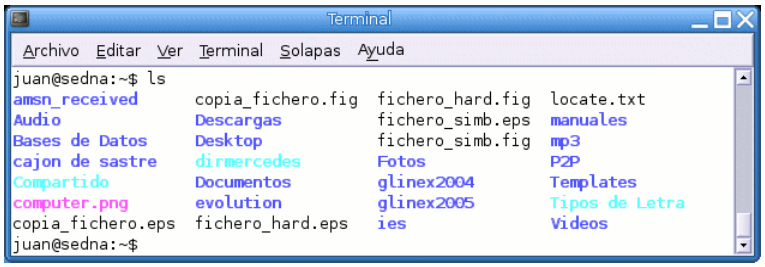
así, por ejemplo, cuando ejecutemos el comando `ls` veremos los ficheros/directorios de distintos colores y podremos usar `cd..` como sinónimo de `cd ..`

Antes de ponerlos en el fichero `~/.bashrc` debemos practicar con ellos desde la línea de comandos. Si hemos realmente definido el alias `ls` anterior y ejecutamos

[5] Las páginas man dan una información exaustiva de los mismos.

```
$ ls
```

comprobaremos que, dependiendo de qué tipo de fichero estemos considerando, se ve de distinto color:



type

```
type comando
```

El comando type indica cómo interpretaría la shell el comando pasado como argumento. Si ejecutamos

```
$ type ls
```

obtendremos:

```
ls is aliased to ‘ls --color=auto’
```

unalias

```
unalias nombre_alias...
```

Con `unalias` podemos quitar los alias definidos, así si ejecutamos

```
$ unalias ls
```

los nombres de ficheros no aparecerán en distinto color y si ahora ejecutamos

```
$ type ls
```

obtendremos:

```
ls is hashed (/bin/ls)
```

es decir, `ls` se quedaría con las opciones que tiene por defecto.

2.3.2. Manipulación de archivos y directorios

La mayoría de los comandos que aparecen en este grupo son ya conocidos por los que venimos del MSDOS, lo que ocurre es que puede que se nos haya olvidado su nombre completo[6].

cd

```
cd [directorio]
```

[6] ¿quién se acuerda ya del `edlin`?

Retomemos a nuestro linuxero THALES, que se encuentra trabajando en su directorio de usuario `/home/Thales`. Tiene que moverse por el árbol de directorios y desplazarse al directorio raíz, para ello ejecuta:

```
$ cd /
```

Después se mueve a

```
$ cd /etc/X11
```

para ver el contenido de un fichero. Una vez terminada la labor, vuelta a casa

```
$ cd
```

y listo, el sistema lo lleva a `/home/Thales`.

Pero siempre se olvida algo, necesita volver al directorio en el que se encontraba anteriormente (`/etc/`X11) y ejecuta:

```
$ cd -
```

cp

```
cp [opciones] fuente destino
```

Es el comando para copiar ficheros. Una vez en su directorio home, recuerda que tiene que copiar el fichero `/home/Thales/curso/entrega_3.gz` al subdirectorio `/ed04linux`, para hacer esto escribe:

```
$ cp /home/Thales/curso/entrega_3.gz /ed04linux
```

file

```
file archivo...
```

Este comando muestra el tipo del archivo que le pasemos como argumento. THALES no recuerda con qué aplicación lo hizo y para ello ejecuta:

```
$ file /ed04linux/entrega_3.gz
```

y ve en el terminal :

```
entrega_3.gz: gzip compressed data, was "entre-
ga_3", from Unix, max compression
```

con lo que recuerda que ese fichero no es otro que `entrega_3` comprimido con gzip. Tras descomprimirlo (véase 2.3.5) , ejecuta de nuevo:

```
$ file /ed04linux/entrega_3
```

y el resultado ahora es:

```
entrega_3: ISO-8859 text
```

es decir, es un documento de texto.

ls

```
ls [opciones] [archivo, directorio]
```

Quizás, junto con `cd`, el comando más usado en Linux sea `ls` (o alguna de sus variantes); `ls` muestra el contenido de un directorio en un listado que por defecto está ordenado alfabéticamente. La sintaxis básica es:

```
$ ls [opciones] [archivo, directorio]
```

donde las **opciones** más importantes son:

a Muestra todos los archivos (hasta los "ocultos", los que empiezan por ".")

f Muestra el contenido de los directorios en el orden en el que están almacenados en el disco.

i Muestra el inodo de los archivos listados.

m Lista los directorios separando los nombres por comas.

r Invierte el orden usual de mostrar el directorio

s Muestra el tamaño de los archivos.

t Ordena los archivos por fecha de creación, primero los más recientes.

R Muestra recursivamente el directorio y sus subdirectorios.

mkdir

```
mkdir [-p[7]] directorio...
```

Continuemos con THALES. Ahora tiene que crear un nuevo subdirectorio en `/ed04linux` donde guardar los gráficos de la entrega cuatro; tras situarse en `/ed04linux` escribe:

```
$ mkdir -p graficos/entrega_4
```

listo, ya tiene su flamante directorio `entrega_4` (con la opción `-p` se ha creado, si no existía, el subdirectorio `graficos`). Después ejecuta `cd` para situarse de nuevo en su home de usuario.

mv

```
mv [-i[8]] origen destino
```

pero necesita mover el fichero `peguin.png` que se encuentra en `~/curso` al nuevo directorio creado y entonces ejecuta:

```
$ mv curso/penguin.png /ed04linux/graficos/entrega_4/
```

rm

```
rm [opciones] archivo
```

Ahora recuerda que ya no necesita el fichero original `entrega_3.gz` (estaba en `/home/Thales/-curso`) y decide borrarlo:

```
$ rm curso/entrega_3.sgml
```

rmdir

```
rmdir directorio...
```

[7] Con esta opción crea los directorios precedentes en caso de que no existan
[8] Pregunta antes de sobreescribir un archivo de destino que ya exista.

THALES se da cuenta de que ni ese directorio `(/home/Thales/curso/)` ni su contenido los necesita y decide borrarlos, para ello ejecuta:

```
$ rmdir /home/Thales/curso/
```

y recibe un error del sistema, ¡`rmdir` sólo borra directorios vacíos! (¿qué se la va a hacer? a grandes males...), así que escribe:

```
$ rm -r /home/Thales/curso/
```

y listo, ha borrado el directorio `curso` y todos los archivos, directorios y subdirectorios contenidos en él.[9]

➥Para practicar:

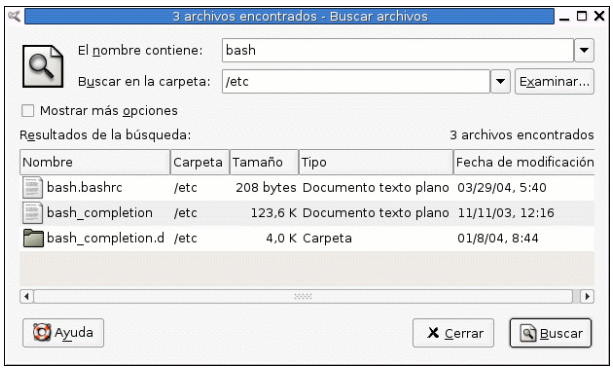
Mirar las opciones del comando `rm` y definir el alias:

```
$ alias rm="rm -i"
```

para que este comando pida confirmación antes de borrar un fichero. Si te parece buena idea, define este alias en el fichero .bashrc ■

2.3.3. Localización de archivos

Con `mc` o con `gnome-search-tool` (en modo gráfico: **Acciones→Buscar archivos**...) la búsqueda de ficheros está "tirada".



locate

```
locate patrón
```

Otra forma de buscar ficheros es usando el comando `locate`.

Supongamos que queremos modificar el fichero `sources.list`, pero no recordamos su ubicación, así que ejecutamos:

[9] `rm` no tiene vuelta atrás, los ficheros borrados no van a la papelera, así que cuidado especialmente con los comodines y con la opción -r.

La "r" viene de Recursivo ¿os imagináis qué pasaría si como root escribís?:

```
# rm -r /
```

O si en un directorio cualquiera, como root, ejecutamos esta otra, pensando en eliminar los ficheros ocultos (cuyo nombre empiezan por punto):

```
# rm -r .*
```

Recordaremos la película del aprendiz de brujo en la que las escobas no paraban de traer cubos de agua y desearemos pararlo como sea, aunque casi siempre demasiado tarde. Lo que ha ocurrido es que una de las expansiones de `.*` será el fichero `..` que es precisamente el directorio superior. ¡¡¡Socorro, ayuda!!!

```
$ locate sources.list
```

y el sistema nos devuelve[10] el mensaje:

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

La razón del mensaje es clara; el comando, en la búsqueda, utiliza el fichero `locatedb` que contiene la base de datos de nombres de ficheros almacenados en el sistema, así que previamente debemos actualizarla, para ello, como root, escribiremos:

```
# updatedb &
```

ejecutamos el comando en segundo plano ya que tiene que localizar y almacenar los nombres de todos los ficheros y esto puede llevar algún tiempo. De esta forma, podremos seguir trabajando en nuestro terminal mientras el sistema lleva a cabo esa tarea. Tan sólo tendremos que actualizar la base de datos de ficheros cada vez que se haya producido un cambio sustancial en el número de ficheros.

Tras finalizar, si ejecutamos de nuevo

```
$ locate sources.list
```

el sitema nos devolverá una salida parecida a ésta:

```
/etc/apt/sources.list
/etc/apt/sources.list-guadalinex
/etc/debtags/sources.list
/usr/share/doc/apt/examples/sources.list
/usr/share/man/es/man5/sources.list.5.gz
/usr/share/man/fr/man5/sources.list.5.gz
/usr/share/man/man5/sources.list.5.gz
```

El comando

```
$ locate patron
```

muestra todas las concordancias en la base de datos de nombres de ficheros con ese `patron`. Por eso el listado anterior incluye otros ficheros.

find

```
find [camino...] [expresión]
```

Con `find` podemos encontar archivos basando su búsqueda en distintas características de los mismos.

El número de opciones de `find` es muy elevado (`$ man find`)

which

```
which comando
```

Si queremos conocer el path completo de un determinado comando o ejecutable, como por ejemplo de `lyx`, usaremos:

```
$ which lyx
```

la respuesta sería:

[10] si no se ha actualizado recientemente la base de datos de nombres de ficheros

```
/usr/bin/lyx
```

whereis

```
whereis comando
```

si no nos basta con esta información y, además, queremos saber qué páginas del manual acompañan al programa escribiremos:

```
$ whereis lyx
```

el resultado es:

```
lyx: /usr/bin/lyx /usr/share/lyx /usr/share/man/man1/lyx.1.gz
```

➥**Para practicar**

- Uso de `find`
  - Encuentra todos los archivos que hay en el directorio actual y en sus subdirectorios con extensión `.txt`
    `$ find . -name "*.txt"`[11]
  - Encuentra los ficheros con permisos 777
    `$ find * -perm 777`[12]
- Localizar con `locate` los archivos de nombre internet/Internet, para eso hemos de añadir el párámetro -i
  `$ locate -i internet`
- Buscar el path del comando `ls` y las *páginas de manual* de este comando.
- Como un usuario normal hacer lo mismo con el comando `fdisk` ¿por qué which no lo encuentra?
- Haz un `su` a root y repite de nuevo el apartado anterior.
- Para entender la razón de lo que está pasando ejecuta, primero como usuario y después como root:

  `echo $PATH` ■

2.3.4. Procesamiento de archivos

Con respecto a las órdenes para procesar archivos lo idóneo es saber que están ahí y las posibilidades que tenemos con ellas. Algunas son bastante especializadas y su uso suele ser ocasional. En caso de necesitarlas, siempre tendremos a nuestra disposición las páginas man.

Hemos utilizado en repetidas ocasiones a lo largo del curso `cat` y `less` en su forma más estándar.

Otra orden que puede sacarnos de apuro en un determinado momento es `split`. Aunque los medios de almacenamiento extraibles cada vez tienen mayor capacidad, puede ocurrir, por ejemplo, que una copia de seguridad no quepa en un CD o sencillamente que queramos guardar la "tercera entrega" y sólo dispongamos de disquetes, `split` permite "romper" un fichero en fragmentos de un tamaño prefijado que mas tarde podemos volver a fusionar en un solo fichero.

split

```
split opciones fichero [prefijo]
```

El fichero `entrega3.pdf.gz` ocupa 2.2 Mb y queremos fragmentarlo en trozos de 1Mb, para lo que ejecutamos:

[11] Las dobles comillas son necesarias para que no interprete el asterisco. Prueba sin comillas.
[12] En estos dos casos, el asterisco significa el camino y sí queremos que se expanda, por eso no va entre comillas.

```
$ split -b 1m entrega3.pdf.gz trozosentrega3-
```

Podemos comprobar el resultado haciendo un

```
$ ls -l
-rw------- 1 juan users 2316913 2005-04-23 20:18 entrega3.pdf.gz
-rw-r--r-- 1 juan users 1048576 2005-04-23 20:21 trozosentrega3-aa
-rw-r--r-- 1 juan users 1048576 2005-04-23 20:21 trozosentrega3-ab
-rw-r--r-- 1 juan users 219761 2005-04-23 20:21 trozosentrega3-ac
```

Ahora podemos copiar los "trozos" en disquetes, por ejemplo, con la orden

```
$ mcopy trozosentrega3-aa a:
```

y lo mismo con los otros dos. Finalmente, una vez los ficheros en la máquina destino con

```
$ cat trozosentrega3-* >entrega3.pdf.gz
```

obtenemos el fichero original. Para ampliar, `man split`.

2.3.5. Empaquetar y comprimir ficheros.

Linux dispone de múltiples utilidades y programas para comprimir y descomprimir ficheros. Recordemos que en modo gráfico disponemos de `File Roller`, **Aplicaciones→Accesorios→File Roller**, que ya lo vimos en la segunda entrega, así que ahora nos detendremos en los más utilizados en modo comando, `tar` para empaquetar y `gzip` para comprimir.

tar

```
tar opciones archivo.tar fichero1 [fichero2 ...]
```

Este comando permite empaquetar o desempaquetar ficheros. El concepto de empaquetar aquí es el de meter varios ficheros y/o directorios en un solo fichero. Posteriormente podremos recuperar esa estructura de ficheros y directorios en el lugar donde queramos.

Las **opciones** más usuales son:

c para crear archivos empaquetados

x para expandir archivos empaquetados

t para mostrar el contenido de un fichero tar empaquetado

v almacenamos/visualizamos la información en forma detallada

f para indicar que `archivo.tar` es un fichero.

z filtrar el archivo a través de `gzip` (tanto para comprimir como descomprimir).

M para crear/desempaquetar usando varios discos.

archivo.tar es el nombre del archivo tar y

ficheroi nombre[13] del directorio/fichero o directorios/ficheros a empaquetar separados por espacios

Supongamos que deseamos empaquetar dos ficheros llamados `ed03linux1.txt` y `ed03linux2.txt`, en un fichero tar de nombre `ed03linux.tar`, escribiremos:

[13] Podemos usar comodines

```
$ tar -cvf ed03linux.tar ed03linux1.txt ed03linux2.txt
```

también podíamos haber escrito:

```
$ tar -cvf ed03linux.tar ed03linux?.txt
```

Si lo que queremos es empaquetar el directorio `entrega_3`, en el fichero `entrega_3.tar`, la sintaxis sería:

```
$ tar -cvf entrega_3.tar entrega_3
```

Si lo que deseamos es desempaquetar un fichero tar, en vez de escribir la opción `-c` escribiremos `-x`, así para desempaquetar el contenido de la entrega anterior escribiremos:

```
$ tar -xvf entrega_3.tar
```

Si solamente queremos ver el contenido del fichero empaquetado (tar), ejecutaremos.

```
$ tar -tvf entrega_3.tar
```

gzip

```
gzip [opciones] archivo
```

gzip permite comprimir ficheros.
Las opciones básicas son:

d para descomprimir archivos

t para comprobar que la compresión se ha realizado con éxito

1-9 es el nivel de compresión, el 1 indica menor ratio y mayor rapidez, el 9 daría como resultado un archivo más pequeño pero un mayor tiempo de compresión. El nivel por defecto es 6

archivo es el nombre del archivo a comprimir

Ya tenemos empaquetado nuestro archivo `entrega_3` y deseamos comprimirlo al máximo y verificar que todo está bien, usaremos:

```
$ gzip -9 entrega_3.tar
```

el resultado es el fichero `entrega_3.tar.gz`, si queremos comprobar la "integridad" del fichero escribiremos:

```
$ gzip -tv entrega_3.tar.gz[14]
entrega_3.tar.gz:      OK
```

Si ahora queremos descomprimir este fichero tenemos dos opciones:

```
$ gzip -d entrega_3.tar.gz
```

o bien escribimos directamente

```
$ gunzip entrega_3.tar.gz
```

[14] La `v` de "verbose", es decir, para que me muestre más información en el terminal de cómo ha ido el proceso de testeo.

Los dos a la vez: tar y gzip Cabe la posibilidad de empaquetar y comprimir directamente sin tener que usar un comando tras otro, una posibilidad consiste[15] en escribir:

```
$ tar -czvf nombre_fichero.tar.gz origen
```

o bien

```
$ tar -czvf nombre_fichero.tgz origen
```

en ambos casos, la opción `-z` es la que señala que vamos a comprimir. La extensión es `tgz` es equivalentes a `tar.gz`.

También podemos descomprimir y desempaquetar un fichero `tar.gz` o un `tgz` usando sólo la orden `tar`, la sintaxis sería:

```
$ tar -xzvf fichero.tar.gz
```

o bien

```
$ tar -xzvf fichero.tgz
```

➥Para practicar: "targz"

El objetivo de esta práctica reside en aprender a empaquetar/comprimir directorios. Vamos a trabajar con el subdirectorio `/usr/share/doc/mozilla-browser`.

Notar que estamos trabajando como un usuario normal y no como root, pues un error como tal, en la sintaxis del último comando puede ser desastrosa para el sistema.

1. Empaquetar y comprimir
   ```
   $ tar -cvf ~/mozilla.tar /usr/share/doc/mozilla-browser
   $ ls -l mozilla.tar          (es grande)
   $ gzip mozilla.tar
   $ ls -l mozi*                (ya es de menor tamaño)
   ```
2. Ahora de un tirón
   ```
   $ tar -czvf ~/mozilla.tgz /usr/share/doc/mozilla-browser
   $ ls -l mozi*
   ```
   ¿Salen del mismo tamaño?
3. Descomprimirlo de una pasada:
   ```
   $ tar -xzvf mozilla.tgz
   ```
   Comprobar que todo ha salido bien; tendremos, colgando del sitio donde hemos descomprimido, el árbol `/usr/share/doc/mozilla-browser`.
4. Ahora con formato zip:
   ```
   $ zip -r ~/mozilla.zip /usr/share/doc/mozilla-browser
   ```
   ¿Cuál comprime mejor?
5. Por último, borremos el directorio creado al descomprimir mozilla.tgz:
   ```
   $ rm -r ~/usr
   ```

■

bzip2, bunzip2

Para comprimir y descomprimir ficheros existen más herramientas que las ya comentadas pero la única que merece mención especial es `bzip`. La extensión de este tipo de ficheros es `.bz2` [16], para comprimir un fichero escribiremos[17]

[15] Sin usar tuberías

[16] Es un formato con mejores ratios de compresión que los que ofrece `gzip`; `bzip2` comprime ficheros utilizando el algoritmo de compresión de texto por ordenación de bloques de Burrows-Wheeler.

[17] Para ampliar es mejor consultar la manpage que acompaña al programa

```
$ bzip2 fichero
```

y para descomprimir un fichero:

```
$ bunzip2 fichero.bz2
```

➥ **Para practicar**

Empaquetemos y comprimamos otra vez el subdirectorio `/usr/share/doc/mozilla-browser`, pero ahora con:

```
$ tar -cjvf ~/mozilla.tar.bz2 /usr/share/doc/mozilla-browser
```[18]

¿Cuál es el más comprimido?

Para descomprimirlo desempaquetarlo:

```
$ bunzip2 mozilla.tar.bz2
$ tar -xf mozilla.tar
```

De una sola vez con:

```
$ tar -xjvf mozilla.tar.bz2
```

Borremos ahora toda la "basura" genereda:

```
$ rm mozi*
$ rm -r ~/usr
```

■

2.3.6. Control de tareas

Se entiende como *proceso* a cualquier programa en ejecución. El comando `ps` lista los procesos en ejecución en ese momento.

ps

```
ps [opciones]
```

donde las **opciones** más usuales son:

l Formato grande

u Da el nombre de usuario, la hora de comienzo y el uso de los procesos de este usuario de la máquina.

a Muestra también procesos de otros usuarios.

x Muestra los procesos sin terminal de control.

r Muestra sólo procesos que estén activos.

txx Muestra los procesos controlados por el terminal txx

Por ejemplo:

```
$ ps
```

muestra los procesos en ejecución del usuario actual; una posible salida es:

```
PID  TTY      TIME        CMD
3846 pts/1 00:00:00 bash
3899 pts/1 00:00:00 ps
```

el PID es el número de indentificador del proceso para la shell y CMD el nombre del proceso.

Existen otros procesos en ejecución que no han sido listado por el comando anterior, si queremos verlos todos:

```
$ ps -aux | less
```

[18] es equivalente a usar:
`$tar -cvf ~/mozilla.tar /usr/share/doc/mozilla-browser | bzip2`

Primer y segundo plano

Casi todas las shell ofrecen la posibilidad de controlar la ejecución de los procesos y desde esta perpectiva, a los procesos se les conoce también con el nombre de *tarea*s.

Generalmente cuando lanzamos un procesos lo hacemos en **primer plano**, introducimos el comando pulsamos enter y cuando el proceso ha terminado deja libre la shell para introducir nuevos comandos. A veces algunos procesos necesitan algún tiempo para terminar y no hacen nada interesante mientras tanto; en este caso lo mejor es lanzarlo en **segundo plano**. Para ello ejecutamos[19]

```
$ comando &
```

Conviene clarificar también la diferencia entre **interrumpir** y **suspender** un programa. Cuando interrumpimos un proceso (generalmente con **[Control]+[c]**) este muere, deja de estar en memoria, mientras que si lo suspendemos (generalmente con **[Control]+[z]**), el proceso se para temporalmente y podremos decir al sistema que continúe con la tarea más tarde.

fg, bg, jobs, kill

Supongamos que queremos actualizar nuestra base de datos de nombres de ficheros para el comando `locate`, para ello ejecutamos:

```
# updatedb
```

Observamos que tarda en terminar y suspendemos su ejecución con **[Control]+[z]** . El sistema nos devuelve el mensaje:

```
[1]+        Stopped        updatedb
```

que es autoexplicativo, el [1] es el número de tarea, el signo + señala la última suspendida.

Si ahora queremos que continúe pero en segundo plano, basta con ejecutar[20]

```
# bg
```

para hacerlo continuar en primer plano,

```
# fg
```

Si el proceso fué lanzado en segundo plano con,

```
# updatedb &
```

el sitema devuelve un mensaje como este[21]

```
[1]     1095
```

la tarea 1 con número de proceso (PID) 1095.

Si intentamos detener el proceso con `[Control]+[z]`, el sistema ni se entera (el proceso está corriendo en segundo plano), así que previamente debemos traerlo a primer plano con `fg` y después ya podemos suspenderlo, relanzarlo o matarlo.

El comando `jobs` (interno de la shell) informa sobre el estado de los procesos (`ps` también).

Con `kill` podemos matar un proceso, la sintaxis más usual es:

[19] ya lo hemos hecho en reiteradas ocasiones a lo largo del curso

[20] Si hubieran mas tareas suspendidas,
`# bg%numero_tarea.`
fg del inglés *foreground* y **bg** de *background*.

[21] el número no tiene por qué coincidir

```
kill [señal[22]] PID...[23]
```

donde `señal` es opcional y en general toma dos valores

-15 (`SIGTERM`) es la señal por defecto y no siempre es capaz de "matar" todos los procesos

-9 (`SIGKILL`) es el "Rambo" de las señales, acaba con cualquier proceso.

Si no especificamos ninguna señal, estamos mandando la 15 y de alguna manera le estamos diciendo al proceso que muera por las buenas; es deseable que sea así, pues de esta forma el proceso puede cerrar los ficheros, descargar los datos de memoria a disco y decirle a sus hijos (en caso de que los tuviera) que también se mueran por las buenas.

Si nos vemos obligados a utilizar la señal 9, lo matamos bien muerto, sin tiempo a que cierre ficheros ni descargue datos de memoria a disco. Moraleja, intentaremos mandarle primero un `kill` normal y si no hay manera pasaremos a la artillería pesada.

Supongamos que hemos cerrado `mozilla` y que notamos que el sistema "está lento", escribimos:

```
$ ps -ax
```

y ¡date!,vemos que `mozilla` sigue en ejecución con el PID

```
...
3940 tty1 S 0:01 /usr/lib/mozilla-1.0.1/mozilla-bin
...
```

ejecutamos entonces:

```
$ kill -9 3940
```

y listo, se acabó `mozilla` (en sentido figurado, claro está). Si ahora ejecutamos de nuevo:

```
$ ps -ax
```

no debería aparecer `mozilla` por ningún lado.

Si la lista de procesos es muy larga también podemos filtrar la salida con `grep`:

```
$ ps -ax | grep mozilla
```

➥ Para practicar

1. Ejecutar la secuencia de comandos
   ```
   # updatedb &
   # tar -czf home.tgz /home
   [Control]+[z]
   # jobs
   # ps
   ```
2. Uso conjunto de `ps` y `kill`
   En modo gráfico abrir una xterm y ejecutar en segundo plano el programa `gedit`
   ```
   $ gedit &
   ```
   Comprobar la ID del proceso anterior con
   ```
   $ ps -a
   ```
   y la diferencia de usar:
   ```
   $ ps -ax
   $ ps -aux
   ```
   Matar el programa con:
   ```
   $ kill -15 PID_gedit
   ```

■

[22] Con
```
$ kill -l
```
podemos visualizar todas las señales posibles (es una ele minúscula).

[23] En este caso no hemos puesto ni `$` ni `#` ya que el root podrá "matar" procesos de todos los usuarios, pero un usuario tan sólo podrá "matar" los suyos.

at

En Guadalinex no se instala este comando por defecto, así que antes de nada debemos instalarlo:

```
#apt-get update
#apt-get install at
```

Además, para poder comprobar lo que se expone sobre el comando `at` respecto al envío del correo hemos de instalar el programa `mailx` y configurar de forma adecuada `exim`[24].

```
#apt-get install mailx
```

El comando `at` posibilita planificar la ejecución de tareas; permite que le especifiquemos tanto la fecha como la hora para activarse. Una vez activo, `at` se encargará de hacer ejecutar las órdenes programadas (órdenes no interactivas). Su sintaxis es:

`at hora [fecha] lista_comandos`

Por ejemplo supongamos que son las 3 h pm y hemos quedado a las 4 h pm, somos tan despistados que cuando nos ponemos con el ordenador se nos olvida todo, en ese caso podemos decirle a `at` que nos avise dentro de una hora escribiendo:

`$ at now +60 minutes` [25]

tras pulsar intro podremos escribir aquello que consideremos oportuno, por ejemplo:

`at>echo "No te despistes, tienes una cita"`

cuando terminemos de introducir los comandos deseados pulsaremos `[Ctrl]+[d]`.

A las cuatro `at` nos enviará un correo con el texto anterior que podremos visualizar con la orden `mail`.

`at` permite distintas formas para especificar la fecha y hora en que debe activarse. Así, el tiempo se puede especificar en la forma HHMM o HH:MM para llevar a cabo una tarea en el mismo día. Por ejemplo la orden anterior es equivalente a:

`$ at 16:00`

Con `at` es posible usar `midnight` (medianoche), `noon` (mediodia), `teatime` (4 de la tarde) o `tomorrow` (mañana). También podemos anteponer a la hora `am` o `pm`.

Si queremos que `at` se ejecute en un día distinto al que estamos, pondremos la fecha en la forma 'mes día' por ejemplo, May 12.

Asociado al comando `at` tenemos los comandos:

atq muestra un listado de los trabajos en espera de ejecución.

atrm para eliminar trabajos en espera.

➥ **Para practicar:** `at`

Usando el comando `at` programar un trabajo para dentro de 2 minutos (o un tiempo razonable) que:

1. Te mande un correo con el texto que te parezca, una idea: "Curso Linux te saluda"
2. Comprobar/leer el correo con (este comando se ve después , para realizar este apartado debéis mirar antes en la página 37):

   `$mail`

   `&1`

   Para salir

   `&q` ■

[24] Se trata del agente de transporte de correo (MTA) de Guadalinex: "el cartero". Para configurar `exim` se puede ejecutar el comando `eximconfig`. Os remitimos a la ayuda instalada del programa para conocer más sobre él.

[25] También podemos dar el tiempo de espera como un incremento de un número de `weeks` (semanas), `days` (días) u `hours` (horas).

cron (anacron)

Se trata de "la herramienta" que usa Linux para planificar tareas. Para conocerla mejor:

```
$ man cron
```

Veamos el contenido de un fichero de configuración de `cron` básico:

```
$ cat /etc/crontab
```

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file.
# This file also has a username field, that none of the other
    crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || run-parts --
    report /etc/cron.daily
47 6    * * 7   root    test -x /usr/sbin/anacron || run-parts --
    report /etc/cron.weekly
52 6    1 * *   root    test -x /usr/sbin/anacron || run-parts --
    report /etc/cron.monthly
#
```

Listado 2.1: /etc/crontab

Expliquemos un poco este fichero, las líneas tienen la forma:

```
fecha nombre_usuario comando
```

- Como siempre, las líneas que comienzan por `#` son comentarios.
- Con las 2 primeras líneas asignamos las variables de entorno con que se va a ejecutar `cron`.
- ¿Qué hacen las 3 últimas?
  - Con ellas se controlan las acciones a realizar cada día (`/etc/cron.daily`), semana (`/etc/cron-weekly`) y mes (`/etc/cron.monthly`)[26]. Así, por ejemplo:
    `/etc/cron.daily/find` ejecuta cada día `updatedb` con ciertos parámetros
  - Para facilitar la coordinación con `anacron` la llamada de `run-parts` para los directorios `/etc/cron.daily`, `/etc/cron.weekly` y `/etccron.monthly` es

    ```
    test -x /usr/sbin/anacron || run-parts --report /etc/cron.daily
    ```

    De esta forma, si `anacron`[27] se instala será el responsable de la ejecución de esos scripts.
  - Con `run-parts` indicamos que se ejecuten los scripts pasados como parámetro en las fechas seleccionadas al inicio de la línea. El formato en que se codifican las fechas es:

CAMPOS	minuto	hora	día	mes	día de la semana
rango	0-59	0-23	1-31	1-12	0-7

[26] Para controlar las tareas cada hora `/etc/cron.hourly`

[27] `anacron` es un programador de tareas similar a `cron`, pero a diferencia de este último no requiere que el sistema esté en ejecución permanentemente.

- El 0 y el 7 del día de la semana corresponde al domingo (se puede usar *sun*, *sat*, ...).
- 3-6 equivale a la lista de números 3, 4, 5, 6
- El asterisco significa cualquier valor del rango permitido
- Podemos incrementar un valor con el formato `/numero`. Por ejemplo, podemos conseguir que un comando se ejecute cada 8 horas escribiendo `*/8` en el campo hora.

shutdowm

```
shutdowm [opciones] tiempo [mensaje]
```

Es un comando para cerrar el sistema. A continuación exponemos su uso más corriente.

```
# shutdowm -h now
```

o equivalentemente

```
# halt
```

para reiniciar el sistema la orden es:

```
# shutdowm -r now
```

o bien

```
# reboot
```

Los parámetros están claros, `h` para "halt" y `r` para "reboot", y la opción "now" por ahora mismo. Podemos pasarle como argumento el tiempo antes de cerrar/reiniciar el sistema, así en vez de *now* podemos escribir:

```
# shutdowm -h +5
```

con lo que el sistema se cerrará dentro de cinco minutos. Avisa con un mensaje.

uname

```
uname [opciones]
```

Por último, con

```
$ uname -a
```

conseguimos toda la información del sistema, una posible salida es:

```
(Sistema Hosts   Versión del núcleo y fecha de la compilación)
Linux sedna 2.6.5 #1 Wed Sep 29 16:49:48 CEST 2004 i686 GNU/Linux
```

Si no ponemos ninguna opción sólo se nos muestra el nombre del sistema operativo.

2.3.7. Administrar ficheros

Algunos de los comandos que disponemos para administrar sitemas de ficheros como `mkfs, fsck, mount` o `umount`, ya se vieron en la tercera entrega, así que ahora sólo mencionaremos algunos más:

df

```
df [opciones] [sistema_archivos]
$ df
```

cuya salida podría ser (no tiene por qué coincidir):

```
S.ficheros         Bloques de 1K   Usado    Dispon Uso% Montado en
/dev/hda2             50403028   4995476  42847196  11% /
tmpfs                   257332         0    257332   0% /dev/shm
```

es autoexplicativa. Si deseamos tener una salida más "comprensible" podemos escribir:

```
$ df -h
```

en cuyo caso la información se nos mostrará como sigue:

```
S.ficheros          Tamaño  Usado  Disp Uso% Montado en
/dev/hda2              49G  4,8G   41G  11% /
tmpfs                 252M     0  252M   0% /dev/shm
```

du

```
du [opciones] [nombre_archivo...]
```

Lista el espacio ocupado por los archivos o directorios que cuelgan desde donde se invoca. Si se ejecuta sin argumentos es poco práctico. Una de las formas más corriente de uso es:

```
$ du -sh directorio
```

que da el total del espacio ocupado por ese directorio en formato *humano*, es decir, añade una letra indicativa del tamaño, como M para megabytes. Por ejemplo, en mi equipo la salida de:

```
$ du -sh manuales/
```

es

```
210M manuales/
```

fdformat

```
fdformat device
```

Cuando queramos formatear un disco flexible escribiremos (sin tener el disco montado):

```
$ fdformat /dev/fd0
```
[28]

Recordar que exite una utilidad gráfica para formatear disquetes a la que se accede con  **Aplicaciones→Accesorios→Formateador de disquetes,** o desde un terminal gráfico ejecutando el comando

[28] En el supuesto de que nuestro disco flexible sea `/dev/fd0`. Para ver los parámetros de fdformat, ejecutar: `man fdformat`

```
$ gfloppy
```

fdisk

```
fdisk device
```

Desde Linux podemos ejecutar el `fdisk` de Linux para visualizar o modificar las particiones del disco duro. Hay que ejecutarlo como root, en modo texto, es un poco más árido que el `fdisk` del DOS, menos intuitivo que `cfdisk` y por supuesto que `QtParted` que ya utilizamos en la instalación de Guadalinex.

➥ **Para practicar:**

Comprobar el espacio ocupado/disponible:

```
$df -h
```

Espacio que ocupa nuestro directorio de trabajo:

```
$du -sh ~/
```

Particiones del disco:

```
#fdisk -l /dev/hda
```

■

2.3.8. Comunicaciones y redes.

Veremos aquí algunos comandos que pueden sernos de utilidad cuando trabajemos con redes (se estudian en los dos últimos capítulos de esta entrega)

finger

```
finger [-lmsp] [usuario...] [usuario@host...]
```

Proporciona información sobre los usuarios conectados al sistema. Por ejemplo:

```
$ finger
```

muestra entre otros datos: el directorio de conexión, el nombre completo, la fecha de conexión, etc. Si queremos que la información sea más detallada escribiremos:

```
$ finger -l
```

Si sólo queremos información del usuario THALES escribiremos:

```
$ finger Thales
```

who

```
who [opciones]
```

Similar a la anterior.

w

```
w [usuario]
```

w nos da información sobre los usuarios que están conectados en ese momento y sobre sus procesos.

```
$ w
 23:22:42 up 56 min,  3 users,  load average: 0,03, 0,06, 0,05
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
juan     :0       -                22:27   ?xdm?  46.83s  0.29s x-session-manager
juan     pts/0    :0.0             23:05    0.00s  0.09s  0.00s w
mercedes :20      -                23:11   ?xdm?  46.83s  0.22s x-session-manager
```

write

```
write usuario [terminal]
```

El comando `write` nos permite mandar un mensaje a otro usuario conectado a la misma máquina. Previamente ese usuario debe tener `mesg`[29] en "y", en el caso de que ese usuario tenga `mesg` en "n" el sistema nos avisará con el mensaje de error:

```
write: "usuario" has messages disabled
```

Supongamos que deseamos enviar un mensaje al usuario THALES, y que tiene activa la opción de que le envíen mensajes, en ese caso escribiríamos

```
$ write Thales
```

el sistema espera que tecleemos el mensaje, y una vez escrito el texto, por ejemplo

```
Hola Thales, que no se te olvide la cita.
```

pulsaremos `[Ctrl]+[d]` y el mensaje será enviado.

wall

```
wall [archivo]
```

Si queremos enviar un mensaje no a un solo usuario, sino a todos los usuarios conectados, usaremos `wall`. Si lo que queremos es mandar un fichero escribiremos:

```
$ wall < fichero.txt
```

y el contenido de este fichero será enviado al terminal de todos los usuarios conectados al sistema.

mail

Recordemos de nuevo que para el correcto funcionamiento de este comando necesitamos el programa `mailx`, que Guadalinex no instala por defecto, así que:

```
#apt-get update
#apt-get install mailx
```

Una vez instalado la sintaxis a utilizar es:

```
mail [usuario]
```

En el caso de que nuestro usuario de destino **no** esté conectado el mejor comando para comunicarnos con él es `mail`. Si queremos enviar un mensaje a `THALES` escribiremos[30]:

```
$ mail Thales[31]
Subject: Cita
Te recuerdo que tenemos una cita
```

[29] el comando **mesg** permite activar o desactivar la posibilidad de recibir mensajes de otros usuarios, funciona como un "interruptor" con sólo dos estados y o n. Para activarlo:
`$ mesg y`

[30] Para que funcione tal cual se expone en los apuntes véase el **Stop** de la página 32

[31] Esta es la sintaxis en cualquier distribución. En Guadalinex debemos escribir `mail usuario_destino@localhost` o bien `usuario_destino@nombre_maquina`, por ejemplo `mail Thales@guadalinex` (si así se llama nuestra máquina).
Debido a la configuración de mail en Guadalinex, la dirección del remitente aparecerá como `usuario_remitente@andaluciajunta.es`

tras escribir el texto del mensaje pulsamos `[Ctrl]+[d]`(o insertamos un "punto" al inicio de una nueva linea) y se nos mostrará la opción

```
Cc:
```

por si queremos mandar una copia del mensaje a otro usuario.

Cuando THALES se conecte al sistema , éste le avisará de que tiene un correo (si se conecta en modo texto):

```
You have new mail.
```

Tanto en modo texto como gráfico, ejecutando `mail` podrá visualizarlo, borrarlo, etc:

```
$ mail
Mail version 8.1.2 01/15/2001. Type ? for help.
"/var/mail/Thales": 1 message 1 new
>N 1 paco@andaluciajun Tue Apr 13 15:24 13/370 Cita
&
```

Por ejemplo, para visualizar el correo tan sólo tiene que pulsar sobre el número que hay antes del mensaje:

```
& 1
```

para borrarlo ejecutar

```
& d 1
```

para más ayuda pulsar ?

```
& ?
```

para salir

```
& q
```

Si queremos enviar un mensaje a un usuario de otra máquina escribiremos:

```
$ mail usuario@nombre_maquina
```

Para ampliar sobre este comando lo mejor es mirar en la ayuda.

Capítulo 3

Introducción a las redes

> El concepto de trabajo en redes es probablemente tan antiguo como lo es el de las telecomunicaciones. Imagínese por un momento, gente viviendo en la Edad de Piedra, en donde los individuos usen tambores para transmitirse mensajes. Supóngase que un hombre de las cavernas A quiere invitar a otro hombre B a una partida de choques de piedra. Lamentablemente viven tan distantes, que a B le sería imposible escuchar el tambor de A cuando éste lo llame. ¿Qué puede hacer A para remediar esto? Él podría 1) ir caminando al sitio de B, 2) conseguir un tambor más grande, o 3) pedirle a C, quien vive a mitad de camino que reenvíe el mensaje. La tercera elección es denominada Trabajo en Redes. (*Guía de Administración de Redes Segunda Edición*, Olaf Kirch)

3.1. Introducción

Linux, tal como lo conocemos, es posible gracias a "La Red"[1] y "La Red" se ha expandido también en gran parte, gracias a Linux. En este capítulo vamos a estudiar algunos de los programas que nos permiten trabajar en Red con nuestro Guadalinex. Algunos de ellos no podremos probarlos a no ser que tengamos un servidor que nos permita la conexión, así que no todas las cuestiones tratadas aquí se podrán trabajar por igual. Dependerá de cómo esté conectado nuestro equipo y de los servicios de red a los que tengamos acceso para poder trabajar un número mayor o menor de aspectos de este tema. El tema de redes es además lo suficientemente amplio para constituir un curso por sí solo, de hecho lo que vamos a ver aquí son las nociones básicas, es en el curso avanzado de Linux donde se tratan en profundidad las cuestiones y utilidades que aquí esbozamos.

Hay determinados servicios de red que seguramente se están trabajando ya con aplicaciones más que conocidas y comentadas en el curso: el "todoterreno" Mozilla[2], Ximian Evolution, etc. Así que sólo analizaremos aquellos programas o utilidades "novedosas" a estas alturas del curso.

En todos los ejemplos de este capítulo hemos supuesto que la conexión la hacemos con un usuario de nombre `thales` y una máquina llamada `tux.midominio.org`. Para conexiones reales tendremos que adecuar estos datos a los de cada uno en particular.

Comencemos esta sección con:

[1] Nos referimos a Internet

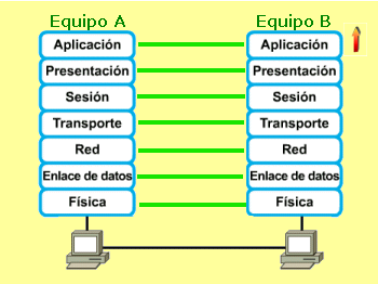
[2] Con Mozilla disponemos de una aplicación muy personalizable para

- Navegar por la web
- Gestionar nuestras cuentas de Correo y Noticias
- Crear páginas web

3.2. Redes TCP/IP: conceptos básicos

3.2.1. Protocolos de Red

Protocolo Es un conjunto de normas y procedimientos para la transmisión de datos que ha de ser observado por los dos extremos de un proceso de comunicaciones (emisor y receptor[3]). El modelo OSI (*Open Systems Interconection*) definido por la ISO (*International Standards Organization*) propone un modelo en capas en donde cada una de las capas se ocupa de una determinada tarea y presenta un interfaz bien definida para sus capas adyacentes (superior e inferior). Las capas del modelo OSI son, desde la superior a la inferior: Aplicación, Presentación, Sesión, Transporte, Red, Enlace y Físico. Entre los protocolos del nivel físico se encuentran Ethernet o Token Ring.



Ejemplos de algunos protocolos son:

IPX/SPX (*Internetwork Packet Exchange*) protocolo para redes Novell Netware.

NetBIOS (*Network Basic Input/Output System*) desarrollado por IBM fué adoptado después por Microsoft.

NetBEUI (*NetBIOS Extended User Interface*) versión mejorada de NetBIOS, es el protocolo predominante en las redes Windows.

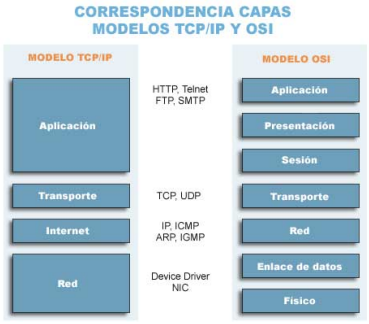
AppleTalk protocolo de comunicación para ordenadores Apple Macintosh.

TCP/IP TCP (*Transmission Control Protocol*, se encuentra en el nivel de transporte), IP (*Internet Protocol*, se encuentra en el nivel de red). En ellos se basan las comunicaciones en Internet. Pertenecen a un conjunto mayor de protocolos que se llama suite o conjunto de protocolos TCP/IP[4]. Se describen a partir de una "pila" de capas que, aunque no tiene una equivalencia total con el modelo OSI, se asemeja a él: capa de subred (tramas Ethernet y direccciones MAC; cable de cobre, de fibra óptica, etc), capa de red/internet (paquetes IP, dirección IP y mensajes ICMP), capa de transporte (puertos de servicio y protocolos TCP y UDP) y capa de aplicación (programas de cliente y de servidor: servidor ftp, cliente telnet, etc).

[3] Y todos los elementos intermedios si los hubiere

[4]

- TCP permite controlar y gestionar el proceso por el que la información se divide en unidades individuales de datos (llamadas paquetes) y cómo son ensamblados de nuevo en la máquina de destino.
- IP se encarga de llevar los paquetes de una máquina a otra haciendo uso de las direcciones IP de las máquinas locales y remotas.



3.2.2. Introducción a las direcciones IP.

Las direcciones del Protocolo Internet (IP) están compuestas por cuatro *bytes* (para la versión IPv4). Estas direcciones desde hace tiempo se dice que están llegando a su agotamiento[5] y la nueva versión del protocolo (IPv6) dispone de un rango de direcciones mucho más elevado[6].

Cada máquina de una red TCP/IP[7] debe tener una dirección IP única. Como Internet es una red TCP/IP, no deben existir dos direcciones IP iguales en ella.

La convención es escribir estas direcciones en la denominada *notación decimal separada por puntos* (*dotted decimal notation*). De esta forma cada byte (octeto) es convertido en un número decimal (en el rango 0-255). Por ejemplo, una dirección válida sería: 192.168.1.5

Cada interfaz[8] de una máquina de la red o de un encaminador (router) tiene una dirección IP. Todas las direcciones dentro de una misma red IP tienen un número de bits en común. Consideremos el siguiente ejemplo[9]:

	Decimal	Binario
Dirección del Host	192.168.1.5	11000000.10101000.00000001.00000101
Máscara de red	255.255.255.0	11111111.11111111.11111111.00000000
Porción de red	192.168.1.	11000000.10101000.00000001.
Porción de Host (máquina)	.5	.00000101
Dirección de Red	192.168.1.0	11000000.10101000.00000001.00000000
Dirección de Difusión (o broadcast)	192.168.1.255	11000000.10101000.00000001.11111111

- Por host nos referimos a una máquina, ya sea un servidor o un puesto de usuario final. En este caso, nuestro querido Linux. En este caso 192.168.1.5
- Al número de bits que comparten todas las direcciones de una red se le llama máscara de red (*netmask*), y su papel es determinar qué direcciones pertenecen a la red y cuáles no. Las máquinas que pertenecen a una misma red IP, pueden comunicarse directamente unas con otras. Para comunicarse con máquinas de una red IP diferente, deben hacerlo mediante mecanismos de interconexión como *routers*, *proxys* o *gateways*. Al ser 255.255.255.0 indica que sólo el último byte se reserva para la porción de host y que los tres primeros bytes se correponden con la porción de red.

[5] Mediante diversas técnicas, como el mejor aprovechamiento de las direcciones o la utilización de direcciones de red privadas dentro de las organizaciones, se ha conseguido frenar este proceso

[6] Como unas cuantas por cada metro cuadrado de la tierra, posibles gracias a 128 bits de direccionamiento

[7] De igual manera que cada persona en España tiene un número de DNI

[8] Por *interfaz de red* entendemos el dispositivo que nos une a la red, por ejemplo una tarjeta de red.

[9] Recordemos que para pasar de binario a decimal sólo tenemos que sumar las potencias de dos de la forma adecuada. Por ejemplo:
$00100101_2 = 0 \cdot 128 + 0 \cdot 64 + 1 \cdot 32 + 0 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 32 + 4 + 1 = 37_{10}$

- A la parte de la dirección IP que es común a todas las direcciones que se encuentran en una red IP, se le llama la *porción de la red*.
- Los bits restantes son llamados *porción de la máquina*.
- La dirección de red es siempre el menor número de dirección dentro del rango de la red y siempre tiene la porción de máquina codificada toda con ceros. En el ejemplo, 192.168.1.0.
- La dirección *de* difusión (*broadcast*) es una dirección especial a la que escucha cada máquina en la red IP además de a la suya propia. Esta dirección es a la que se envían los paquetes de datos si se supone que todas las máquinas de la red lo deben recibir. Ciertos tipos de datos, como la información de encaminamiento y los mensajes de aviso son transmitidos a la dirección de difusión para que cada estación en la red pueda recibirlo simultáneamente. Para ello se utiliza la dirección más alta posible en la red, conseguida con la porción de red a la que se añade el resto de bits (de la porción de host) todos con valor a uno (1). En el ejemplo anterior sería 192.168.1.255.

Por razones administrativas, durante el desarrollo inicial del protocolo IP se formaron, de forma un poco arbitraria, algunos grupos de direcciones como redes, y estas redes se agruparon en las llamadas *clases*. Estas clases proporcionan un cierto número de redes de tamaño estándar que pueden ser reservadas. Los rangos reservados son[10]:

Clase de red	Máscara de red	Direcciones de red desde-hasta
A	255.0.0.0	0.0.0.0-127.255.255.255
B	255.255.0.0	128.0.0.0-191.255.255.255
C	255.255.255.0	192.0.0.0-223.255.255.255
Multicast	240.0.0.0	224.0.0.0-239.255.255.255

Las direcciones que usemos dependen de lo que se vaya a hacer exactamente.

Redes privadas no conectadas a Internet

Si estamos construyendo una red privada y no tenemos intención de conectar nunca esa red a Internet entonces podríamos elegir las direcciones que queramos, pues no vamos a colisionar con nadie.

Por razones de seguridad y consistencia, se han reservado algunas direcciones IP de red específicamente para este propósito. Están descritas en el RFC1597 y son las que siguen:

Direcciones reservadas para redes privadas		
Clase de red	Máscara de red	Direcciones de red desde-hasta
A	255.0.0.0	10.0.0.0 - 10.255.255.255
B	255.255.0.0	172.16.0.0 - 172.31.255.255
C	255.255.255.0	192.168.0.0 - 192.168.255.255

Estas direcciones no se corresponden con las de ninguna máquina en Internet y no se encaminarán a través de los "routers" de Internet. Podremos utilizarlas de forma interna, con la ventaja

[10]Esta división es cada vez menos utilizada y se expone por razones históricas

de que si conectamos mediante un proveedor a Internet, nuestras direcciones no coincidirán con las de ninguna máquina de Internet.

En este caso, puede que haya multitud de equipos en redes distintas con estas mismas direcciones IP[11] pero como estas direcciones no se "rutean" no hay por qué coordinar su uso con las agencias de registro. En el caso de que se conecten a Internet, se habilitan mecanismos como la traducción de direccciones (NAT[12]) o el uso de intermediarios (proxys) para que las direcciones que salgan a Internet sean direcciones válidas.

3.3. Guadalinex en una red IP

Si deseamos instalar una máquina Linux en una red IP existente entonces deberíamos tener clara la siguiente información:

- Dirección IP del Host
- Dirección IP de la red
- Dirección IP de broadcast
- Máscara de red IP[13]
- Dirección del encaminador (router)
- Dirección del Servidor de Nombres de Dominio (DNS)

En nuestros ejemplos, realizaremos la configuración de una máquina en la red 192.168.1.0[14], con dirección 192.168.1.5 y cuya máscara de red sea 255.255.255.0[15]. El rango de direcciones en la red será desde la 192.168.1.0 a la 192.168.1.255. Tendremos en total hasta 254 direcciones posibles en nuestra red local de las 256 que nos permite el último byte (2^8), porque dos direcciones, como hemos visto, tienen un propósito específico (la dirección de la propia red y la de *broadcast*).

La 192.168.1.0 es la dirección de la red y por tanto no utilizable como dirección por ninguna máquina.

La 192.168.1.255 es la dirección de broadcast y sirve para referirse a todas las máquinas de la red.

Utilizaremos la dirección 192.168.1.1 como la dirección del router que nos sacará hacia Internet, que también se suele denominar *gateway*.

Esa es la vista lógica de nuestra red. La vista física será, en general, similar a una de las dos que siguen

- Un hub (o concentrador)[16], que forma nuestro segmento de red y al que irán conectados los ordenadores de la red, el servidor linux y el router. Además, el router dispondrá de una salida ADSL/RDSI hacia el exterior.

[11] Dentro de cada red, cada equipo tendrá direcciones IP diferentes: dos personas que no viajan (direcciones IP que no se "rutean") pueden tener iguales DNI en estados diferentes pero nunca dentro del mismo país.

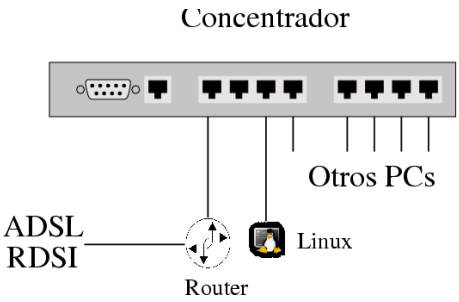
[12] *Network Address Translation* o Traducción de Direcciones de Red

[13] Conociendo la dirección IP de la máquina (host) y la máscara de red, podemos conocer la dirección de la red y la de broadcast. Prueba a hacerlo con la dirección de máquina 150.214.5.11 y la máscara de red 255.255.255.0

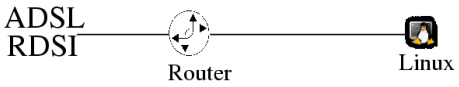
[14] Al tomar esta dirección para la red (véase la tabla anterior) nos garantizamos que si accedemos a Internet mediante un proveedor de acceso no tendremos problemas de encaminamiento.

[15] Se puede escribir de la forma 192.168.1.0/24, esta es otra notación para las redes. Significa que la dirección de la red es la 192.168.1.0 y la máscara correspondiente son 24 bits empezando por la izquierda (255.255.255.0 ó 8+8+8+0 ó /24)

[16] Mejor un switch (o conmutador).



- Nuestro Linux se conecta directamente al router, y este se encargará de que salgamos hacia el exterior.



En la mayoría de los casos, los dispositivos de red se crean automáticamente por el controlador de dispositivos mientras se inicia y localiza el hardware. Por ejemplo, el controlador Ethernet (para nuestra tarjeta de red local) crea interfaces `eth0`, `eth1`... secuencialmente según va encontrando tarjetas Ethernet.

3.3.1. Configuración del interfaz de red


Para saber cómo trabajar sobre estos comandos en modo gráfico véase 3.3.4 en la página 54.

Cuando hablamos de configurar una interfaz de red nos referimos al proceso de asignar direcciones apropiadas a un dispositivo (tarjeta de red) y asignar valores adecuados a otros parámetros configurables. El programa genérico usado para hacer esto es la orden `ifconfig` (*interface configuration*).

Si lo hacemos mediante línea de comandos, sería una orden como:

```
# ifconfig eth0 192.168.1.5 netmask 255.255.255.0 up
```

En este caso estoy configurando la interfaz Ethernet `eth0`, con dirección IP 192.168.1.5 y máscara de red 255.255.255.0. El `up` del final de la orden le dice al interfaz que debería activarse, pero normalmente se puede omitir, ya que es el valor por defecto. [17]

Mediante el comando:

```
$/sbin/ifconfig -a
```

podemos ver las interfaces que tenemos configuradas:

```
eth0      Link encap:Ethernet  HWaddr 00:0C:29:21:D2:6F
          inet addr:192.168.1.5  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe21:d26f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45 errors:0 dropped:0 overruns:0 frame:0
          TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

[17] Para desactivar una interfaz, simplemente hay que ejecutar

```
#ifconfig eth0 down
```

```
          RX bytes:5347 (5.2 KiB)  TX bytes:7778 (7.5 KiB)
          Interrupt:10 Base address:0x1080
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

que nos dice que la interfaz `eth0` tiene la dirección hardware 00:0C:29:21:D2:6F (en hexadecimal). Ésta es una dirección de la propia tarjeta y cada fabricante se ocupa de que sea única en el mundo. Se denomina dirección MAC. Es una dirección física, que normalmente no se puede cambiar; al contrario que la dirección IP, que es una dirección lógica y puede modificarse. Cuando estamos en una red de área local (LAN) son éstas las direcciones mediante las que se comunican las máquinas. Existe un protocolo llamado ARP (Protocolo de resolución de direcciones) que se encarga de preguntar y traducir entre direcciones IP y direcciones MAC.

Comprobamos si se ha configurado con la dirección de red, broadcast y máscara de red deseadas. Si se encuentra en modo UP es que está habilitada.

La interfaz `lo` (loopback) es una interfaz virtual que se refiere a nuestra propia máquina, sin salir por ninguna interfaz de red, sólo para uso interno. Su dirección es siempre 127.0.0.1.

Si disponemos de un router de IP 192.168.1.1 y ejecutamos desde nuestra máquina Linux[18]

```
$ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=0.9 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.8 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.9 ms

--- ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.8/0.8/0.9 ms
```

Estamos mandando paquetes por la red al router. Si el resultado ha sido como el anterior, podemos llegar a éste de forma satisfactoria.

Si por el contrario, la salida nos indica que no puede llegar (100 % *packet loss*) al router, debemos revisar nuestras conexiones de red.

```
$ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) from 0.0.0.0 : 56(84) bytes of data.

--- 192.168.1.1 ping statistics ---
11 packets transmitted, 0 packets received, 100% packet loss
```

El comando `ping` es una utilidad de red muy práctica. Trabaja de la siguiente forma: Nuestra máquina envía paquetes de datos a la dirección de destino (en este caso 192.168.1.1) que, automáticamente, nos responde como si fuera un *boomerang*. Si todo es correcto, nos llegarán los paquetes de vuelta a nuestra máquina.

Configurando el encaminamiento (routing).

> *El Encaminamiento IP es el proceso por el que una máquina decide por dónde dirigir un paquete IP que haya recibido.*

[18] Se detiene el comando con las teclas `[CTRL]+[C]`.

Ya tenemos nuestras direcciones IP asignadas. Continuemos ¿Cómo funciona el encaminamiento? Cada máquina tiene una lista de reglas, llamada tabla de encaminamiento (o tabla de *routing*). Esta tabla contiene columnas que suelen contener al menos tres campos:

- Una dirección de red de destino,
- El nombre de la interfaz a la que se va a encaminar el paquete, y
- Opcionalmente, la dirección IP de otra máquina que cogerá el paquete en su siguiente paso a través de la red si es que no podemos llegar directamente a ella.

En Linux se puede ver esta tabla usando las órdenes:

```
# /sbin/route -n
```

o

```
# /bin/netstat -nr
Kernel IP routing table
Destination  Gateway     Genmask    Flags  Metric  Ref  Use  Iface
127.0.0.0    0.0.0.0   255.0.0.0      U      0      0    0    lo
192.168.1.0  0.0.0.0   255.255.255.0  U      0      0    0   eth0
0.0.0.0    192.168.1.1   0.0.0.0      UG     0      0    0   eth0
```

Veamos su significado. Los paquetes con destino a la red 127.0.0.0/255.0.0.0 (ponemos la dirección de la red seguida de su máscara) deben ser enviados a la interfaz `lo`. Los paquetes con destino a la red 192.168.1.0/255.255.255.0 (por ejemplo la dirección 192.168.1.1) deben ser enviados al interfaz `eth0` (la primera tarjeta de red). Para el resto de redes que no están especificadas explícitamente (especificado como la dirección 0.0.0.0) debemos mandar los paquetes a través de un intermediario (el router 192.168.1.1) para que sea éste el que se ocupe de mandarlos a su destino. Es decir, le decimos "yo no sé cómo mandar paquetes a la dirección 150.214.4.34, ocúpate tú de ello". En cierto modo es lo más fácil y, además, como ese es el trabajo del router, lo hará bien.

El comando para la ruta al interfaz de loopback se crea automáticamente. Para crear la ruta a nuestra red local utilizamos el comando:

```
#/sbin/route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

que quiere decir: para la red 192.168.1.0 con máscara 255.255.255.0 dirige los paquetes por el interfaz `eth0`. Esta ruta le permitirá establecer conexiones IP directamente con todas las estaciones de su segmento Ethernet; aquel accesible directamente mediante la tarjeta de red `eth0`.

Para crear la ruta por defecto, introducimos:

```
#/sbin/route add default gw 192.168.1.1
```

Añade la ruta por defecto[19] (cuando no encontramos una ruta específica para salir) y manda los paquetes a la dirección 192.168.1.1, que ya se encargará de dar salida a los paquetes[20].

Para comprobar que funciona correctamente podemos hacer `ping` a una máquina de fuera de nuestra red y ver si realmente los paquetes pueden salir al exterior.

`$ping 150.214.4.34`

Si nos llegan los paquetes de vuelta, estupendo. En caso contrario, debemos asegurarnos de que hemos realizado correctamente la configuración de las rutas, y por si acaso, reiniciado nuestra máquina.

[19] Denominada "default"

[20] Si no tenemos router ni ninguna forma de conectarnos a Internet (cada vez más inusual), no especificaremos esta ruta.

Configuración del sistema DNS

La función del *sistema de resolución de nombres* (DNS) es proporcionar un servicio para convertir las denominaciones "amistosas para el hombre" de las máquinas como `thales.cica.es` a direcciones IP "amigables para la máquina" como 150.214.5.10.

Los nombres de dominio de Internet tienen una estructura en árbol invertido. Un dominio puede ser fragmentado en subdominios. Un *dominio de nivel superior*[21] (*toplevel domain*) es un dominio especificado en la parte más a la derecha del nombre de máquina o dominio. Algunos ejemplos de los más comunes son:

COM Organizaciones Comerciales

EDU Organizaciones Educativas[22]

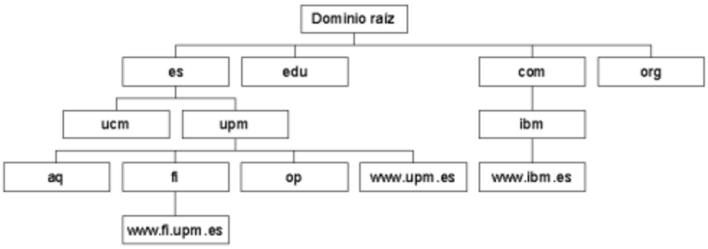
GOV Organizaciones Gubernamentales

MIL Organizaciones Militares

ORG Otras organizaciones, normalmente sin ánimo de lucro

NET Organizaciones relacionadas con InterNet

Designador de País éstos son códigos de dos letras que representan a un país en particular (**es**=España).



Cada uno de estos dominios de nivel superior tiene subdominios. Los dominios de nivel superior basados en el nombre de un país suelen estar divididos en subdominios basados en `com`, `edu`, `gov`, `mil` y `org`.

↬ Por ejemplo: `com.au`, `gov.au`, para las organizaciones comerciales y gubernamentales en Australia respectivamente.

El siguiente nivel de división suele representar el nombre de la organización. Los siguientes subdominios varían, a menudo basándose en el siguiente nivel en la estructura departamental de la organización a la que pertenecen, pero pueden estarlo en cualquier criterio considerado razonable y con significado claro para los administradores de la red de la organización.

↬ Por ejemplo, `cec.juntadeandalucia.es` (Consejería de Educación y Ciencia, dentro de la Junta de Andalucía y dentro de España)

La parte más a la izquierda de un nombre de máquina es el nombre único asignado a la máquina, y es llamada *hostname*, la porción del nombre a la derecha del nombre de la máquina es el *domainname* (nombre de dominio), y el nombre completo es llamado *Fully Qualified Domain Name* (Nombre de Dominio Completamente Cualificado).

↬ Por ejemplo para el nombre completo `thales.cica.es`, el hostname sería `thales` y el domainname sería `cica.es`.

[21] Están especificados en el RFC-920
[22] Solo para EE.UU

El software de resolución de nombres proporciona el servicio de traducción haciendo consultas a un `Servidor de Nombres de Dominio` (*Domain Name Server*), por lo que deberá saber la dirección IP del servidor de nombres (*nameserver*) que vaya a usar.

También podemos hacer resoluciones de nombres de forma local en nuestra máquina. El fichero `/etc/hosts` es donde se pone el nombre y dirección IP de las máquinas locales. En este fichero, ponemos la equivalencia entre el nombre de una máquina y su dirección IP. La desventaja de este método frente a los servidores de nombres es que habrá que poner el fichero al día si la dirección de alguna máquina cambia. En un sistema normal, las únicas entradas que suelen aparecer son la interfaz de loopback (prueba en bucle) y el nombre de la máquina local[23].

```
#more /etc/hosts
127.0.0.1 localhost  localhost.localdomain
192.168.1.5 linux linux.cec.juntadeandalucia.es mailhost
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Se puede especificar más de un nombre de máquina por línea, normalmente el nombre de la máquina, su nombre completo con el dominio y posibles alias (o nombres alternativos) de la máquina.

El fichero `/etc/resolv.conf` es el de configuración principal del código de resolución de nombres. Su formato es bastante simple. Es un fichero de texto con una palabra clave por línea. Hay tres palabras clave de uso frecuente, que son:

domain esta palabra clave especifica el nombre de dominio local.

search ésta especifica una lista de dominios alternativos para completar el nombre de una máquina.

nameserver ésta, que puede utilizarse hasta tres veces, especifica una dirección IP de un servidor de nombres de dominio para consultarlo cuando se resuelvan nombres.

Por ejemplo, nuestro `/etc/resolv.conf` podría parecerse a éste:

```
domain cec.juntadeandalucia.es
search cec.juntadeandalucia.es juntadeandalucia.es
nameserver 195.235.113.3
nameserver 80.58.0.33
nameserver 150.214.4.34
```

Las palabras clave domain y search no pueden aparecer juntas. Cogería solamente la última.

Para comprobar que realmente resolvemos los nombres, podemos hacer

```
$ping thales.cica.es
```

Lo primero que hace la máquina será traducir el nombre `thales.cica.es` a su dirección IP que es con la que trabajan las tarjetas de red. Después mandará los paquetes a la dirección indicada, a través del router si no estamos en la misma red.

Una utilidad interesante de diagnóstico[24] es el comando:

[23] El resto de líneas se corresponden con entradas para la nueva versión del protocolo, la IPV6 y las dejamos para el año que viene -:) .

[24] Otra utilidad interesante es el comando `tcpdump` que escucha en una interfaz de red todo el tráfico que pasa por ese segmento de red.

traceroute que nos permite ver por los sitios por los que van pasando los paquetes en el camino hasta su dirección de destino.

Por ejemplo,

```
$/usr/sbin/traceroute metalab.unc.edu
traceroute to metalab.unc.edu (152.19.254.81), 30 hops max, 38 byte packets
1 192.168.3.254 (192.168.3.254) 1.197 ms 1.085 ms 1.050 ms
2 192.168.254.5 (192.168.254.5) 45.165 ms 45.314 ms 45.164 ms
3 obsgate (192.168.2.254) 48.205 ms 48.170 ms 48.074 ms
4 obsposix (160.124.182.254) 46.117 ms 46.064 ms 45.999 ms
5 cismpjhb.posix.co.za (160.124.255.193) 451.886 ms 71.549 ms 173.321 ms
6 cisap1.posix.co.za (160.124.112.1) 274.834 ms 147.251 ms 400.654 ms
7 saix.posix.co.za (160.124.255.6) 187.402 ms 325.030 ms 628.576 ms
8 ndf-core1.gt.saix.net (196.25.253.1) 252.558 ms 186.256 ms 255.805 ms
9 ny-core.saix.net (196.25.0.238) 497.273 ms 454.531 ms 639.795 ms
10 bordercore6-serial5-0-0-26.WestOrange.cw.net (166.48.144.105) 595.755 ms 595.174 ms *
11 corerouter1.WestOrange.cw.net (204.70.9.138) 490.845 ms 698.483 ms 1029.369 ms
12 core6.Washington.cw.net (204.70.4.113) 580.971 ms 893.481 ms 730.608 ms
13 204.70.10.182 (204.70.10.182) 644.070 ms 726.363 ms 639.942 ms
14 mae-brdr-01.inet.qwest.net (205.171.4.201) 767.783 ms * *
15 * * *
16 * wdc-core-03.inet.qwest.net (205.171.24.69) 779.546 ms 898.371 ms
17 atl-core-02.inet.qwest.net (205.171.5.243) 894.553 ms 689.472 ms *
18 atl-edge-05.inet.qwest.net (205.171.21.54) 735.810 ms 784.461 ms 789.592 ms
19 * * *
20 * * unc-gw.ncren.net (128.109.190.2) 889.257 ms
21 unc-gw.ncren.net (128.109.190.2) 646.569 ms 780.000 ms *
22 * helios.oit.unc.edu (152.2.22.3) 600.558 ms 839.135 ms
```

Nos indica las redes y routers por los que atraviesan los paquetes desde la máquina en la que se ha lanzado el comando hasta la máquina `metalab.unc.edu`.

3.3.2. Configuración gráfica.

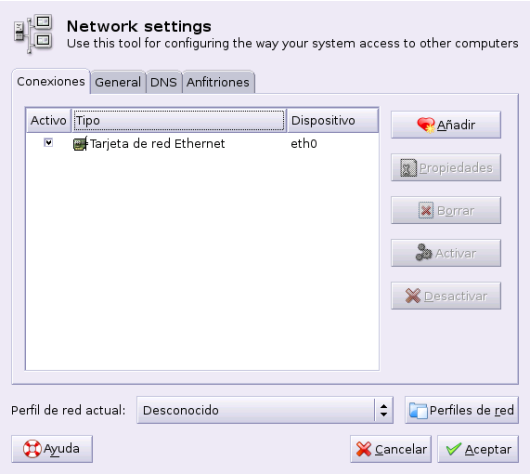
Para un sistema Guadalinex podemos configurar la red en modo gráfico lanzando desde Gnome[25]

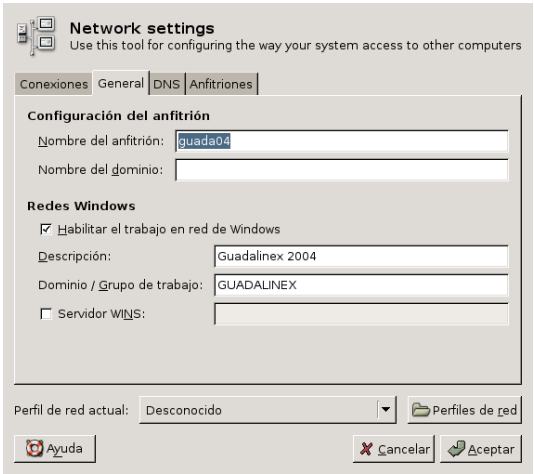
 **Aplicaciones→Configuración→Sistema→Red**

o directamente desde una **xterm**:

```
# network-admin &
```

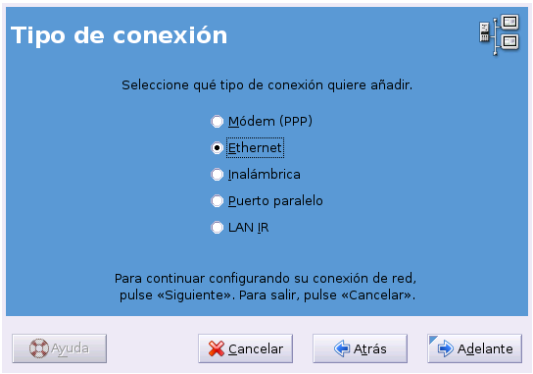
[25] En el apartado 6.3.1 de la primera entrega estudiamos este tema al configurar la conexión a internet usando un router ADSL.

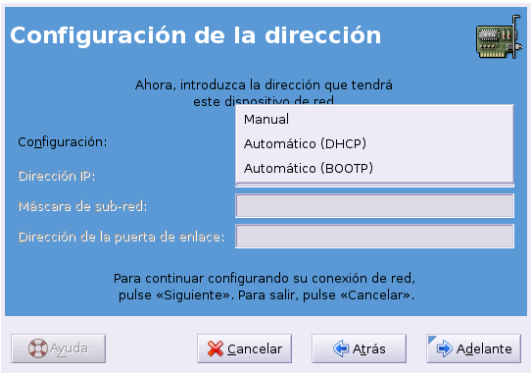




Si pulsamos sobre la pestaña **Conexiones** podremos optar por configurar nuestra red. Lo usual es que la tarjeta haya sido detectada y configurada en el arranque/instalación y desde esta ventana podremos editarla.

Si no es así, pulsando sobre **Añadir** podemos configurar nuestra nueva interfaz de red. Seleccionamos **Conexión Ethernet** y debemos elegir la tarjeta correspondiente

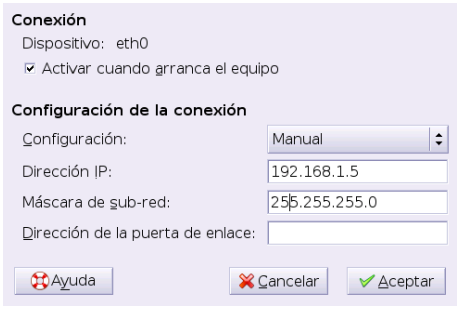




Tendremos la posibilidad de permitir que la configuración se obtenga de un servidor de alguno de estos protocolos (DHCP, BOOTP) que se la proporcionará al arrancar, o bien, si optamos por mantener la **Configuración Manual** podremos introducir la dirección IP (192.168.1.5), la máscara de red (255.255.255.0) y la **Dirección de la Puerta de enlace**[26] (192.168.1.1).

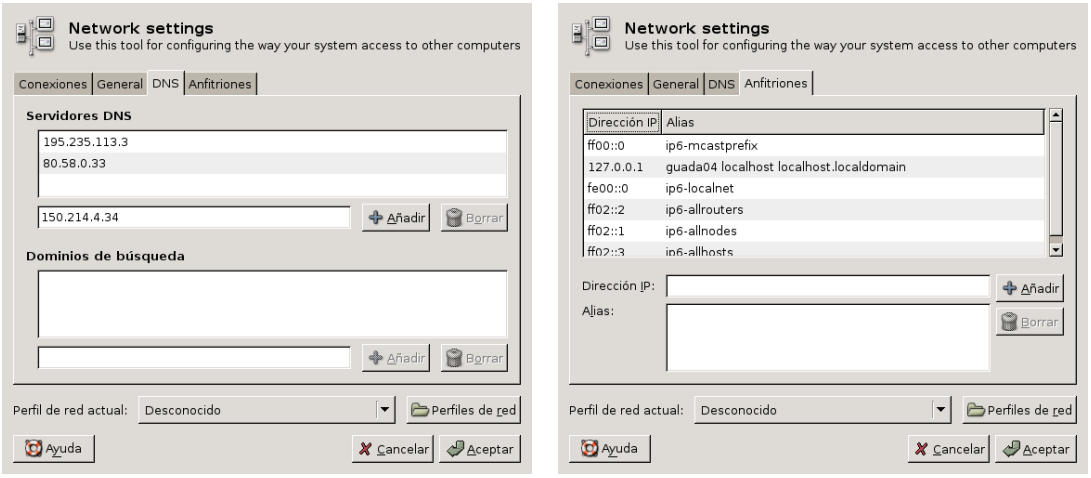
Si en la ventana principal del programa nos situamos sobre un dispositivo ya instalado en el sistema y pulsamos sobre el botón **Propiedades,** podemos cambiar las opciones anteriores o acceder a otras posibilidades de configuración. Deberíamos dejar marcada la opción de activar el interfaz en el arranque (**Activar cuando arranca la computadora**):

[26]Si tenemos un router ADSL u otro Linux que hace de pasarela, debemos poner aquí su dirección IP para que podamos salir al exterior. Si no, debemos dejar esta casilla en blanco.



Configuración del sistema DNS

Pulsamos en la pestaña **DNS** e introducimos las IP de nuestros servidores de nombres. Se trata de rellenar los datos necesarios en estos campos, necesitamos conocer el nombre de nuestro servidor de Internet, que lo escribiremos en el campo **Nombre del dominio** (no es necesario) y los números DNS de nuestros servidores de nombres. En el caso de la red de ejemplo con la que estamos trabajando escribiríamos como DNS 195.235.113.3, 80.58.0.33 y 150.214.4.34, que serían los DNS primario, secundario y terciario.



Si deseamos resolver nombres localmente (modificar el fichero `/etc/hosts` en modo gráfico) pulsaremos sobre la pestaña: **Anfitriones**. Podemos modificar las entradas de ese fichero o añadir más pulsando sobre **Añadir**

Llegados a este punto, después de **Aceptar** y volver a la pestaña **General**, nos garantizamos que esté activo el interfaz verificando que esté marcada la casilla **Estado** y cerramos. Para comprobar que realmente resolvemos los nombres, podemos hacer

```
$ping thales.cica.es
```

Lo primero que hace la máquina será traducir el nombre `thales.cica.es` a su dirección IP que es con la que trabajan las tarjetas de red. Después mandará los paquetes a la dirección indicada, a través del router si no estamos en la misma red.

A modo de resumen

Para un sistema Guadalinex, la configuración que hemos hecho se guardaría en el directorio `/etc/network/`, contiene los ficheros que leerá el sistema al arrancar y activar la red. El contenido del fichero `/etc/network/interfaces` será similar a[27]:

[27] Para conocer las opciones de este fichero

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
```

Listado 3.1: /etc/network/interfaces

Si modificamos con un editor este fichero y deseamos releer la configuración ejecutaremos

```
# /etc/init.d/networking restart
```

3.3.3. Configuración: servidores y servicios de red

Los servidores de red y los servicios son aquellos programas que permiten a un usuario remoto hacer uso de nuestra máquina Linux. Los programas servidores (*demonios*) escuchan en los puertos de red. A cada aplicación (FTP o Telnet, por ejemplo) se le asigna un único número, a modo de casillero, llamado PUERTO. Cuando se produce una solicitud de conexión a dicho puerto, se ejecutará la aplicación correspondiente. Los puertos de red son el medio de llegar a un servicio determinado en una máquina en concreto, y es así como un servidor conoce la diferencia entre una conexión telnet y otra de FTP que le lleguen.

Algunas asignaciones estándar de puertos son[28]

Servicio o Aplicación	Puerto
File Transfer Protocol (FTP)	21
Secure SHell (ssh)	22
Telnet	23
Simple Mail Transfer Protocol (SMTP)	25
Gopher	70
Finger	79
Hypertext Transfer Protocol (HTTP)	80
Network News Transfer Protocol (NNTP)	119
Post Office Protocol (POP3)	110

↬ Por ejemplo, cuando nuestro navegador Web (Mozilla) pide una página web del servidor de los cursos (`mileto.cica.es`) dirige su solicitud al puerto 80. De esa forma, el servidor mileto, sabe que se trata de una solicitud de página Web y pasa el control de la llamada al programa encargado de servir las páginas Web (en ese caso Apache) que procede a tomar el control de la solicitud.

Hay dos modos de operación para los demonios de red:

autónomo (standalone) el programa demonio de red escucha en el puerto de red asignado y, cuando llega una conexión, se ocupa él mismo de dar el servicio de red. En este modo suele trabajar, por ejemplo, un servidor web (como Apache).

esclavo del servidor inetd (o xinetd) `inetd` se trata de un súper-demonio (siempre están en ejecución) cuya finalidad es estar a la espera de que se produzca alguna solicitud de conexión del exterior. Si esto pasa `inetd` analiza esa solicitud determinando qué servicio le están solicitando y le pasa el control a dicho servicio[29].

`$man interfaces`

El fichero `/etc/hostname` contendrá el nombre de la máquina.

[28] Véase la sección siguiente: Fichero `/etc/services` en la página 53

[29] Su funcionamiento sería el siguiente. Escucha en una serie de puertos para los que se ha configurado. En caso de recibir una petición por uno de ellos, sabe a qué programa debe llamar. Si es el 23, llama al servidor de `telnet`;

inetd

Por defecto, en Guadalinex `inetd` viene deshabilitado. Para habilitarlo, lanzamos la secuencia de menús **Aplicaciones→ Configuración →Servicios** (o desde la línea de comandos `#services-admin`) y marcar la casilla de activación.

`/etc/inetd.conf` es el fichero de configuración para el demonio servidor `inetd`. Su función es la de almacenar la información relativa a lo que `inetd` debe hacer cuando recibe una petición de conexión a un servicio en particular. Para cada servicio que deseemos que acepte conexiones de red, debemos decirle a `inetd` qué demonio servidor ejecutar, y cómo ha de hacerlo.

Es un fichero de texto en el que cada línea describe un servicio. Al igual que en otros ficheros, cualquier texto precedido del carácter # se considera un comentario[30].

- Si no queremos que nuestra máquina proporcione un servicio determinado, debemos comentar la línea en este fichero, precediéndola del carácter `#`.
- Para activar un servicio procederemos de manera inversa (si la línea está comentada) o añadiendo la línea adecuada (en general se añadirá al instalar el servicio).

Para que los cambios tengan efecto podemos[31]:

1. Decirle que vuelva a cargar el fichero de configuración

   ```
   /etc/init.d/inetd reload
   ```

2. Reiniciar el superdemonio

   ```
   /etc/init.d/inetd restart
   ```

Fichero /etc/services

El fichero `/etc/services` es una base de datos sencilla, que asocia un nombre que nosotros podamos entender, con un puerto de servicio de la máquina. Su formato es bastante simple. Es un fichero de texto en el que cada línea representa una entrada a la base de datos. Cada entrada comprende tres campos separados por cualquier número de espacios en blanco (espacio o tabulador).

si es al 22, llama al servidor de `ssh`. Por ejemplo, si la máquina remota solicita una transferencia de fichero por el puerto 21, le pasará la solicitud al servidor de ftp

[30] Cada línea contiene siete campos separados por cualquier número de espacios en blanco (espacio o tabulador). La explicación de esos campos escapa a las pretensiones de este curso.

[31] También se activarán si reiniciamos el ordenador.

Los campos son:
`nombre puerto/protocolo sobrenombres # comentario`

↪ Un ejemplo de líneas en el fichero `/etc/services` .

```
smtp  25/tcp   mail #servicio de correo
```

Al servicio `smtp`[32] asociamos el puerto 25, por protocolo `tcp` y le asignamos un alias `mail`.

3.3.4. Gnome-netinfo

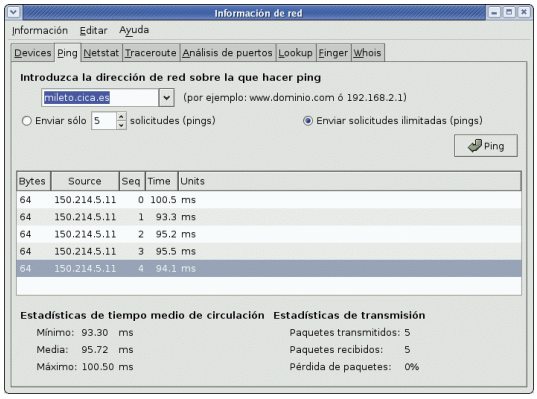
Es un programa muy interesante, se trata de interfaz gráfico de usuario para las utilidades más comunes de red. Para entender mejor su significado os remitimos a 3.3.1 en la página 44 cuando se explican en modo comando.

Podemos ejecutarlo con la cadena de menús **Aplicaciones→Internet→Gnome Network** o bien desde un xterm con:

```
$gnome-netinfo
```

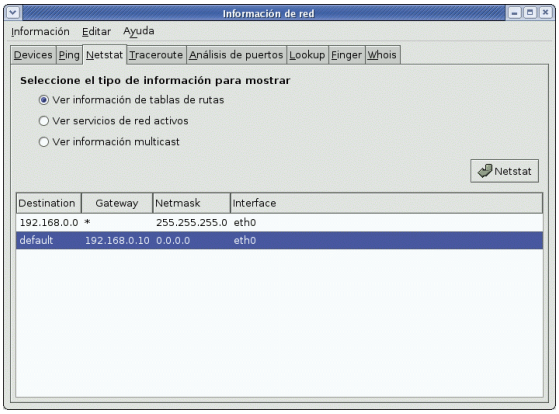
¿Qué significado tiene cada pestaña?

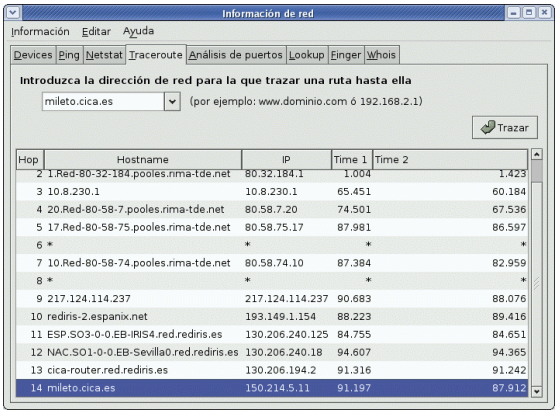
Devices desde aquí obtenemos información detallada sobre nuestra interfaz de red.



Ping Para hacer un `ping` a una máquina remota, ya lo hemos estudiado.

Netstat Muestra las conexiones de red, tabla de rutas, estadísticas de uso de la red, ...



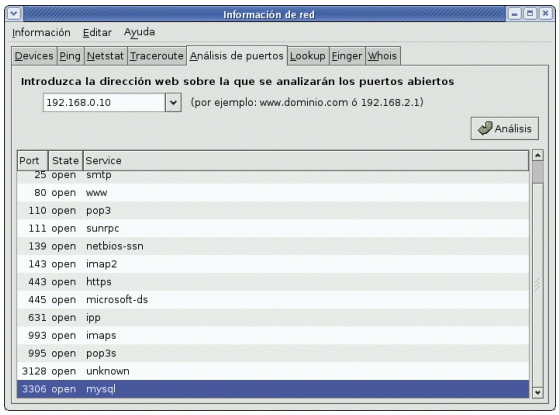


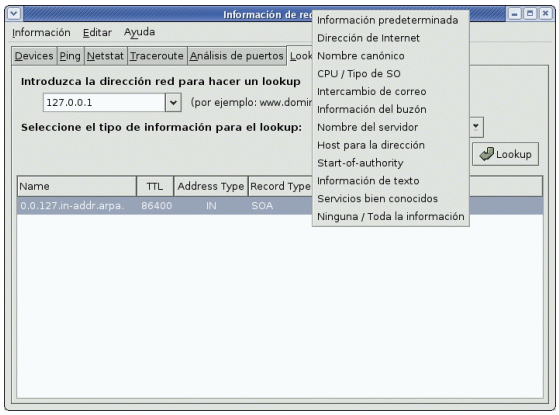
Otra utilidad de diagnóstico es el comando:

[32] `http://es.wikipedia.org/wiki/SMTP`

traceroute que nos permite ver los sitios por los que van pasando los paquetes en el camino hasta su dirección de destino. Por ejemplo, con los datos de la captura nos indica las redes y routers por los que atraviesan los paquetes desde la máquina en la que se ha lanzado el comando hasta la máquina `mileto.cica.es`

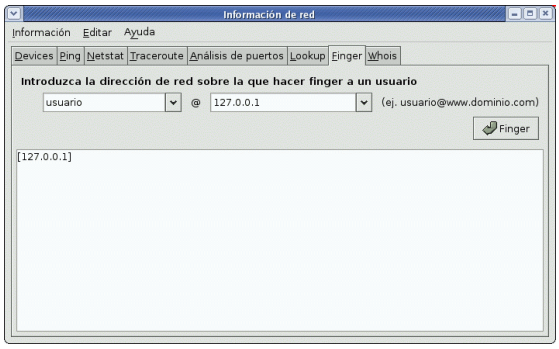
Análisis de puertos se trata de un escáner de puertos, no debemos usarlo "contra" máquinas no conocidas. Su uso se debería restringir a comprobar la seguridad de nuestro sistema para testear qué puertos son los que tenemos abiertos.

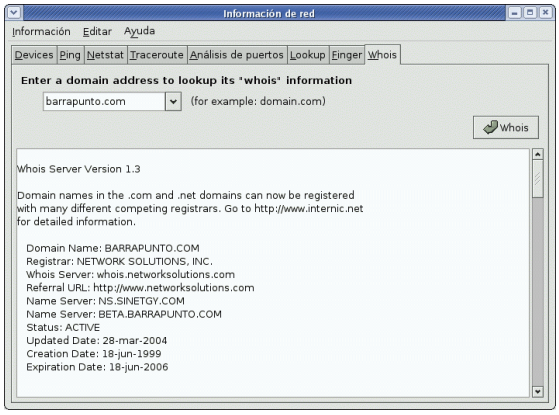




Lookup para obtener el nombre de una máquina conocida su IP, para obtener la IP si conocemos su nombre, etc.

Finger permite mostrar (si el servicio está activo) información sobre los usuarios de un sistema (en este caso[33] 127.0.0.1).





Whois permite obtener información sobre los dominios registrados: nombre, empresa que lo registró, etc.

[33] Denominada de bucle local (*loopback*), es una dirección especial, que utiliza la propia máquina para acceder a sus procesos locales.

Capítulo 4

En Red-ando con Guadalinex

> Cuando trabajas con Linux estás ante un sistema operativo orientado al trabajo con redes de ordenadores. ¿Qué nos empuja a poder afirmarlo tan categóricamente? Ya te darás cuenta poco a poco. (*Manual Avanzado de Linux* de Raúl Montero Rivero, Ed. Anaya)
>
> Las computadoras son mucho más útiles cuando están conectadas a una red, pero, desgraciadamente, estas redes hacen que las computadoras estén expuestas a un gran número de accesos no autorizados, y los sistemas Linux no son inmunes a este tipo de actividades. *(Hackers en Linux*, Brian Hatch y otros)

4.1. Servicio http

El protocolo de transferencia de hipertexto (HTTP) establece la forma en la que los clientes Web (Mozilla, Firefox, Lynx, ...) solicitan páginas web a los servidores y la forma en que estos últimos las sirven. Los clientes (navagadores) se encargan de mostrar de forma adecuada la página web pedida, en general escrita en HTML, dándole el formato adecuado: nos muestran tablas, efectos sobre el texto, objetos insertados como gráficos, sonido, ... Mientras que el servidor web es el encargado de alojar y servir las páginas web que solicita el navegador direccionadas por un URL (*Uniform Resource Locator*, es decir, localizador uniforme de recurso). Los URL´s son las "direcciones" de los recursos de Internet y su estructura básica es la siguiente:

protocolo://host/ruta

protocolo tipo de servicio o protocolo utilizado (HTTP, FTP, etc.). Para acceder a páginas web es HTTP.

host Nombre de dominio completamente cualificado (por ejemplo thales.cica.es), nombre del host que puede ser resuelto por el servidor (thales, sabiendo que se encuentra en el dominio cica.es) o dirección IP.

ruta_de_acceso. Es opcional y hace referencia a un recurso localizado dentro del servidor: un directorio, un fichero, una imagen, etc.

Ejemplos de URL´s :

- http ://www.iesmurgi.org
- http ://thales.cica.es/almeria/index.php
- ftp ://ftp.rediris.es/pub

4.1.1. Como cliente

Otros navegadores Web

Hay un par de "perlas" instaladas en nuestro sistema que muestran su valía cuando estamos obligados a trabajar con un navegador web que no consuma recursos del sistema (por ejemplo cuando navegamos por la red usando otra máquina a la que a su vez nos hemos conectado en red), se trata de `lynx`. Lynx es un navegador Web un tanto especial, si queremos usarlo para buscar algo en Google, una vez conectados sólo hay que escribir[1]

```
# lynx http://www.google.es
```

Si deseamos visitar otra página escribiremos `g` y después la URL a la que deseamos acceder. Para movernos por él hemos de usar los cursores y la barra espaciadora y para salir la letra `q`.

Otro navegador, más evolucionado que el anterior y tambien en modo texto[2], `links`, para ejecutarlo

```
$links
```

Tiene un menú de contexto al que podemos acceder con la tecla de función **F9**[3]. En primer lugar deberíamos ponerlo en castellano desde el menú **Setup**

[1] También podemos iniciarlo sin escribir la dirección y acceder a ella una vez en el programa usando la tecla `g`

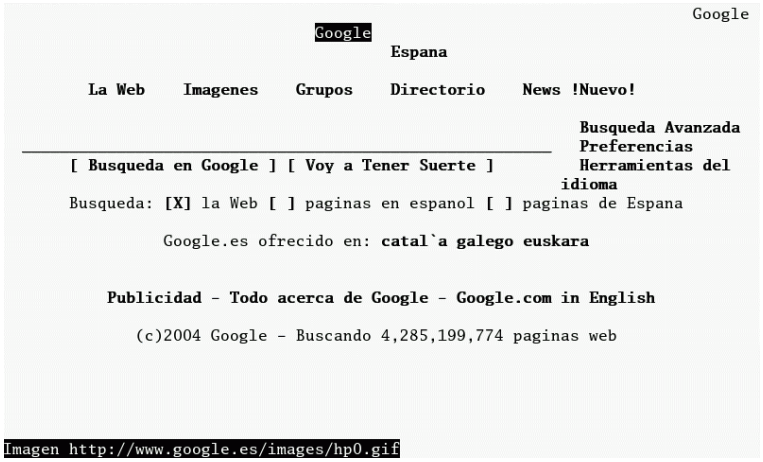
[2] No pensaréis que solo existen navegadores en modo texto. ¿Qué os parecería poder bajar el correo también de esta forma? Para ese cometido podemos usar el cliente de correo en texto plano `mutt` (un cliente de correo en texto plano es aquel que solo permite componer mensajes de correo electrónico en texto en formato ASCII). No se instala por defecto, así que antes de poder probarlo:

```
#apt-get update; apt-get install mutt
```

Un mini manual sobre su uso lo podéis encontrar en `http://linux-cd.com.ar/manuales/rh9.0/rhl-gsg-es-9/s1-eclients-textmail.html`.

[3] También si pulsamos con el ratón sobre la zona en la que aparece el menú.

y guardar la configuración (**Save options**). Para conectar con una web usaremos el menú **Fichero** y en URL escribiremos la URL de la página. Otra vez Google



Con las teclas de cursor o con el menú superior podremos navegar por nuestras web favoritas. Para salir, de nuevo, podemos usar la letra `q`.

4.1.2. Como servidor: Apache

Apache es el servidor Web (protocolo HTTP) más utilizado en el mundo actualmente. Se encuentra muy por encima de sus competidores, ya sean gratuitos o comerciales[4]. Por supuesto, es el más utilizado en sistemas Linux.

En su forma más simple, un servidor web transmite páginas en formato HTML a los navegadores cliente (Netscape, Explorer, Opera...). Pero el servidor web hoy día puede hacer mucho más, ya sea por sus propios medios o mediante su integración con otros programas.

Para instalarlo[5]:

```
# apt-get install apache2-mpm-prefork apache2-doc apache2-dev
```

Una vez instalado el servidor disponemos de un script que nos permite controlar sus estado, se trata de

[4]En Marzo de 2005 casi el 70 % de los servidores Web usan Apache, para saber exactamente los datos en la actualidad se puede consultar
`http://news.netcraft.com/archives/web_server_survey.html`

[5]El único fundamental es el primer paquete

`apache2ctl`

Admite los argumentos en línea de comandos:

`start`	para arrancar el servidor. Si ya está en marcha, nos avisará de ello.
`graceful`	con este parámetro le indicamos al servidor que relea los ficheros de configuración sin cerrar las conexiones activas. Las conexiones nuevas se iniciarán con la nueva configuración.
`restart`	reinicia el servidor (en su caso con la nueva configuración), pero a diferencia del anterior cierra las conexiones activas.
`stop`	cierra el servidor y, por tanto, las conexiones activas.

➥ **Para practicar** Aunque los anteriores son los más usuales, también podemos usar: `fullstatus`, `status` y `configtest`. Comprobar qué funcionalidad tienen. ■

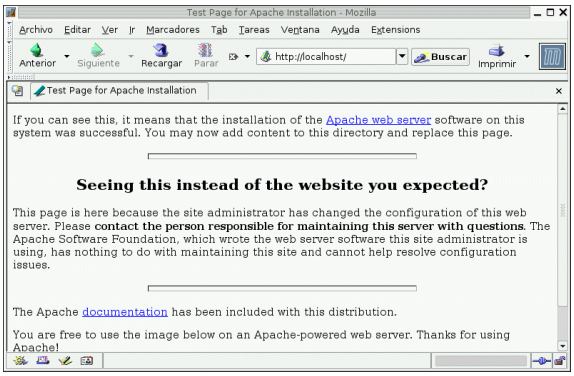
Pongamos en marcha nuestro servidor Web con

```
#apache2ctl start
```

y, para comprobar que funciona, podemos apuntar nuestro navegador preferido a la dirección

```
http://127.0.0.1
```

Si conseguimos una pantalla de bienvenida del servidor apache (*It worked!*), ya hemos contactado con nuestro servidor.



Si hemos instalado el paquete del manual podremos acceder a la extensa documentación[6](en castellano) que acompaña al programa desde esta misma página:

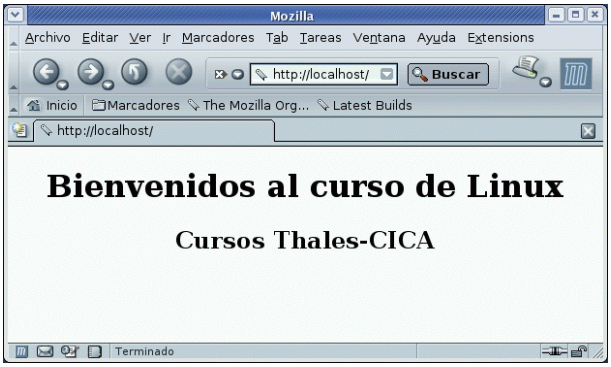
[6]A nuestra disposición también en `http://httpd.apache.org/docs-2.0/es/`

El siguiente paso es poner nuestras propias páginas en el servidor, en vez de las de bienvenida de apache. Sin más que ponerlas en el directorio `/var/www/apache2-default`[7] y empezando con la página `index.html`, podremos ver nuestras propias páginas.

➥ **Para practicar:** Comprobar que si ponemos el fichero `index.html`

```
$cat index.html
<center>
  <h1>Bienvenidos al curso de Linux</h1>
  <h2>Cursos Thales-CICA</h2>
</center>
```

en la ruta adecuada se obtiene:



4.2. Telnet y ssh

4.2.1. Acceso remoto: telnet

Dentro de las labores de un administrador de sistemas está el acceso remoto a los mismos, ya sea para buscar información en algún fichero del sistema, para copiar información o ejecutando en remoto algún comando.

Usando `telnet` podemos acceder a una máquina remota de la misma forma que lo haríamos si estuviéramos sentados delante de la consola y utilizásemos su teclado para introducir los comandos.

El término telnet proviene de *TELecommunication NETwork*. El punto débil de este tipo de conexiones es que todos los datos se transmitirán en claro en la red. Si un usuario captura los datos que viajan en la red con programas como `tcpdump` o `ettercap` podemos poner en compromiso la seguridad de nuestro sistema.

Los comandos que se teclean por parte del usuario son transmitidos directamente a la máquina remota y la respuesta de ésta es mostrada en la pantalla del usuario. De esta forma el sistema local es transparente al usuario, el cual tiene la sensación de estar conectado directamente a la máquina remota[8].

Para que podamos iniciar una sesión `telnet` se tienen que dar un par de condiciones:

[7] Para poder situarlas en `/var/www` tenemos que comentar la línea

```
#RedirectMatch ^/$ /apache2-default/
```

en `/etc/apache2/sites-available/default`, después ejecutaremos

```
#apache2ctl restart
```

para reiniciar el servicio.

[8] De esta forma podremos utilizar los recursos de ese ordenador (por ejemplo, ejecutando determinadas aplicaciones matemáticas para las que nuestro ordenador no tiene potencia suficiente).

- Que tengamos una cuenta de usuario en la máquina con la que queremos conectar.
- Que el servidor tenga un servicio de `telnet` activo.

Para acceder al sistema remoto se nos solicitará la identificación para poder entrar al sistema. Por ejemplo[9] para acceder a la máquina (inexistente) `tux.midominio.org` escribiremos

```
$telnet tux.midominio.org
```

a continuación se nos pedirá el nombre de usuario y la contraseña (de igual forma que si entramos en Guadalinex en modo texto).

➥ Para practicar

1. Para disponer de un servidor telnet sólo hemos de instalarlo con

   ```
   # apt-get install telnetd
   Leyendo lista de paquetes... Hecho
   Creando árbol de dependencias... Hecho
   Se instalarán los siguientes paquetes NUEVOS:
     telnetd
   0 actualizados, 1 se instalarï¿¿n, 0 para eliminar y 123 no actualiza-
   dos.
   Necesito descargar 40,7kB de archivos.
   Se utilizarán 139kB de espacio de disco adicional después de desempaque-
   tar.
   ```

   Y reiniciar el superdemonio inetd

   ```
   # /etc/init.d/inetd restart
   Restarting internet superserver: inetd.
   ```

   Podemos probar que funciona con[10]

   ```
   $telnet 127.0.0.1
   ```

   e introducir un nombre de usuario y conraseña válidos en ese sistema.

2. Si podemos acceder a algún servidor vía `telnet`, es interesante probar la posibilidad que nos ofrece Linux de trabajar en modo gráfico con programas situados en otro equipo, para esto tendremos que:

   Desde un Xterm de la máquina local ejecutaremos[11]

   ```
   $ xhost +máquina_remota
   ```

   después haremos un telnet a la máquina remota y una vez conectados escribiremos

   ```
   $ export DISPLAY=máquina_local:0
   ```

   por último ya sólo tenemos que ejecutar el comando que deseemos, por ejemplo, podéis probar con

   ```
   $ mozilla &
   ```

3. Como el servicio telnet es peligroso, mejor que si lo eliminamos de nuestro sistema con

   ```
   #apt-get install telnetd-
   ```

■

[9] Previamente deberemos haber establecido la conexión con nuestro proveedor de Internet.

[10] Suponemos que no tenemos tarjeta de red, si sí que tenemos cambiad la dirección del bucle local por la de vuestar tarjeta

[11] Donde `máquina_remota` es o bien la dirección IP de la máquina remota, o bien el nombre de esa máquina

4.2.2. SSH

ssh Protocolo *Secure Shell*, se usa para conexiones de red cifradas y autentificadas de forma más segura.

Desafortunadamente las conexiones vía `telnet` tienen un problema grave de seguridad ya que los datos se envían sin cifrar. Así, cualquier intruso puede interceptar nuestros datos y obtener nuestro nombre de usuario y password del sistema además del contenido de la comunicación.

La solución que se adopta es utilizar un sistema alternativo denominado SSH. `ssh` cifra los datos antes de pasarlos a la red, descifrándolos cuando llegan a su destino. El procedimiento de cifrado asegura que el intruso que capture los datos será incapaz de descifrarlos y verlos.

Para iniciar la conexión (seguimos con nuestra máquina ficticia de ejemplo) escribiremos[12]:

```
$ ssh -l thales tux.midomio.org
```

o equivalentemente

```
$ ssh thales@tux.midomio.org
```

La primera vez que conectemos aparecerá

```
The authenticity of host 'tux.midominio.org (xx.xx.xx.xx)' can't be es-
tablished.
RSA key finger-
print is 49:8c:9c:10:a9:c5:5d:e2:cd:88:65:f0:dc:02:f4:cf.
Are you sure you want to continue connecting (yes/no)?
```

Escribimos `yes` e **Intro**. Cuando siga, y aparezca

```
Warning: Permanently ad-
ded 'tux.midominio.org' (RSA) to the list of known hosts.
thales@tux.midominio.org's password:
```

será el momento de introducir la contraseña.

No pensemos que es algo para "hackers" y que no nos puede afectar. En los primeros centros TIC [13], los ordenadores de los centros TIC permitían conexiones `ssh`. De esa forma, como los alumnos conocían las IPs de las máquinas iniciaban sesiones con otros ordenadores del instituto (por ejemplo, el de la mesa del profesor) y podían borrar datos o "trastear" sobre ellos.

Por ejemplo, supongamos que la IP de la mesa del profesor es 192.168.3.24, si un alumno/a escribe:

```
$ssh usuario@192.168.3.24
```

como todos los ordenadores tienen un sólo usuario de nombre y contraseña `usuario`, cuando se le solicite escribirá los datos que le permiten autenticarse e iniciará una conexión con la máquina del "profe". Si ahora escribe:

```
$turnoff
```

el "pobre profe" verá que se le apaga la máquina como por arte de magia. Pero puede ser aún peor y que le borren cualquier dato que piensa está a buen recaudo en su ordenador.

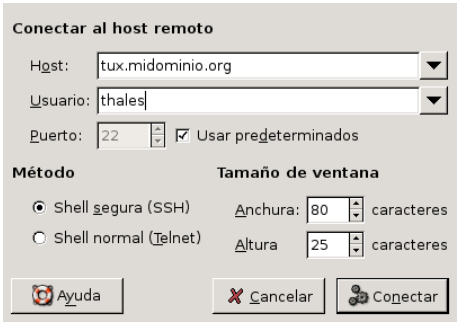
[12] Cada servidor SSH tiene un identificador único y secreto, llamado *host key*, para identificarse frente a los clientes que se conectan. La primera vez que nos conectamos a un servidor, la parte pública de la *host key* se copia en nuestra cuenta local (asumiendo que respondemos *yes*). Cada vez que nos conectemos a este servidor, el cliente SSH comprobará la identidad del servidor remoto con esta clave pública. Dicha clave pública, así como la del resto de máquinas con las que nos vayamos conectando se encuentra guardada en `$HOME/.ssh/known_hosts`

[13] Ya han solucionado este tema.

Conectar en modo comando y (gráfico)

Como siempre la forma sencilla para el final. Podemos conectar vía `telnet` o `ssh` usando el programa `gnome-remote-shell`, para acceder a él podemos usar la cadena de menús de Gnome **Aplicaciones→Menú Debian→Aplicaciones→red→Remote Shell** o bien desde una xterm escribir

```
$gnome-remote-shell
```

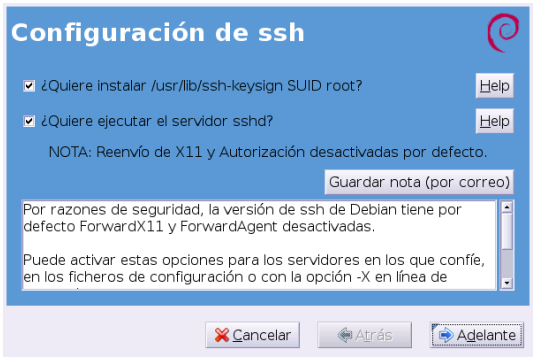


Su uso no presenta problema, sólo hemos de escribir el nombre o IP de la máquina con la que vamos a iniciar la conexión, el tipo de protocolo a usar (telnet o ssh) y listo, se inicia la conexión.

Servidor ssh

El paquete se instala por defecto, pero por defecto sólo permite que seamos cliente ssh, es decir, que podamos conectarnos nosotros a otra máquina en la que se ejecuta el programa servidor. Para que nuestro Guadalinex sea servidor del servicio SSH hemos de ejecutar

```
#dpkg-reconfigure ssh
```



y marcar las opción deshabilitada de: **Ejecutar el servidor sshd**.

➥ Para practicar

1. Comprobemos que funciona. Para eso

   ```
   $ssh root@127.0.0.1
   ```

   y comprobemos que nos conectamos con nuestra propia máquina. Podemos comprobar que hay una conexión ejecutando, por ejemplo:

   ```
   $w
   ```

   Para salir de la conexión escribimos `exit`
2. Si deseamos permitir que en las conexiones ssh se puedan ejecutar aplicaciones gráficas hemos de cambair la configuración del demonio de forma adecuada, para eso cambiemos la línea

```
X11Forwarding no
```

a

```
X11Forwarding yes
```

del fichero de configuración del servidor: `/etc/ssh/sshd_config`. Después reiniciemos el servicio con

```
#/etc/init.d/ssh restart
```

Ahora para conectar escribiremos

```
$ssh -X thales@127.0.0.1
```

y comprobemos que funciona ejecutando, por ejemplo:

```
$gedit
```

■

4.3. FTP y SFTP

4.3.1. ftp

Mediante una conexión ftp (*File Transfer Protocol* o Protocolo de Transferencia de Ficheros) podemos cargar y descargar archivos de la red. Este servicio puede verse dividido en dos partes:

- Los usuarios con cuenta en el sistema pueden acceder a su propio sistema de archivos y cargar y descargar información.
- Utilización anónima. En este caso pueden copiarse los ficheros de un servidor, a través de FTP, sin necesidad de usar una contraseña. En general, si nuestra conexión es anónima se nos informará al entrar en el sistema de que se nos aplican ciertas restricciones y que sólo podremos ver aquellas zonas del sistema de ficheros que permiten este tipo de acceso.

Para iniciar en modo comando una sesión ftp de este tipo escribiremos:

```
$ ftp tux.midominio.org
```

➥ **Para practicar** Intentemos una conexión con el servicio de ftp anónimo de rediris

```
$ ftp ftp.rediris.es
   Connected to ftp.rediris.es (130.206.1.5).
   220-=(<*>)=-.:. (( Welcome to ftp.rediris.es )) .:.-=(<*>)=-
   220-You are user number 199 of 1500 allowed
   220-< <
   220-Bienvenido al FTP anónimo de RedIRIS.
   220-Welcome to the RedIRIS anonymous FTP server.
   220-
   220-Este servidor FTP se ejecuta en una Ultra Enterprise 450 con 4
   220-procesadores conectados a varios dispositivos de almacenamiento
   220-que totalizan una capacidad superior a 1.8 Terabytes.
   220-Parte del hardware fué donado amablemente por Sun Microsystems.
   220-This server runs on a 4-processor Sun Ultra Enterprise 450
   220-connected different storage devices totalizing 1.8+ Terabytes.
   220-Part of the hardware was kindly donated by Sun Microsystems.
   220-
   220-Este FTP es de acceso público y funciona gracias a la infraestructura
   220-(máquinas y técnicos) de Red y Sistemas de Información de RedIRIS;
   220-es accesible desde todo el mundo gracias a todas las personas
   220-involucradas en el desarrollo de la Internet.
   220-
   220-Localice ficheros en: http://sunsite.rediris.es/busquedas/?lang=es
   220-
```

```
220-Find files at: http://sunsite.rediris.es/busquedas/index.en.php?lang=en
220->>
220-Local time is now 20:54 and the load is 2.45. Server port: 21.
220-Only anonymous FTP is allowed here
220 You will be disconnected after 5 minutes of inactivity.
Name (ftp.rediris.es:paco):
```

Como nuestra conexión es anónima escribiremos que somos el usuario `anonymous`

```
Name (ftp.rediris.es:paco): anonymous
```

para después introducir como contraseña una dirección de correo "válida"

```
Password:
```

y se iniciará la conexión.

Si la conexión no es anónima tendríamos que introducir el nombre del usuario que inicia la conexión así como su palabra de paso.

Para saber qué podemos hacer podemos ejecutar

```
ftp> help
Commands may be abbreviated.  Commands are:

!               debug           mdir            sendport        site
$               dir             mget            put             size
account         disconnect      mkdir           pwd             status
append          exit            mls             quit            struct
ascii           form            mode            quote           system
bell            get             modtime         recv            sunique
binary          glob            mput            reget           tenex
bye             hash            newer           rstatus         tick
case            help            nmap            rhelp           trace
cd              idle            nlist           rename          type
cdup            image           ntrans          reset           user
chmod           lcd             open            restart         umask
close           ls              prompt          rmdir           verbose
cr              macdef          passive         runique         ?
delete          mdelete         proxy           send
```

y si deseamos ayuda sobre un comando en concreto

```
ftp> help get
get             receive file
```

Para terminar podemos usar

```
ftp> exit
221 Goodbye. You uploaded 0 and downloaded 0 kbytes.
```

■

Para saber más sobre los comandos del ftp se puede consultar

El capítulo 9 de la *FAQ sobre Linux para principiantes* `http://es.tldp.org/FAQ/FAQ_Linux/Html/FAQ_Linux-9.html`

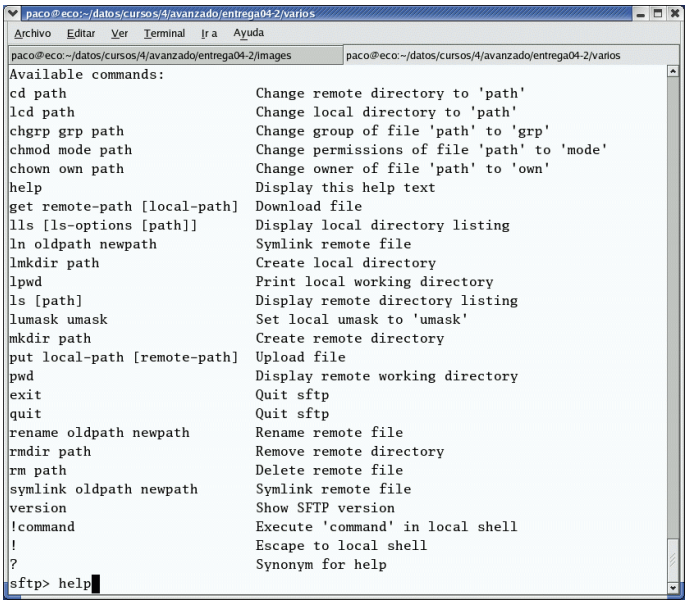
Estos *Apuntes de ftp*: `http://www.ignside.net/man/ftp/index.php`

4.3.2. sftp

¿Y qué es el `sftp`? Se trata de una especie de `ftp` pero seguro. Es decir, con `sftp` podemos conetarnos con un servidor de forma similar al clásico `ftp`, pero en este caso, tanto la autenticación como las transacciones de datos se cifran y aunque algún hacker malvado esté a la "escucha" no podrá obtener nada de nosotros.

Por ejemplo, para que `usuario` inicie una conexión sftp en modo comando con el servidor `tux.midomio.org` escribiremos

```
$ sftp thales@tux.midominio.org
```



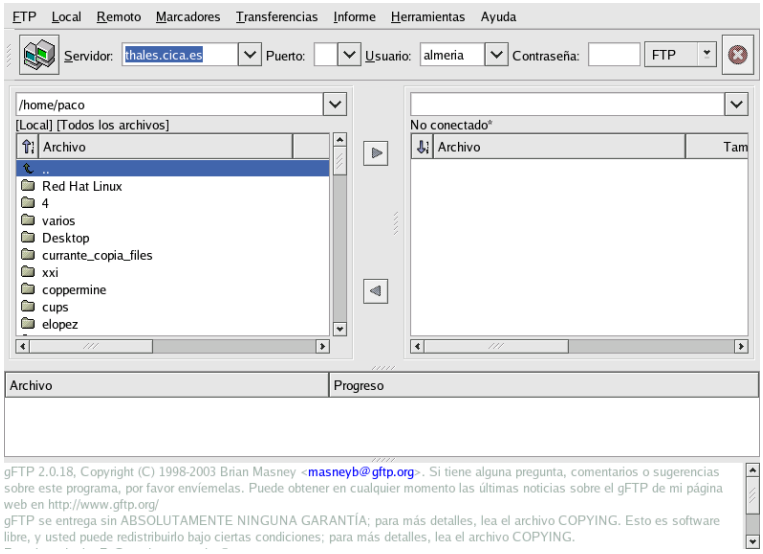
Los comandos de los que disponemos son similares a los del ftp. Pero, ¿por qué ir tan rápido?¿por qué no vemos ninguno?. Porque para realizar ambos tipos de conexión tenemos una utilidad gráfica[14] que nos puede sacar del atolladero de tener que estudiarlos, se trata de:

4.3.3. gFTP

Guadalinex incorpora el cliente gFTP[15] que nos permite conexiones en modo ftp y sftp. Se trata de un programa de transferencia de ficheros en modo gráfico que está a nuestra disposición en casi todas las distribuciones de Linux. Podemos acceder a él desde el escritorio con **Aplicaciones→Internet→gFTP** o bien desde un terminal con el comando

```
$ gftp &
```

[14] Recordar que con el programa `mc` también podemos realizar fácilmente conexiones ftp.
[15] Ya hablamos de él en la segunda entrega



Se ejecuta en una ventana, donde podemos distinguir:

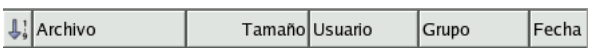
- En la parte superior de la ventana está la barra de menús que permiten acceder a todas las posibilidades del programa.
- Barra de herramientas. En ella indicaremos la dirección del servidor remoto con el que queremos establecer la conexión, el puerto de conexión (el 21 para una conexión FTP, lo pone por defecto), el nombre de usuario y la contraseña que nos identifique[16]. Resaltar dos botones:

   Para iniciar una conexión una vez introducidos los datos.

   Para desconectar

  Si, estando conectado, pulsamos de nuevo sobre  se cierra la conexión (podemos hacerlo también desde **Servidor→Desconectar**) . El botón  interrumpe el establecimiento de conexión.
- Barra de dirección.
- La parte central de la ventana está dividida en dos zonas, el lado izquierdo para el árbol del directorio local y el derecho para el del servidor al que conectemos. El campo superior de este recuadro muestra el directorio activo. Para transferir un fichero basta seleccionarlo y pulsar sobre la flecha de dirección.

  Desde ellas:

  - Pulsando sobre 

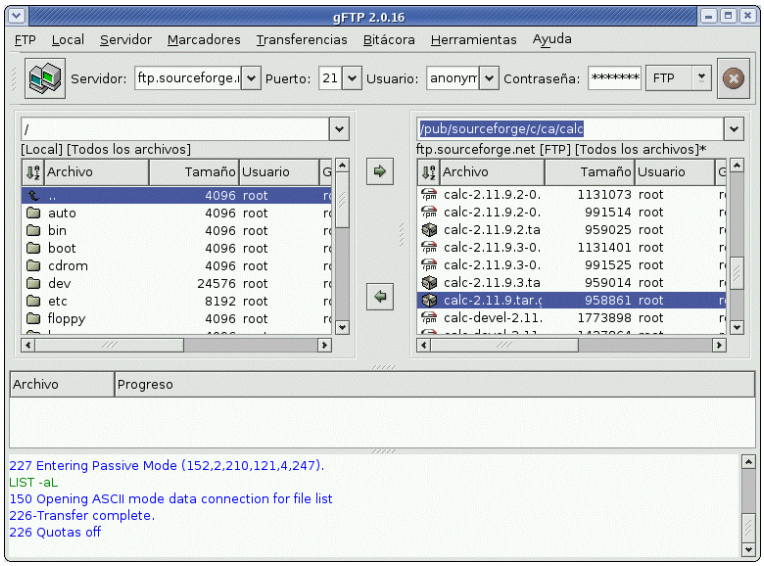
podemos ordenar los ficheros por nombre, fecha de modificación tamaño, etc
  - Si pulsamos sobre una de las ventanas con el botón derecho del ratón obtenemos el mismo resultado que desde **Local** o **Servidor** del menú principal

[16] Muchos servidores permiten la descarga de ficheros a personas anónimas (que no tienen cuenta en la máquina), en este caso debemos poner como usuario **anonymous** y como contraseña nuestra dirección e-mail.

- Ventana informativa que indica el progreso de las transferencias.
- Ventana de mensajes de la aplicación.

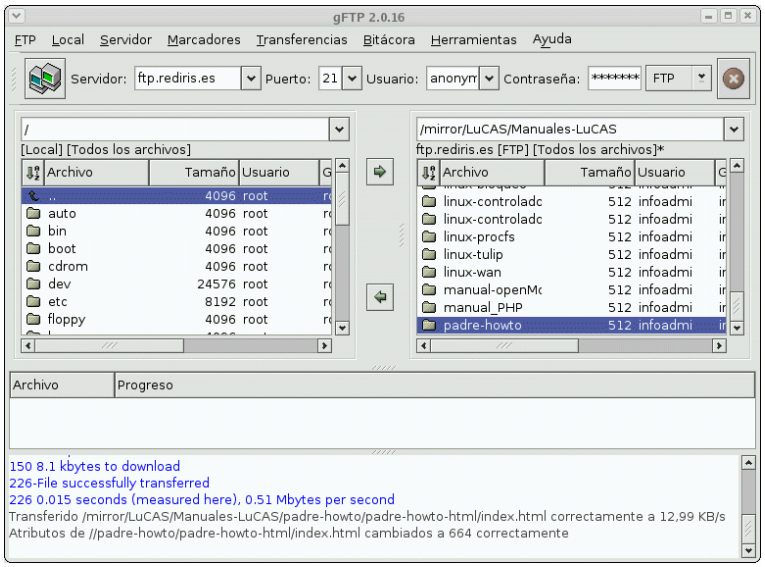
➥ **Para Practicar:** Rediris, Debian

- Pulsando sobre **Marcadores→General Sites →Source Forge** del menú principal conseguir llegar hasta la ruta `/pub/sourceforge/c/calc/` y bajarse las fuentes (`.tar.gz`) de la última versión del programa.



Una vez terminada la sesión desconectamos del servidor.

- Conectar con RedIris y bajar el directorio:`/mirror/LuCAS/Manuales-LuCAS/padre-howto` Escribiremos en el campo servidor: `ftp.rediris.es` y, tras movernos por el sistema de ficheros hasta la ruta especificada, bajaremos el directorio:



Conexiones sftp

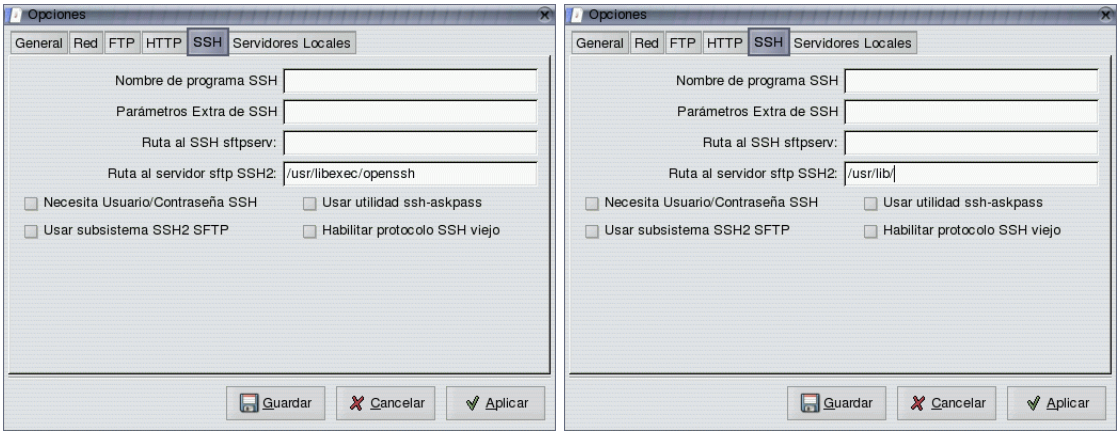
Antes de poder trabajar con él en modo **sftp** hemos de generar un par de ficheros que nos garantizan que la conexión es segura (véase la nota pie de página 62), para eso hemos de trabajar en modo terminal e iniciar una sesión `ssh` con el servidor con el que deseamos conectar para que se generen . Pero OJO, sólo es necesario hacerlo la primera vez que conectemos desde ese ordenador o si al ejecutar el programa comprobamos que no conecta.

Además, para que podamos trabajar con SFTP hemos de cambiar la configuración del programa, para eso pulsamos en el menú principal sobre **FTP** y en la ventana que aparece sobre **Opciones**

pestaña **SSH** y en el último campo escribimos:

`/usr/libexec/openssh` si nuestro servidor remoto es un Linux ejecutando la distribución Fedora o Red Hat

`/usr/lib` si la máquina servidora de ficheros es otra Guadalinex



Guardamos y **Aplicamos** y ya podemos conectar vía sftp.

Ejemplo Para realizar una conexión solo tenemos que rellenar los campos indicados: `servidor`, `usuario` y `contraseña`.

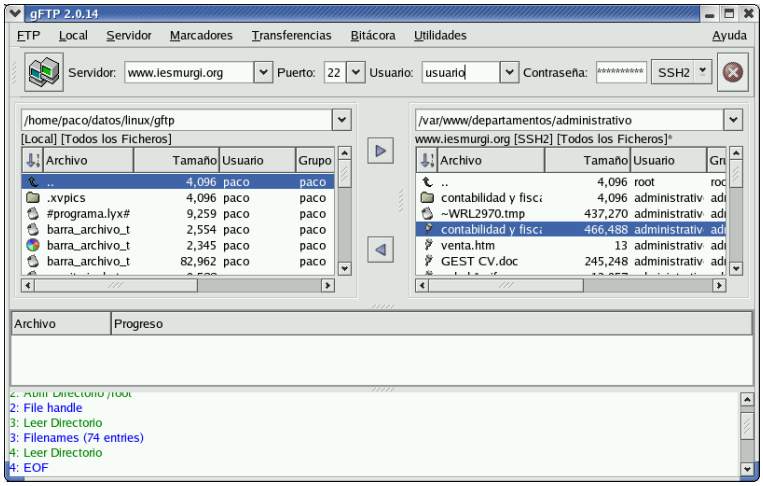
En nuestro caso serán

servidor: `tux.midominio.org`

usuario: `thales`

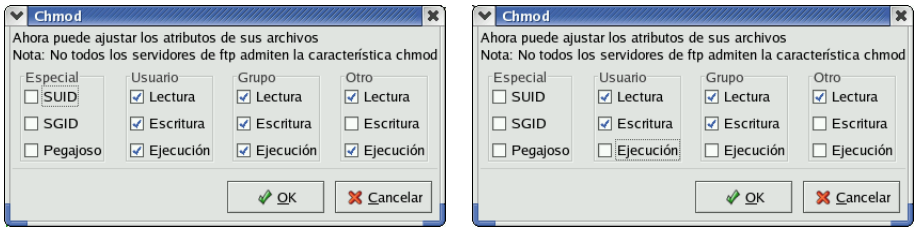
contraseña: `xxxxxxxx`

y optar en el menú desplegable que hay junto a **Contraseña** por conexión **SSH2**



Permisos Si iniciamos una conexión autentificada vía sftp o ftp, podemos modificar los permisos de un fichero o directorio de la máquina remota. Para eso sólo tenemos que marcarlo en vídeo inverso (situarnos sobre él) y tras pulsar sobre el botón derecho del ratón

optar por **Modificar Atributos.**



4.4. Cortafuegos

Un cortafuegos (*firewall*) de red es un dispositivo que divide la red en varias zonas con el objetivo de que el tráfico no autorizado entre o salga de la zona protegida[17].

Con la instalación de un cortafuegos vamos a conseguir que nuestro sistema sea mucho menos vulnerable a los intrusos. Es un sistema de seguridad que protege y aisla nuestra red.

A la máquina que tiene instalado el cortafuegos se le denomina *Bastion Host* y actúa controlando el "tráfico", los datos que pasan a través de él se analizan y se filtran siguiendo unas reglas que previamente hemos definido al configurar el cortafuegos.

En Guadalinex 2004 disponemos de un firewall que viene con el sistema. Se llama Firestarter. En realidad es un interfaz gráfico para iptables[18], pero que nos será muy útil para configurar nuestro cortafuegos, ya sea como cortafuegos personal o como cortafuegos para proteger una red.

Podemos lanzar el cortafuegos desde **Aplicaciones→Configuración→Sistema→Firestarter** o ejecutando desde un terminal [19]

```
#firestarter[20]
```

La primera vez que ejecutemos el programa nos aparecerá un asistente que nos va a guiar en su configuración. Pulsamos sobre [**Adelante**] y se abre una nueva ventana en la que podemos seleccionar el interfaz de nuestro sistema que se conecta a Internet. En este caso es `eth0` para conectarnos a un router ADSL. Con esta opción estamos seleccionando el firewall personal para proteger nuestra máquina de ataques externos.

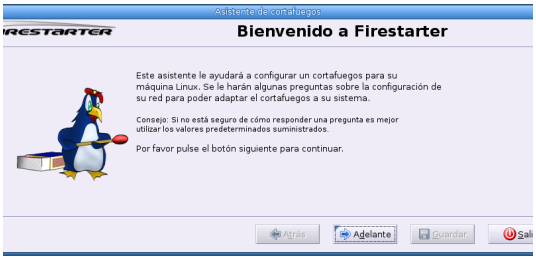
Además, marcamos la casilla **Iniciar el cortafuegos al conectarse al servidor**, para que se inicie de forma automática cuando nos conectamos a internet.

[17] Para ampliar `http://www.rediris.es/cert/doc/unixsec/node23.html`

[18] La función de *iptables* es la de establecer, mantener e inspeccionar las reglas de filtrado de paquetes IP en el núcleo de Linux. Iptables decide qué paquete de información puede pasar, según unos criterios que se almacenan en unas listas.

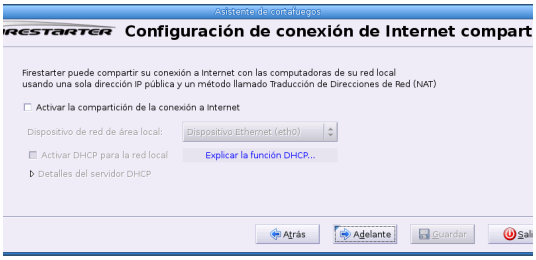
[19] Nos pedirá la contraseña del root

[20] La versión que viene con Guadalinex no es la última. Si queremos tener la última versión tendríamos que actualizar desde los repositorios de Debian
`#apt-get install firestarter`,
pero tiene bastantes problemas de dependencias y no es fácil si no tenemos cierta experiencia.



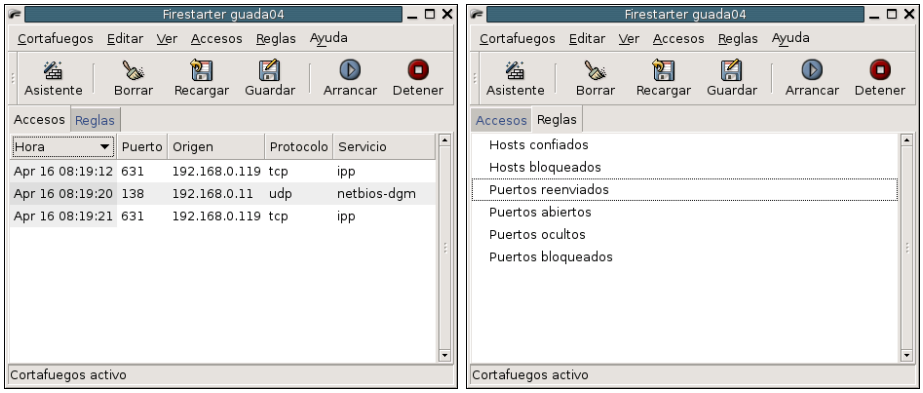
La siguiente pantalla nos permite decidir si el firewall de nuestro sistema funcionará también como pasarela para otros sistemas, convirtiéndose en el guardián de nuestra red. Para ello, debemos contar con otra interfaz de red distinta de la anterior, que se conectará a la red protegida.

En este caso, como solamente queremos un firewall personal, no activamos la opción **Activar la compartición de la conexión a Internet**. Si quisiéramos que éste sea el cortafuegos de nuestra red, debemos especificar la interfaz conectada a la red de área local interna y si queremos activar el servidor de DHCP para dar direcciones automáticamente.



Pulsamos sobre [**Guardar**] y ya tenemos una configuración básica del cotafuegos para uso personal.

Nos aparece ahora una ventana en la que podemos ver, entre otras cosas, las conexiones activas en nuestra máquina (pestaña **Accesos**). Desde ella podemos también modificar las opciones o volver a ejecutar el asistente seleccionando **Cortafuegos→Ejecutar asistente**. La pestaña **Reglas** nos permite ver las conexiones que han sido bloqueadas por el cortafuegos, añadir nuevas reglas y eliminar o modificar reglas, tanto de tráfico entrante como saliente.



4.5. Samba

Usando Samba[21] podemos compartir y utilizar recursos de sistemas de ficheros Linux e impresoras con sistemas Windows 3.11, 9x, NT, 2000, XP. Samba es rápido y sencillo de configurar. Linux[22] con Samba puede trabajar como servidor y como cliente. Como servidor ofrece recursos

[21] La página principal de Samba es: `http://www.samba.org`

[22] Por defecto, Guadalinex instala los paquetes `samba` y `samba-common`.

(discos e impresoras) para que los utilicen las máquinas windows. Como cliente utiliza los servicios ofrecidos por las máquinas windows (discos e impresoras).

Samba debe sus "orígenes" a Andrew Tridgell. Necesitaba poder compartir archivos desde el DOS a su servidor UNIX y consiguió el primer programa sobre 1992, que si bien funcionó dejaba bastante que desear. En 1994, tras retomar el proyecto inicial pero ahora con la idea de interconectar en la misma red Windows y Linux, apareció de forma "oficial" Samba.

Instalación

Si bien se instala por defecto, lo mejor es actualizar los paquetes a la última versión[23]:

```
# apt-get install samba samba-common smbclient samba-doc smbfs
```

4.5.1. Configuración

Una vez instalados los paquetes estamos casi listos para funcionar, ya que los demonios que requiere se ponen en marcha por defecto al arrancar el sistema operativo[24]. Antes de activarlos tendremos que configurar la máquina Linux y la máquina Windows.

El proceso consiste:

Configuración de las máquinas Güindows

Para trabajar con Samba tendremos que tener cargados los protocolos TCP/IP, por tanto, en Red comprobaremos que tenemos instalados[25] esos protocolos

---

[23]

- Los dos últimos paquetes no son "indispensables", se trata de la documentación de SAMBA y de poder disponer de los comandos `mount` y `umount` para el sistema de ficheros `smb`.
- También podemos bajarlos de
  `http://us2.samba.org/samba/ftp/Binary_Packages/Debian/samba3/dists/stable/main/binary-i386/`
  y usar `dpkg`.
- Si deseamos disponer del demonio `winbindd` habrá que instalarlo:

  ```
  #apt-get install winbind
  ```

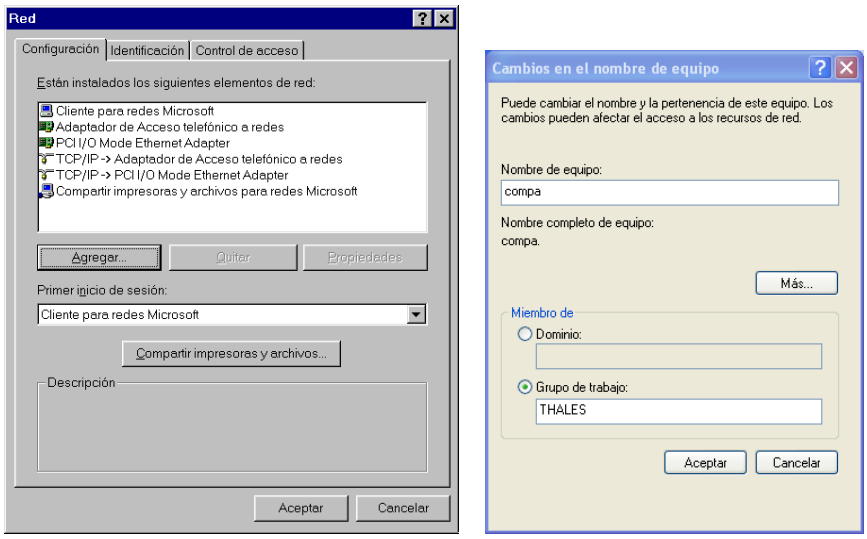
[24]Si deseamos activarlos sin reiniciar el sistema escribiremos:

```
#/etc/init.d/samba start
```

Si optamos por cambiar las opciones de arranque podemos usar la herramienta gráfica:

```
#services-admin
```

[25]Notar que las capturas son un poco viejas, pero hemos preferido mantenarlas porque en Windows 95 no se instalaba por defecto el protocolo TCP/IP. No es así en los Windows posteriores.



y asignaremos una dirección IP a nuestras máquinas (por ejemplo 172.26.0.11), en máscara de Red pondremos 255.255.255.0

Asignaremos un nombre a nuestra máquina (BAG) y pondremos el grupo de trabajo en el caso de que no estuviese ya definido (THALES). Repetiremos este proceso con la máquina COMPA.

Configuración de la máquina Linux

Archivo de configuración de Samba El archivo de configuración de Samba es `/etc/samba/smb.conf`[26].

Mediante este archivo podemos controlar la gran cantidad de opciones disponibles del programa, aunque tratar a fondo todas y cada una de ellas está más allá del objetivo de esta entrega.

Se divide en varias secciones, cada una de ellas determinada por un nombre entre corchetes: `[impresoras]`, `[global]`, etc. En cada sección encontramos una serie de parámetros compuestos por pares de clave/valor. Los comentarios comienzan con punto y coma (`;`) o mediante una almohadilla (`#`).[27] Ejemplifiquemos esto con una parte de la sección `[global]`:

```
[global]
# La almohadilla indica que estamos en un comentario
   remote announce = 172.26.0.255 172.26.2.44
# El siguiente parámetro está marcado como comentario y
# el segundo no, es decir, está activo
;   local master = no
   os level = 33
```

Si modificamos el fichero de configuración tendremos que reiniciar el servidor[28] con:

```
#/etc/init.d/samba restart
```

El fichero que se instala en un sistema actualizado de Guadalinex 2004 es de la forma

```
1 [global]
  netbios name = G2004-1108897514
```

[26] Antes de tocar este archivo deberíamos hacer una copia por si acaso.

[27] Normalmente apareceren marcados los parámetros como comentarios utilizando el punto y coma, y se deja la almohadilla para comentarios normales.

[28] Es deseable actualizar el fichero `/etc/hosts` con las direcciones IP y el nombre de cada máquina a la que vamos a acceder. De esta forma con sólo escribir el nombre del equipo al que queremos acceder, el servidor buscará en ese fichero el nombre del equipo y dirección IP correspondiente.

```
    server string = Guadalinex 2004
    workgroup = GUADALINEX
    wins support = no
    encrypt passwords = true

# Do something sensible when Samba crashes: mail the admin a backtrace
# panic action = /usr/share/samba/panic-action %d

[compartido]
    path = /home/compartido
    comment = Directorio compartido en Guadalinex 2004
    writeable = no
    guest ok = yes
    guest only = yes
    browseable = yes
```

Listado 4.1: /etc/samba/smb.conf

y deberíamos adecuarlo a nuestros intereses.

Para comenzar a trabajar y conocer las posibilidades que nos ofrece el programa sólo vamos a comentar las opciones más usuales de este fichero.

Sección [global]

En esta sección configuraremos parámetros para todo el servidor SAMBA así como algunos valores predeterminados a las otras secciones. Veamos algunas de las opciones más usuales[29].

Comencemos por ajustar el grupo de trabajo[30] en el que nos encontramos. Por ejemplo, si nuestro grupo de trabajo es `GUADALINEX`, escribiremos:

```
workgroup = GUADALINEX
```

Con el parámetro

```
server string = Guadalinex 2004
```

indicamos el nombre que identificará al servidor cuando lo consultan los clientes SAMBA. Con la directiva

```
netbios name = G2004_1108897514
```

establecemos el nombre NetBIOS de la máquina.

- Si no se porporciona el nombre del grupo de trabajo tomará como grupo predeterminado `WORKGROUP`.
- Si no establecemos el nombre NetBIOS de la máquina tomará el que se obtenga de ejecutar el comando `hostname`

Le dice al componente NMBD de Samba que deshabilite su servidor WINS

```
wins support = no
```

Activa la encriptación de las password, es el modo por defecto

```
encrypt passwords = true
```

[29] Para conocer todas las opciones véase la traducción del libro de Oreilly *Usando SAMBA* (imprescindible) disponible con la documentación que acompaña al programa y traducido en:
`http://es.tldp.org/Manuales-LuCAS/USANDO-SAMBA/`
`http://www.sobl.org/modules.php?name=Downloads`

[30] Deberá estar limitado como máximo a nueve caracteres, sin espacios y todos en mayúsculas.

Vamos a comentar además algunas que no aparecen:

Sección [homes]

Con esta sección podemos controlar de qué forma accederán los clientes al directorio principal de usuario en el servidor Linux. La configuración aquí establecida permite que los parámetros sean válidos para todos lo usuarios y no hay que especificar una configuración para cada usuario por separado.

```
comment = Directorios de usuario
browseable = no
writable = yes
valid users =%S
```

Analicemos cada una de las líneas anteriores:

comment cadena de identificación que se muestra a los clientes que examinan el servidor Samba.

browseable al establecerlo a `no` conseguimos que el explorador de Windows no nos muestre los `$HOME` de otros usuarios del sistema Linux. Si no somos ningún usuario registrado del sistema no veríamos ningún `$HOME`. Sin embargo, si nos conectamos como un usuario del sistema Linux aparecerá una carpeta de nombre ese usuario a la que podremos acceder con nuestra contraseña. Si lo establecemos a `yes` aparecerán todas las carpetas de los `$HOME` de usuario, aunque no podríamos acceder a ellas salvo que la validación sea la correcta (nombre de usuario y contraseña)

writable (sinónimo de **writeable**) permite que un usuario pueda crear y modificar archivos en su directorio `$HOME` cuando inicia una conexion SAMBA. Podemos conseguir esto mismo sustituyendo esta línea por

```
read only = no
write ok = yes
```

valid users lista de usuarios que pueden conectarse a un recurso (en este caso, la variable `%S` contiene el nombre actual del recurso).

Secciones personalizadas

Usando las secciones personalizadas podemos compartir impresoras o directorios de una manera no genérica. La idea que subyace en estas secciones consiste en poder compartir directorios para grupos de usuarios o permitir que determinados directorios sean de acceso público. A su vez, si tenemos varias impresoras conectadas a nuestra máquina y no queremos darlas todas a compartir, podemos usar esta sección para dar a compartir sólo una impresora en concreto. Por ejemplo, el fichero por defecto de Guadalinex trae una sección personalizada de la forma

```
[compartido]
path = /home/compartido
comment = Directorio compartido en Guadalinex 2004
writeable = no
guest ok = yes
guest only = yes
browseable = yes
```

Con esta configuración, disponemos de un directorio de uso compartido y visible por todos (`browseable = yes`) que no está exento de riesgos de seguridad ya que:

- no es necesario nombre de usuario ni contraseña para acceder al recurso (`guest ok = yes`), y
- aunque la conexión sea autentificada, en este recurso se accede como anónimo (`guest only = yes`)

Con el ejemplo que se lista a continuación permitimos acceso al servicio `trabajos` a los usuarios del grupo llamado `clase`:

```
[trabajos]
  comment = Directorio compartido de la clase de informática
  path = /home/trabajos
  browseable = yes
  writable = yes
  printable = no
  valid users = @clase
```

Las líneas de este ejemplo significan:

- damos a compartir el directorio `/home/trabajos`
- La identificación de este servicio es: `Directorio compartido de la clase de informática`
- Al estar `browseable` en `yes`, se mostrará la carpeta del recurso compartido en el explorador de Windows siempre que accedamos al servidor SMB.
- Permitimos que se pueda escribir en él.
- Con `printable = no` indicamos que no es un servicio de impresión.
- Con el parámetro `valid users = @clase` restringimos el acceso a este directorio a miembros del grupo `clase`.

Cuando terminemos de configurar nuestro sistema podemos usar:

```
$ testparm
```

y comprobar que todo está perfectamente.

Si instalamos varios Guadalinex y no actualizamos SAMBA tendremos que ajustar el nombre NetBIOS ya que por defecto pone en todos ellos el mismo. Si actualizamos SAMBA este problema desaparece.

4.5.2. Swat

Vamos a comentar una herramienta que permite configurar Samba usando el navegador Web, se trata de SWAT (*Samba Web Administration Tool*). SWAT se incluye como parte del paquete estándar de Samba. La idea de este programa (hay más con esta misma filosofía) consiste en facilitar la configuración del servidor.

```
# apt-get install swat
```

Una vez instalado, hemos de descomentar la línea adecuada (una que comienza por `swat`) del fichero `/etc/inetd.conf` y releer la nueva configuración

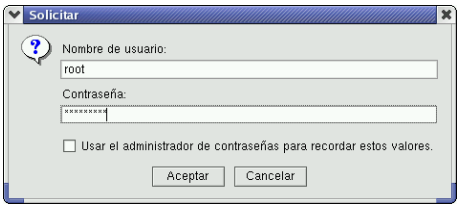
```
#/etc/init.d/inetd reload
```

Uso

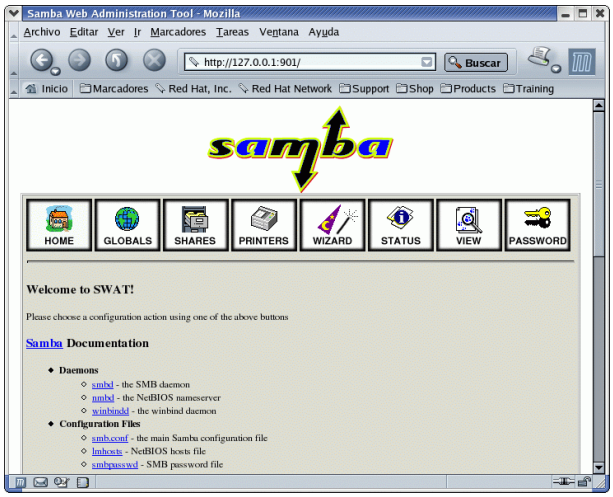
Para usar el programa SWAT tendremos que iniciar un navegador y nos conectaremos a la URL

`http://localhost:901/`

Tras la ventana que nos pide los datos del root



accederemos a la página principal de la aplicación:



Los iconos de la parte superior de la pantalla nos permiten acceder a diferentes páginas de SWAT:

página principal, en ella entre otras cosas tenemos enlaces a la documentación del paquete Samba. Entre otros el libro de Oreilly (pero en Inglés) comentado en esta misma sección.

desde aquí podemos manipular la sección globals del archivo `/etc/samba/smb.conf`. Podemos modificar los valores de los distintos parámetros, obtener ayuda sobre ellos y/o mantener el valor predeterminado. Para grabar los cambios pulsaremos sobre [**Commit Changes**].

usando esta página podremos añadir, modificar o borrar recursos compartidos. Por defecto la pantalla inicial muestra sólo los parámetros de uso más frecuente del archivo `smb.conf`. Si pulsamos sobre [**Advanced View**] tendremos la posibilidad de configurar alguno de los parámetros menos usados.

para configurar las impresoras.

desde esta página podemos limpiar el archivo `smb.conf` de todos los comentarios y valores por defecto.

desde aquí podemos comprobar el estado de Samba. Además de ver cómo están las "cosas" podemos arrancar y parar los demonios de Samba y ver las conexiones activas de nuestro servidor Samba.

para examinar el contenido del archivo `smb.conf`. Si deseamos ver todas las variables disponibles y sus valores hay que pulsar en **Full View**.

con este enlace aparecerá la pantalla mediante la cual podremos cambiar cuentas de usuarios locales y cuentas del controlador del dominio primario.

4.5.3. A "bailar" la Samba

Acceder desde una máquina Linux a una Windows

Como es evidente sólo podremos acceder a aquellos recursos autorizados en la máquina Windows.

Si optamos sólo por usar esta posibilidad no necesitamos tener instalado el paquete "principal" de la aplicación[31]. Trabajando de esta forma Samba no comparte ningún recurso con otro sistema, se limita a acceder a los recursos compartidos en los servidores de recursos la red. Si trabajamos sólo como cliente debemos ajustar a nuestro grupo de trabajo el archivo `smb.conf`, la única línea necesaria en ese fichero sería:

```
workgroup=THALES
```

Acceder en modo gráfico es fácil, sólo hemos de pulsar sobre el escritorio en **Equipo**→**Red**[32]

y acceder a la **Red de Windows**[33]

A continuación, y si el recurso al que queremos acceder de la red Windows está protegido con nombre de usuario y contraseña escribiremos los datos pedidos.

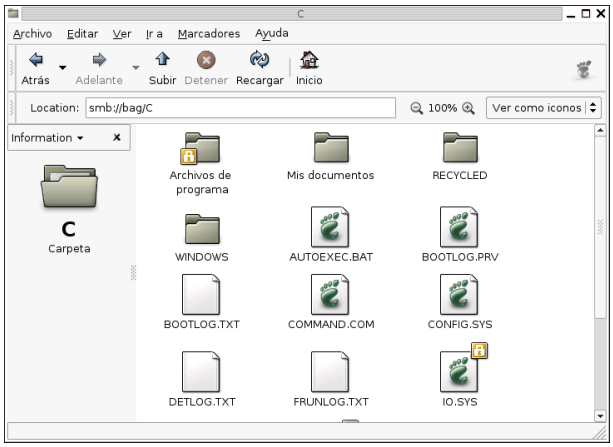
Como ejemplo de lo que podemos hacer vamos a prestar atención a la captura que sigue. Se trata de una máquina en la que se ejecuta Windows 98 con la unidad C: como recurso compartido y a la que hemos accedido desde Guadalinex.

[31] Sólo hay que instalar `samba-client` y `samba-common`.

[32] Equivale a abrir **Nautilus** y escribir `network://`

[33] Pero, además de la posibilidad de acceder en modo gráfico, también lo podemos hacer en modo texto. Para eso se puede usar una de las utilidades más interesantes de las que acompañan a Samba es `smbclient`. Con él podemos acceder desde Linux a los recursos compartidos en máquinas Windows con métodos que incluirían FTP, NFS y los "comandos r", como rcp.

`smbclient` nos permite disponer de un interfaz similar a un FTP, por tanto, es una utilidad cuyo objetivo es accesos temporales a un recurso compartido.



➥ **Para practicar:** comprobar que con el comando que sigue, podemos obtener información de los equipos y recursos compartidos:

```
$smbtree
```

■

smbmount y smbumount

Si deseamos tener la posibilidad de montar un sistema de archivos compartido en el árbol del sistema de archivos de Linux tenemos que usar `smbmount`

Para montar algún recurso de la máquina Windows usaremos el comando:

```
# smbmount //máquina_windows/recurso destino_montaje
```

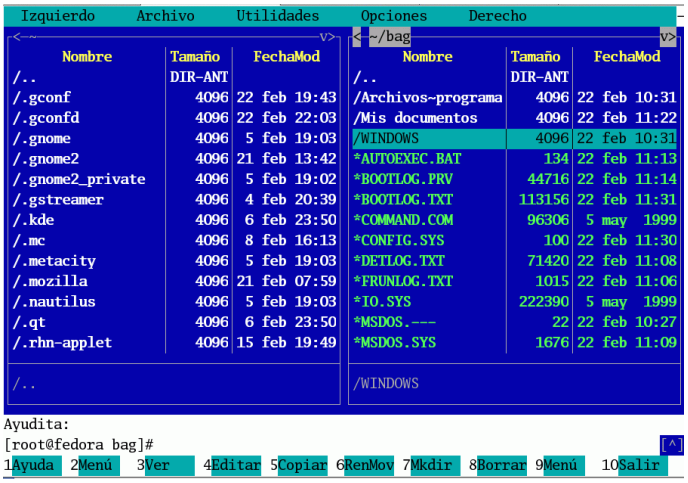
↬ Veamos su utilidad con un ejemplo. Supongamos que tenemos una máquina de nomnbre BAG en la que se ejecuta Windows 98 y que en ella disponemos del recurso `C`, montémoslo en nuestro sistema de archivos. Para eso creemos un directorio destino de montaje:

```
# mkdir /mnt/bag
```

ya sí, usemos ahora

```
# smbmount //bag/C /mnt/bag
Password:
```

Al usar este comando, estamos consiguiendo que el recurso compartido sea montado en `/mnt/bag` y que lo veamos como cualquier otra parte del sistema de archivos Linux.



Para desmontarlo usaremos el comando `smbumount`

```
#smbumount /mnt/bag
```

o sólo `umount`

```
#umount /mnt/bag
```

Acceder desde Windows a la máquina Linux

Un papel más interesante para Linux y Samba en una red Windows es el de servidor de recursos. Para conseguir esto debemos tener instalado el paquete `samba` y tener correctamente configurado el fichero `/etc/samba/smb.conf`.

Si deseamos que los usuarios de la máquina linux puedan acceder a sus $HOME de usuario, hemos de añadir al fichero de configuración de SAMBA una sección de la forma:

```
[homes]
   comment = Directorios de usuario
   browseable = no
   writable = yes
```

y reiniciar el servidor para que relea la nueva configuración:

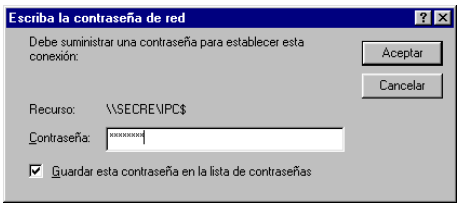
```
#/etc/init.d/samba restart
```

Para acceder (sin hacer más cambios que los ya comentados) desde una máquina Windows a una Linux sólo lo podremos hacer si somos usuarios registrados en el sistema Linux. Además, en Linux debemos dar de alta a ese usuario y definir con qué contraseña puede acceder a nuestro sistema. Si queremos dar de alta al usuario *cursolinux* de la máquina Linux (con igual nombre en la máquina Windows) usaremos:

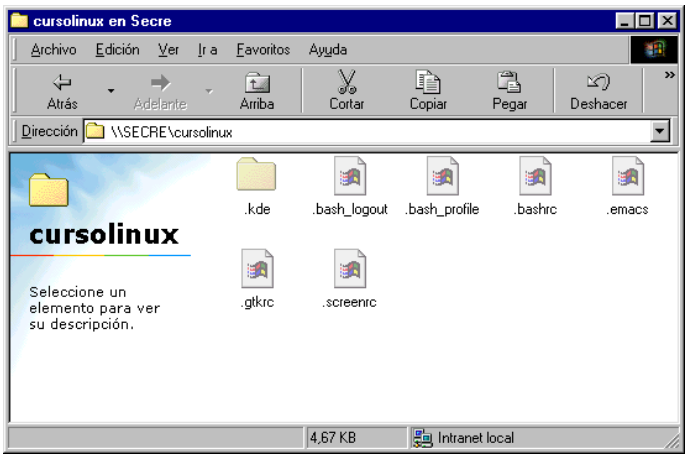
```
#smbpasswd -a cursolinux
ENTER password for cursolinux
New SMB password:
Retype new SMB password:
Added user cursolinux.
```

Podremos cambiar la contraseña después usando el comando `smbpasswd` (sin el parámetro `-a`). Si lo que deseamos es borrar un usuario le pasaremos como parámetro `-x` seguido del nombre de ese usuario.

En la máquina Windows 9x tendremos que entrar **como ese usuario** (si se trata de un XP esto no es necesario) y si en el inicio no ponemos la contraseña, ésta se nos pedirá cuando intentemos acceder a los servicios de Red:



Una vez validada la contraseña (interesa desmarcar la casilla de guardar contraseña ya que si no, nos arriesgamos a que nos fastidien el Linux desde Windows), los directorios a los que tengamos acceso en nuestro sistema Linux se nos mostrarán como carpetas de Windows y podremos trabajar con ellas de la forma habitual.



Es importante resaltar que si el grupo de trabajo no está bien configurado en el fichero `/etc/smb.conf` no veremos a la máquina Linux cuando accedamos a la red.