

Guía de iniciación a
Gnu/Linux
nivel de usuario medio

Copyright (c) 2006 Antonio Becerro Martinez.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

1. INTRODUCCION.

Este curso está dirigido a personas con conocimientos de informática, que por la razón que sea, nunca hayan trabajado con un sistema **Unix**, o similar a **Unix**, como **Gnu/Linux** o **Bsd**. El uso del interprete de comandos queda implícito en el desarrollo del curso. El interprete de comandos que vamos a utilizar es **Bash**, por su compatibilidad y amplia difusión. Se puede utilizar desde el escritorio gráfico, abriendo la **shell** en una ventana o sin escritorio, trabajando directamente en modo texto. En cualquier caso, la **shell** va a ser nuestra compañera inseparable durante todo el curso, ya que vamos a configurar todos los aspectos básicos de nuestro sistema operativo mediante la edición de los archivos de configuración de cada programa, con editores de texto simple como **Emacs** o **Xemacs**. La elección de este método es deliberada. Creo firmemente, que es la mejor forma de comprender y aprender un sistema **Unix**. Las bonitas aplicaciones gráficas, son sin duda muy útiles, pero generan dependencia. El usuario finalmente no llega a saber que es realmente lo que hace. No aprende a ejecutar los procesos que necesita, ya que la aplicación gráfica hace esta tarea por él. Y lo que es aún peor: no llega a captar la lógica interna del funcionamiento de su sistema operativo. Por el contrario, trabajando en la **shell** el sistema se hace transparente. Lo que vemos es realmente lo que hay. Es un poco árido, pero acabamos por controlar la situación. Finalmente, esta experiencia nos pone en el camino para poder utilizar cualquier sistema operativo sin depender de las empresas y de sus políticas.

2. COMANDOS BÁSICOS.

En primer lugar, creamos un fichero nuevo de texto llamado: **hola.txt**, mediante cualquiera de los editores de texto de **Gnu/Linux**, como **emacs** o **xemacs**. Lo editamos, añadiendo varias filas con palabras sencillas, como zapato, pollo, armario, falda o cualquiera que se nos ocurra. Guardamos el fichero en nuestro directorio (**/root**). Para visualizar el contenido del fichero desde la línea de comandos, utilizamos el comando: **cat**. Veamos como:

```
cat hola.txt
```

cat, recoge **hola.txt** y lo envía a la salida estandar (monitor). El comando: **sort**, hace lo mismo, pero muestra la salida ordenada alfabéticamente. Un concepto muy interesante, es el de tubería, pipe o pipeline. Es el signo: **|**, que se obtiene mediante la combinación: **AltGr+1**. Se utiliza para enlazar la salida de un comando con la entrada de otro, de tal manera que se puedan realizar varias acciones consecutivas. Veamos un ejemplo:

```
cat hola.txt | grep za
```

muestra adios.txt y filtra todas las palabras que contengan los caracteres: za

Veamos ahora otros ejemplos:

```
sort -r hola.txt | grep -v so
```

muestra hola.txt invertido alfabéticamente, filtra todo excepto: so.

```
cat adios.txt | grep -v so | mail root
```

muestra adios.txt, filtra todo excepto:so y envía la salida en un correo al root

El comando: **grep**, sirve para filtrar, es decir para hacer selecciones del contenido de un fichero según la cadena de caracteres que se le pase. Es decir, **grep co** significa: busca las palabras que incluyan, de alguna manera, la cadena de caracteres: **co**. Admite varios parámetros, como **-r** (recursivo), **-i** (no tener en cuenta mayúsculas y minúsculas), **-v** (selección inversa). Veamos otro ejemplo:

```
grep -r "hola.txt" /root
```

Filtra hola.txt recursivamente en el directorio del root

Realmente, lo que hacemos es buscar el fichero: **hola.txt**, en el directorio del root. El parámetro **-r** permite que **grep** busque dentro de todos los subdirectorios, la cadena de caracteres que queramos filtrar. Ahora vamos a ver los comandos de uso general. Se utilizan para tareas cotidianas, como moverse por el sistema, o crear o eliminar directorios.

```

touch      # crear fichero vacio.
rm         # borra fichero.
mkdir      # crea directorio.
rmdir     # Elimina directorio.
rm -r     # Elimina directorios llenos.
ls        # Lista directorio.
who       # Nombre de usuario.
cd        # ir a directorio.
cd /      # Ir a directorio raiz.
cp        # copiar.
mv        # renombrar y mover.
ps        # procesos activos.
ps aux    # Todos los procesos.
clear     # Limpiar pantalla.
cd ..     # Ir al directorio superior.
    
```

2. ACCESO REMOTO MEDIANTE SSH.

En el apartado anterior, hemos visto como realizar las tareas más comunes en una consola de texto de un sistema **Unix**. Quizás, nos haya llamado la atención el comando **who**. Este, devuelve el nombre del usuario. ¿Para que podría servir un comando así? ¿Es que no sabemos quienes somos? puede que no lo sepamos, si trabajamos en varias máquinas simultaneamente. Y esto es bastante normal en los sistemas **Unix**, ya que fueron diseñados para el trabajo en red. El protocolo **ssh**, permite este tipo de comunicaciones, de una forma sencilla. Para que funcione, es preciso instalar un programa servidor **ssh** en las máquinas a las que queramos acceder, y un cliente **ssh** en nuestro ordenador. La sintaxis es muy clara. Para conectar un equipo:

ssh dirección ip

Los clientes no son todos iguales. Los hay en modo texto y en modo gráfico. Algunos permiten ejecutar las **X**, es decir lanzar el escritorio del equipo conectado. Esta última posibilidad consume muchos recursos. Es necesario disponer de un ancho de banda considerable para disfrutar de un rendimiento aceptable. En modo texto, no existe esta contrariedad. **ssh** significa: **secure Shell**, es decir **Shell** con encriptación. A diferencia de otros protocolos, como **telnet**, **ssh** es seguro. Podemos tener abiertas varias sesiones en equipos remotos, dentro de la misma consola. El comando **who** sirve para ver en que máquina nos encontramos en cada momento. Para ir cerrando sesiones se utiliza el comando: **exit**.

Supongamos que alguien ha accedido a nuestro equipo, y deseamos expulsarlo. Para ello, solo hay que matar su proceso. Un proceso, es un programa activo. En Windows, se suelen llamar servicios. Cada proceso, tiene asignado un número. Para ver los procesos utilizamos el comando: **ps aux**. El resultado es algo así:

```

Pcantonio:~ # ps aux
USER    PID  %CPU  %MEM  VSZ  RSS   TTY  STAT  START  TIME  COMMAND
root    1    0.0  0.0  588  240 ?    S    12:01  2:38  init [5]
root    2    0.0  0.0   0    0 ?    SN   12:01  0:00  [ksoftirqd/0]
root    3    0.0  0.0   0    0 ?    S<   12:01  0:00  [events/0]
root   28    0.0  0.0   0    0 ?    S<   12:01  0:00  [kblockd/0]
root  5362  0.0  0.8 21116 8348 ?    S    13:28  0:00  kdeinit: dcopserver
root  5364  0.0  0.8  2868  9188 ?    S    13:28  0:00  kdeinit: klauncher
    
```

```
root 5366 0.5 1.2 24412 12596 ? S 13:28 0:00 kdeinit: kded
root 5369 0.0 0.1 2700 1648 pts/47 Ss+ 13:28 0:00 /bin/bash
root 5376 0.1 0.1 2700 1612 pts/48 Ss 13:28 0:00 bash
root 5387 0.0 1.4 29996 15276 ? S 13:29 0:00 kdeinit: knotify
root 5388 0.0 0.0 2164 688 pts/48 R+ 13:29 0:00 ps aux
```

Los procesos de sesión de consola abiertos se reconocen por el identificador: **pts**. La segunda columna (**PID**) es el número del proceso. Mediante el comando: **kill**, matamos el proceso consola del cliente **ssh** conectado a nuestro equipo, y por tanto le expulsamos del sistema. La sintaxis es muy sencilla: **kill número de proceso**. El parámetro: **-9** mata procesos resistentes.

El fichero: **auth.log**, que se encuentra en: **/var/log/**, contiene toda la información de las sesiones de consola. Si lo abrimos, podemos utilizar toda esta información para saber lo que ha ocurrido en nuestro ordenador. Sin embargo, trabajar con tantos datos en un solo bloque es bastante pesado. Resultaría mucho más práctico poder clasificar la información según criterios apropiados. En el capítulo anterior, hemos aprendido a utilizar diversos comandos, como **ls** o **grep** (listar o filtrar respectivamente). Quizás nos parecieron un poco extraños al principio. No parecían tener una utilidad inmediata. Sin embargo, son comandos muy efectivos. Ahora vamos a ver como utilizarlos para administrar el servidor. Supongamos, que varios clientes han accedido a nuestra máquina. Queremos saber, por ejemplo, cuantas veces se conecto el usuario "**antonio**" durante el mes de mayo. Es fácil:

```
cat /var/log/auth.log | grep 192.168.0.101 | grep May
```

La IP: 192.168.0.101, se corresponde con el ordenador de: "antonio". **cat** va a mostrar el fichero **auth.log** por la salida estandar (monitor), pero solo la parte filtrada por el comando **grep**. Bueno, por el comando **grep**, y por sus parámetros. Es decir, va a seleccionar solo las entradas que tengan que ver con el número Ip, que le hemos asignado. Y de estas entradas, las correspondientes al mes de mayo. Gracias a las tuberías o pipes, podemos ir añadiendo tareas en la misma línea. Podríamos enviar un correo con la información filtrada, borrarla, imprimirla por la impresora, guardarla en otro directorio. Lo que se nos ocurra. En este ejemplo mandamos un correo al usuario jose, que tiene la IP: 192.168.0.222

```
cat /var/log/auth.log | grep 192.168.0.222 | grep May | mail jose@hotmail.com
```

El comando: **ping** es interesante, en este contexto. Se utiliza para mandar paquetes de prueba al ordenador que queramos y ver si la red funciona correctamente. Se utiliza:

```
ping nombre de máquina
```

Aparte de enviar paquetes, nos devuelve la IP de la máquina conectada. Es una forma de saber su dirección IP, partiendo del nombre de la máquina.

Nota

El comando: Ctrl + 0 detiene un programa en ejecución.

Nota

El fichero: interfaces, situado en: **/etc/network/** contiene los datos de la conexión a internet.

3. INSTALACION DE KNOPPIX.

Existen muchas distribuciones de **Gnu/Linux**. Una distribución, es un kernel de **Linux**, junto con aplicaciones básicas que permiten una administración de bajo nivel de la máquina, y aplicaciones de alto nivel (Fig 1). Cada distribución, escoge conjuntos de aplicaciones específicos y suele estar orientada a un determinado segmento de usuarios. Entre ellas, **knoppix** destaca como la distribución ideal para el nuevo usuario de **Linux**. Ello es debido, a que el disco de instalación de **Knoppix** tiene capacidad para ejecutar un sistema completo desde el propio disco CD-ROM, sin que sea necesario instalar nada en el disco duro de la máquina.

De esta forma, se puede experimentar un sistema **Gnu/linux** completo, sin ningún riesgo. Es sin duda, la forma ideal de conocer el mundo Linux. Además, **Knoppix** tiene otro par de virtudes. Es una **Debian**, lo cual le confiere una gran calidad, e incluye una enorme cantidad de aplicaciones ya instaladas y listas para ser utilizadas. Si se desea, también se puede instalar en el disco duro. Su sistema de instalación es bastante sencillo, comparado con los de otras distribuciones.

Aparte de todo esto, **Knoppix** puede ser utilizado para muchas otras cosas. Es un excelente disco de rescate, ya que podrá arrancar un ordenador, a pesar de que su disco duro no tenga capacidad de hacerlo. Como incluye programas de grabación de CD, podremos hacer copias de seguridad del sistema dañado. También es posible crear una **knoppix** personalizada, con nuestras aplicaciones favoritas, nuestros documentos de texto, imágenes, vídeos, etc. Podría ser utilizado en presentaciones, conferencias, reuniones, etc, en cualquier ordenador que disponga de lector de CD-ROM, y una cantidad de memoria suficiente. Como nada es perfecto en este mundo, **Knoppix** también presenta alguna desventaja. Es imprescindible disponer de 1,5 GB libres instalando en disco duro. Siempre instala todos los programas.

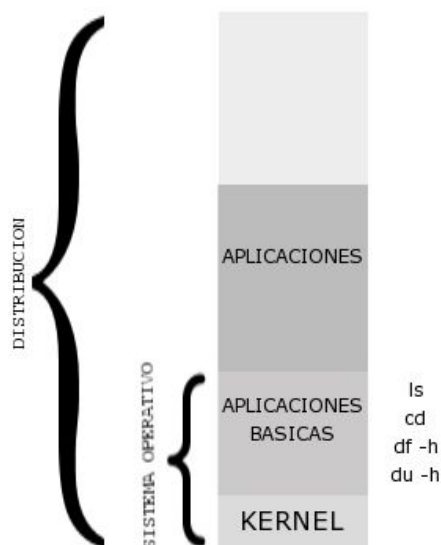


Fig 1.

Knoppix, es una combinación de las ramas estable, inestable y testing de **Debian**. Por eso, tiene tantos programas. En general, funciona bien, aunque el hecho de incorporar programas de la rama inestable, hace que no sea una distribución muy adecuada para un servidor en producción. Sin embargo, es una distro perfectamente adecuada para escritorio (uso doméstico u ofimático).

Veamos ahora el proceso de instalación. Primero arrancamos con el CD de **Knoppix**. Dejamos que arranque desde el CD. Cuando termina, ejecutamos una consola de texto y escribimos:

knoppix-installer

Este comando ejecuta un asistente gráfico de instalación bastante cómodo. Tiene tres niveles según la experiencia del usuario: novato, knoppix y debian. Incluso el modo debian (el más complejo) resulta sencillo. Este último, es recomendable ya que es el que más posibilidades permite. Hecho esto, el asistente buscará las particiones disponibles y nos pedirá que las seleccionemos. Si no tenemos una partición **swap** (intercambio) y otra **linux**, tendremos que crearlas. El instalador dispone de un programa gráfico para realizar esta acción. Sin embargo, no funciona demasiado bien. Es sencillo de utilizar, pero le faltan las prestaciones y la seguridad de sus homólogos en modo texto. Yo recomiendo el clásico **fdisk** de las instalaciones de Gnu/Linux tradicionales. Una vez que tenemos particionado el disco, hay que escoger el tipo de sistema de archivos que queremos para cada partición. Se suele utilizar **ext3** para la partición Linux. **Reiserfs** es más moderno, pero no se considera completamente seguro.

Hecho esto, se pasa directamente a la configuración de los usuarios. En primer lugar, un usuario normal, y más tarde el administrador. Habrá que introducir contraseñas para ambos usuarios. A continuación, hay un apartado para poner nombre al ordenador. Por último, hay que seleccionar donde queremos instalar el **Lilo** (Linux loader). Es recomendable escoger: **mbr** (master boot record). De este modo **Lilo** se instala en el sector de arranque del disco duro, en lugar de en una partición. A partir de

este punto, **Knoppix** empieza a instalar todos los paquetes de la distribución. Cuando finaliza, nos aparece una ventana que nos ofrece la opción de guardar la configuración en un disquete para poder utilizarla en futuras instalaciones.

Como se ha visto, la instalación de **Knoppix** es realmente sencilla. Seguramente más sencilla incluso que la de Microsoft Windows. Normalmente, los nuevos usuarios de Linux tienen bastante miedo al particionado del disco duro. Esto es lógico. La culpa no es de Gnu/Linux, sino de la necesidad de mantener intacta la partición de Windows, y la instalación de un gestor de arranque para ambos sistemas. Una instalación de Gnu/Linux en todo el Disco duro no presenta estas dificultades.

4. INSTALACION DE DEBIAN.

Vamos a realizar una instalación de **Debian**. La versión que vamos a utilizar, es **Woody release 4**. Si el ordenador tiene instalado previamente otro sistema operativo, como **Microsoft Windows**, lo conservaremos intacto en su partición. Introducimos nuestro CD-Rom de **Debian**, y arrancamos el equipo, poniendo especial cuidado en que la **Bios** esté configurada para arrancar un CD-Rom de instalación. Normalmente, solo habrá que situar como dispositivo de arranque primario el CD-Rom, en lugar del disco duro. Los ajustes de la **Bios**, se inician pulsando la tecla **Supr**, durante el arranque del ordenador. El apartado que necesitamos modificar se suele llamar: **boot loader** o algo similar. En todo caso, ello depende de la marca y versión de la **Bios** que tengamos. Hecho esto, se inicia el programa de instalación de **Debian**. Pulsamos **F3**, en el menú que aparece a continuación elegimos el **kernel 2.4**, y seguimos todos los pasos de la instalación (idioma, teclado, etc). Creamos solamente dos particiones, una para la memoria de intercambio (**swap**), y otra para el sistema. Dos particiones es lo mínimo para un sistema Gnu/Linux, aunque podemos añadir las que queramos. Es muy habitual una partición **/users**, para los usuarios, o **/home**, o lo que queramos. **/var** y **/data**, son particiones habituales, en muchos servidores. Formateamos las particiones nuevas y les asignamos los puntos de montaje adecuados. A continuación, escogemos: instalar **el núcleo y los módulos**. Habitualmente, en el caso de un Pc, con componentes comunes, no necesitaremos instalar controladores de dispositivos adicionales. En estos casos, el kernel dispone de suficiente información para manejar el hardware por sí mismo. Cuando llegamos al apartado de elegir **DHCP** o **BOOTP**, no instalamos ni uno ni otro. Un servidor **DHCP**, es aquél que nos asigna una **IP**, dentro del rango que tenga disponible. Es una forma de conectarse a internet prácticamente automática. **BOOTP**, es un método diferente, apropiado en algunos casos concretos (como los **Clusters**). La tarjeta de red, hace un broadcasting (una llamada en busca de servidores disponibles), y se autoconfigura. Así que, si instalamos el servidor **BOOTP** en una máquina, el resto de máquinas se conectarán solas. Nosotros no vamos a utilizar ni **DHCP**, ni **BOOTP**, sino que vamos a configurar manualmente nuestros datos de red. En nuestro caso son los siguientes:

IP: 192.168.0.209
Máscara: 255.255.255.0
Dominio: hileras.net
Pasarela: 192.168.0.100
DNS: 192.168.0.254
192.168.1.100
192.168.1.101
nombre: aula9
Pass: nodo50
Dirección : aula9.hileras.net

Ahora se instala el sistema base. Después, escogemos hacer el sistema arrancable (**Lilo**). No hacemos disco de arranque. No es necesario, ya que siempre podemos utilizar el disco de instalación de **Debian**, y llegar hasta el punto de la instalación que nos interese, para hacer algún cambio, sin realizar ninguna otra modificación. Más tarde, nos pregunta si queremos utilizar **GMT**. Decimos que sí. Es la hora del meridiano de referencia, que se encuentra en el Reino Unido. Como algoritmo de encriptación de contraseñas, elegimos **md5** en lugar de **Des**, un algoritmo más antiguo. Nos va a aparecer, un mensaje acerca de los peligros de utilizar **NIS** (sistema de contraseñas no locales). Es solo una nota informativa. No hay ningún error en la instalación. Ante la pregunta: **Shadows password**, respondemos Yes. Es más seguro. Esto es, que las contraseñas en lugar de guardarse en la carpeta: **/etc/passwd** (que es un directorio accesible a todo el mundo) se guarden en: **/etc/shadow**, que es un directorio accesible solo por el root. Naturalmente, las contraseñas van encriptadas, en cualquier caso.

En el siguiente paso, cuando nos pregunta si queremos instalar los paquetes **Pcmcia** le decimos que no, ya que no estamos

realizando la instalación en un portátil. Ahora vamos a pasar a una parte de la instalación muy importante: la configuración de **Apt**. Este, es el programa que gestiona los paquetes en **Debian**. Es, seguramente, el mejor gestor de paquetes de todas las distribuciones **Linux**. **Apt**, puede buscar, instalar y resolver las dependencias de cualquier paquete **.deb**, e incluso de código fuente, desde múltiples repositorios. Es cómodo de utilizar, robusto y muy capaz. El enfoque que hace **Debian** de este programa es peculiar. Como **apt** es muy flexible, la instalación de **Debian** es minimalista (instalación base). Se instala lo esencial para funcionar, se configura **apt**, y el resto ya se instalará más tarde, a medida que sea necesario. Mediante el comando: **apt-setup** vuelven a salir las pantallas de configuración de **apt**, en caso de que queramos volver a configurarlo. Escogemos como medio de instalación **ftp**, y como servidor **ftp.rediris.com** y como país Spain (**es**). Para instalar paquetes en **Debian** hay muchos métodos. Veamos algunos;

-apt -tar.gz -tasksel -apkg -dselect

No obstante, la herramienta que más se suele utilizar es **apt**. En este curso vamos a utilizar preferentemente **apt**. En cualquier caso, en el siguiente paso de la instalación de **Debian**, va a aparecer **tasksel**. Este es un programa de instalación de paquetes, que muestra un menú, con todos los paquetes agrupados dentro de categorías más o menos genéricas, como X windows system o Desktop environment. Si escogemos cualquiera de estas, **tasksel** instalará todos los paquetes asociados a la categoría seleccionada. **tasksel** puede ser ejecutado en cualquier momento desde una **shell**, mediante el comando: **tasksel**. En nuestro caso, escogemos: X windows system y Desktop environment, así como tcl-tk (conjunto de librerías gráficas esenciales para muchos programas. (Permiten, por ejemplo, configurar el kernel en modo gráfico.)

Seguimos adelante. El instalador nos va a preguntar si queremos instalar **exim**, el gestor de correo por defecto, en **Debian**. Le decimos que no. Ahora llega el momento de configurar el sistema gráfico. Se utiliza el siguiente comando:

dpkg-reconfigure xserver-xfree86

El comando **dpkg-reconfigure** sirve para volver a configurar cualquier paquete en **Debian**. Durante esta parte, tenemos que escoger el monitor, la tarjeta gráfica, el ratón, el teclado, y cualquier otro periférico que tengamos. Naturalmente, tendremos que conocer nuestro hardware de antemano. El hardware estándar no presentará ningún problema. Si tenemos algún periférico especial, puede ser que tengamos que trabajar algo más. Hecho esto, nuestra flamante distribución **Debian** estará completamente instalada.

Veamos ahora, un listado de comandos para el gestor de paquetes **apt**:

apt-get install paquete	# Instala paquete.
apt-cache search cadena de caracteres	# busca paquetes.
apt-get update	# Actualiza la lista de paquetes.
apt-cache show paquete	# Descripción máxima de paquete.
apt-cache depends paquete	# Muestra dependencias.
apt-get install /testing	# Cambio a la rama testing.
apt-get install /unstable	# Cambio a la rama inestable.
apt-get remove	# Borra. Deja ficheros de configuración.
apt-get remove --purge paquete	# Borra paquete y fichero de configuración.
apt-get upgrade paquete	# Actualiza paquete.
apt-get dist-upgrade paquete	# Actualiza todos los paquetes de la distro.
apt-get source paquete	# Descargar código fuente.
apt-get -b source paquete	# Descargar fuente y compilarlo.
apt-get autoclean paquete	# Elimina versiones anteriores.

Por último, vamos a hacer unos test a nuestros medios de instalación, en este caso servidores **ftp**. El programa que vamos a utilizar se llama: **netselect**. En primer lugar, hay que instalarlo mediante **apt**. Vamos a poner a prueba los siguientes servidores:

ftp.rediris.es
ftp.linux.co.uk
ftp.sunet.se
ftp.debian.org

El programa **netselect**, devuelve un valor numérico de cada servidor. El número más bajo, indica una respuesta más rápida por parte del servidor. Es interesante saber cual va mejor, para poder instalar los paquetes que nos interesen, de forma lo más rápida posible.

5. CONFIGURACION DE USUARIOS.

La información sobre los usuarios del sistema se encuentra en un fichero llamado: **passwd**, situado en: **/etc**. Veamos un ejemplo:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:Daemon:/sbin:/bin/bash
lp:x:4:7:Printing daemon:/var/spool/lpd:/bin/bash
mail:x:8:12:Mailer daemon:/var/spool/clientmqueue:/bin/false
news:x:9:13:News system:/etc/news:/bin/bash
uucp:x:10:14:Unix-to-Unix CoPy system:/etc/uucp:/bin/bash
games:x:12:100:Games account:/var/games:/bin/bash
man:x:13:62:Manual pages viewer:/var/cache/man:/bin/bash
at:x:25:25:Batch jobs daemon:/var/spool/atjobs:/bin/bash
wwwrun:x:30:8:WWW daemon apache:/var/lib/wwwrun:/bin/false
ftp:x:40:49:FTP account:/srv/ftp:/bin/bash
gdm:x:50:15:Gnome Display Manager daemon:/var/lib/gdm:/bin/bash
postfix:x:51:51:Postfix Daemon:/var/spool/postfix:/bin/false
sshd:x:71:65:SSH daemon:/var/lib/ssh:/bin/false
ntp:x:74:65534:NTP daemon:/var/lib/ntp:/bin/false
nobody:x:65534:65533:nobody:/var/lib/nobody:/bin/bash
antonio:x:1000:100:antonio becerro:/home/antonio:/bin/bash
yoli:x:1001:100:yolanda becerro martinez:/home/yoli:/bin/bash
snort:x:73:68:Snort network monitor:/var/lib/snort:/bin/bash
zope:x:64:101:Zope:/opt/zope:/bin/false
mysql:x:60:2:MySQL database admin:/var/lib/mysql:/bin/false
vdr:x:100:33:Video Disk Recorder:/var/spool/video:/bin/false
```

Aparte de los usuarios que hayamos creado nosotros, existen muchos otros del sistema. Son necesarios para algunos programas. Cada línea es un usuario diferente. Veamos uno de ellos:

antonio:x:1000:100:antonio becerro:/home/antonio:/bin/bash

antonio, es el nombre del usuario. **x**, es la contraseña (se encuentra en: **/etc/shadow**). **1000**, es el identificador del usuario (ID). 1000 o inferiores son usuarios del sistema. Los superiores a 1000 son el resto de los usuarios. El siguiente número, en este caso: 100, es el grupo. A continuación, tenemos el nombre completo del usuario: **antonio becerro**, su directorio personal: **/home/antonio**, y la ruta a la **bash**. La **bash** es el interprete de comandos. Este es un programa capaz de interpretar ordenes básicas fundamentales para la administración del sistema. Junto con el Kernel, estas ordenes, o comandos (incluso se les podría denominar programas) conforman un sistema operativo Gnu/Linux.

Crear un usuario nuevo es muy sencillo. Utilizamos el siguiente comando:

Adduser nombre de usuario nuevo

Nos va a pedir la contraseña, el nombre completo, y algunas cosas más, poco importantes como el teléfono, el domicilio, etc. Finalmente, tendremos un nuevo usuario. Si editamos el fichero: passwd con **Emacs**, veremos que tiene una línea nueva con los datos del usuario que acabamos de crear.

Nota
El comando: ls -la muestra ficheros ocultos.

Nota
El fichero: **.kde** en: **/home/antoniux**, contiene los datos de la apariencia del escritorio **Kde**.

6. LOS PERMISOS.

Todo sistema multiusuario avanzado, requiere un control completo sobre los derechos de los ficheros. Gnu/Linux, gestiona este aspecto de la misma forma que el resto de sistemas similares a Unix. Cada fichero tiene un propietario. Este pertenece a un grupo. El resto de los usuarios, desde el punto de vista de los permisos, conforman un tercer grupo. Los tres tipos de permisos posibles son: escritura, lectura y ejecución. Se trata de aplicar estos tipos a los grupos de usuarios del sistema, según interese en cada fichero. Parece complicado. Sin embargo, con un poco de ingenio, se puede hacer de un modo práctico y eficaz. Veamos como:

-	r	w	-	r	-	-	r	-	-	-
tipo	dueño	grupo	todos							

Todos los archivos, llevan asociado una cadena de diez caracteres que establece sus permisos con precisión. El primer carácter, indica el tipo de fichero. Un guión es un fichero normal. Una **d**, es un directorio. Los siguientes tres caracteres indican los permisos del propietario del fichero. **r**, significa lectura, **w** significa escritura y **x** significa ejecución. Los tres que vienen a continuación son los permisos del grupo (al que pertenezca el propietario), y los tres últimos caracteres son los permisos del resto de los usuarios. Para visualizar los permisos de un fichero, lo listamos en modo detallado mediante el comando: **ls -la**. El resultado es algo así:

-rw-r--r-- 1 root root 72 May 26 19:29 hola.txt

Veamos. **hola.txt**, es un fichero con permiso de lectura y escritura para el propietario, solo lectura para el grupo, y solo lectura para el resto de usuarios. Ahora, vamos a cambiar los permisos del fichero: **hola.txt**. Se utiliza el comando: **chmod**. El permiso de lectura(r) recibe el valor numérico: 4, el permiso de escritura(w) recibe el valor: 2, y el de ejecución(x) el valor: 1. La forma de otorgar los permisos consiste en introducir el comando **chmod**, un número de tres cifras y el nombre del fichero. Las tres cifras del número, se corresponden la primera con el valor de los permisos del propietario del fichero, la segunda con los permisos del grupo, y la tercera con los permisos del resto de usuarios. Veamos un ejemplo:

chmod 777 hola.txt

Bien, 777. Es decir: 4 + 2 + 1. O lo que es lo mismo, Máximos permisos (r w x) para todos los usuarios. Podemos hacer las pruebas que queramos. El número: 0, significa que no se concede ningún permiso. Para dar permisos a un directorio entero utilizamos el parámetro: **-R** (implica recursividad). Así se cambian los permisos de la carpeta: **/home**

chmod -R 750 /home

El permiso de ejecución, hace referencia a la posibilidad de ejecutar programas. Un usuario que no tenga permisos de ejecución sobre un programa no podrá utilizarlo. La administración de los permisos es una herramienta fundamental en un sistema operativo multiusuario. Los administradores, crean usuarios y grupos y les asignan permisos sobre ficheros y programas, en función del trabajo que desarrollen.

7. PROGRAMANDO UN POCO.

Existen dos tipos de programas. Los guiones de **Shell**, y los binarios. Los guiones de **Shell**, también llamados **Shell scripts**, son ficheros de texto normal, guardados con la extensión: **.sh**. En estos ficheros, escribimos líneas de comandos, como las que hemos visto en capítulos precedentes. Desde la línea de comandos, escribimos el nombre con el que hayamos guardado el **script**, y este ejecuta todas las ordenes que contenga. El lenguaje de **script** de la **Shell** presenta posibilidades del tipo de un verdadero lenguaje de programación. Estos guiones son muy útiles para administrar el sistema y para aumentar la productividad. Por poner un par de ejemplos, podemos utilizarlos para realizar copias de seguridad automáticas de nuestros programas o para ejecutar aplicaciones cuando queramos, mediante el demonio **crontab**. Incluso el sistema operativo, se inicia mediante uno de estos **scripts**. Los binarios, son programas previamente compilados. Se escriben en texto normal, respetando la sintaxis del lenguaje de programación que se utilice (**C**, **Python**, etc). Mas tarde, mediante una aplicación llamada compilador, se convierte en código máquina. Es decir, código binario (ceros y unos), que es el único lenguaje que entiende el ordenador. Vamos a ver a continuación, un ejemplo de programación para **Shell**. Es un ejemplo muy sencillo. En un editor de texto como **Emacs**, creamos un nuevo fichero, llamado: **suma**. Lo editamos, introduciendo las siguientes líneas de texto:

```
echo "BIENVENIDOS A GNU-CAL!"
sleep 2
echo "Soy tu calculadora. Tu me dices lo que quieres calcular y yo te doy el resultado."
sleep 3
echo "¿Cual es el primer valor?"
read valor1
echo "Muy bien. Introduce el segundo valor."
read valor2
sleep 2
echo "El resultado es:"
expr $valor1 + $valor2
echo "Gracias por su tiempo"
```

\$ es una variable. No tiene un valor fijo, sino que toma el valor que se le indique. En este caso, los valores posibles son: **valor1** y **valor2**. Estos, se corresponden con los números que escribe el usuario. **read**, lo que hace es leer estos números. Finalmente, la operación matemática la realiza el comando: **expr**, sobre las variables anteriores. El comando: **echo**, lo único que hace es mostrar en la pantalla la cadena de caracteres. **sleep**, indica al programa el tiempo que tiene que esperar entre la ejecución de una y otra parte del mismo. Se indica con un número en segundos. Este programa es una sencilla calculadora. Tan sencilla, que solo sabe sumar.

Lo guardamos con cambios. Desde la línea de comandos escribimos:

suma

El programa se ejecuta. Se presenta, pide un primer valor, luego un segundo. Da el resultado y finalmente se despide. Es una calculadora muy educada. Un paso previo a la ejecución de **suma**, es dotarle de permisos de ejecución. Ya vimos como hacer esto en el capítulo anterior. Ahora, vamos a cambiar de directorio. Nos encontramos en: **/root**. Ahora, vamos a **/home**. Intentamos ejecutar el programa **suma**. No funciona. Tenemos dos opciones. O bien nos trasladamos de nuevo al directorio **/root** (ruta relativa), o escribimos la ruta completa al programa (ruta absoluta).

Sin duda, seria más cómodo escribir **suma** desde cualquier parte del sistema, y ejecutar directamente nuestra calculadora favorita. Esto es muy fácil. Tan solo hay que copiar el programa al directorio: **/usr/bin**. De esta manera:

cp /root/suma /usr/bin/suma

El directorio: **/usr/bin**, es especial. Está reservado para los programas. El interprete de comandos, sabe que tiene que buscar ahí, cuando escribimos el nombre de un programa.

8. APACHE

El servidor apache, es seguramente el programa de software libre más conocido. Su éxito, radica en su robustez, y su capacidad para soportar una enorme cantidad de características en forma de módulos. Su extensa **API**, permite a los desarrolladores ampliar sus capacidades para hacer casi cualquier cosa. Nosotros vamos a instalar la versión 1 de Apache. Existe versión 2, pero todavía no se considera lo bastante estable, en entornos de producción. Utilizaremos un módulo adicional de encriptación llamado **ssl**. De modo, que la versión se llama: **apache-ssl**.

En primer lugar, instalamos el programa mediante: **apt-get install apache-ssl** El programa de instalación nos va a hacer un par de preguntas: **Server Name** y **Email Address**. La primera es la identificación de nuestro equipo. En nuestro caso: aula9.hileras.net. Como dirección de correo no ponemos nada. Los ficheros de configuración de **Debian** se encuentran en: **/etc** y los de **Apache-ssl** en: **/etc/apache-ssl** El archivo de configuración tiene el siguiente nombre:

httpd.conf

Todo lo que hay que hacer para configurar **Apache-ssl**, es editar este fichero mediante un editor de textos. Podemos utilizar **emacs**, **xemacs**, **kate**, etc. Si no nos defendemos bien en modo de texto, **kate** es un buen editor gráfico, muy cómodo de usar. El fichero **httpd.conf**, está muy comentado. Solo están activas las líneas que no empiezan por el carácter **#**, el resto son comentarios. Podemos escribir nuestros propios comentarios añadiendo líneas precedidas por: **#**. Ante nada, hacemos una copia del fichero original **httpd.conf**. Con guardarlo con otra extensión nos vale. Cada vez que hagamos un cambio, hay que parar y arrancar el servidor Apache. Estos son los comandos:

```

apache -sslctl stop           # para Apache-ssl
apache -sslctl start        # arranca Apache-ssl
apache -sslctl restart     # para y arranca Apache-ssl

```

Lo mínimo que hay que hacer para tener un servidor Apache funcionando es eliminar el carácter **#** (comentario) a la línea: **/ServerName**, y sustituir **ServerName** por el nombre de nuestro servidor. En este caso queda: **/aula9.hileras.net** Para comprobar su funcionamiento paramos y arrancamos el servidor apache, y en un navegador de internet, tecleamos esta url:

https://aula9.hileras.net

Si en lugar de **https**, escribimos **http**, no funciona. Por el momento, el puerto 80 no está abierto (puerto estandar para servidores http) por motivos de seguridad. La página web que aparece en el navegador se llama: index.html, y su ruta es: **/etc/www**, nosotros podremos crear carpetas nuevas dentro del directorio **/etc/www** para nuestras páginas web. Si creamos una carpeta llamada, por ejemplo: empresa1, entonces, la url será: **https://empresa1/aula9.hileras.net**. **Apache-ssl**, asume que la web de inicio de cualquier sitio web debe llamarse: index.html. Como se ve, la instalación y configuración básica de **Apache-ssl**, no es nada difícil.

En principio, parece buena idea personalizar la página de inicio de nuestro servidor. Podemos hacerlo en el editor de texto que estemos utilizando, o ejecutar un editor de html, como **Quanta** o **Bluefish**. Ahora, vamos a ver con un poco más de detenimiento el fichero de configuración de Apache: **httpd.conf**. Los apartados más importantes a tener en cuenta son las siguientes líneas:

```

ServerType standalone # arranca por ti mismo.
Si se desea, se puede hacer que el demonio inetd
arranque el Apache. Ventaja: es cómodo porque en
el inicio de los servicios de red se tiene Apache activo.
Desventaja: si inetd falla, Apache también y vicevers

```

```

ServerRoot # ficheros de configuración.

```

```

Timeout 300 # 300 sg. (5 minutos). Tiempo máximo
de una conexión. Si durante 300 sg. el usuario no hace

```

nada, Apache le desconecta. La idea es no perder ancho de banda en conexiones acabadas.

StartServers 5 # 5 procesos hijos en cola, como máximo. Es decir, 5 usuarios.

MaxClients 150 # 150 usuarios simultáneos, como máximo.

Listen # Es el puerto. Para http normalmente es el 80. https utiliza el 443. https es un protocolo para conexiones seguras o encriptadas.

DocumentRoot # Es la ruta a los archivos de las páginas Web. Por defecto /etc/www

Errorlog # Es la ruta al fichero de errores. Por defecto /var/log/apache-ssl/error.log Hay 7 niveles de información de error, según su importancia: debug, info, notice, warn, error, crit, alert y emerg.

Vamos a ver ahora los enlaces directos o **Alias**. estos, sirven para que un usuario teclee en la **url** de la página, a continuación del dominio de nuestro servidor, el nombre de un directorio y que se le muestre otro diferente. El que nosotros hayamos definido. ¿Para que sirve? Veamos un ejemplo. Supongamos que queremos que los usuarios de nuestro servidor puedan ver el contenido del directorio: **/etc**. Entonces, para hacer un alias escribimos:

Alias /configuracion /etc

El usuario se conecta a la url: **https://aula9.hileras.net/configuración** y el navegador le muestra el contenido de **/etc**. Esta técnica, puede servir para hacer una especie de rudimentario servidor de descarga de ficheros, al estilo **ftp**. En una carpeta ponemos los ficheros a descargar, hacemos el alias, y cuando los usuarios se conecten podrán descargarse los ficheros. Los Alias se escriben en el fichero **httpd.conf**.

Nota

El propietario de Apache (por así decirlo el administrador) es el usuario: www-data, del grupo www-data, por motivos de seguridad.

Ahora vamos a hacer un **directorio privado** accesible mediante login y password (usuario y contraseña) desde un navegador de internet. El comando para esta acción es:

```
htpasswd -c ruta a directorio
```

Entonces, el sistema te pide que le proporciones una contraseña, y que la confirmes escribiendola de nuevo. Esto, genera en el directorio propuesto, un fichero llamado: **.htaccess**. Este fichero está oculto (todos los ficheros precedidos por un punto lo están en **Unix**). No obstante es editable en un editor de texto como **emacs**. Veamos que contiene:

```
AuthName "Directorio privado"
AuthType Basic
AuthUserFile /etc/apache-ssl/htpasswd
require user antonio
```

La primera línea simplemente es el nombre. La segunda el tipo de restricción que se va a aplicar, en este caso: Basic. La tercera línea indica la ruta al directorio al cual queremos aplicar las restricciones. Finalmente, la última línea es el nombre del usuario, en este caso antonio. Si queremos crear otro usuario, simplemente cambiamos antonio por joseba. El fichero **.htaccess** siempre tiene que estar en el directorio al cual se aplican las restricciones.

Veamos, a continuación, como crear un **Virtual Host**. Esto se traduce por alojamiento virtual. La mejor manera de entenderlo es mediante un ejemplo. Unos amigos quieren alojar en nuestro servidor su web. Si hiciésemos una carpeta nueva para ellos en **/etc/www**, técnicamente estaría resuelto. Sin embargo, nuestros amigos tienen su propio dominio, y quieren que sea este, y no el nuestro el que se visualice en la url de su página. Lo que hay que hacer, es una redirección. Los usuarios que se conecten a su dominio, realmente se tienen que conectar a nuestra máquina. Lo primero que hay que saber, es que el **Virtual Host** hereda los parámetros de configuración generales. Realicemos un ejemplo. Lo primero, editamos **httpd.conf**, y añadimos el siguiente contenido:

```
<VirtualHost 192.168.0.209:80>      # mi servidor
    ServerName www.miempresa.com    # url del cliente
    SSLDisable                      # modulo de seguridad. Apagado.
    DocumentRoot /var/miempresa     # ruta a documento html
</VirtualHost>
```

Hecho esto, cuando los usuarios se conecten a: **http://www.miempresa.com**, realmente apache mostrara el contenido de: **/var/miempresa**.

Apache, tiene capacidad para hacer muchas otras cosas, mediante módulos. Estos son fragmentos de código que añaden funcionalidades nuevas. Veamos un ejemplo:

```
mod_speling    # Cuando un usuario comete un error ortográfico, presenta una lista con las
                # las opciones correctas más similares.
```

Para aplicar este modulo, editamos el fichero: **httpd.conf**, buscamos la línea **mod_speling**, borramos el carácter: **#**, y añadimos: **CheckSpelling On**. Queda así:

```
LoadModule speling_module /usr/lib/apache/1.3/mod_speling.so

CheckSpelling On
```

Si ahora hacemos una prueba, veremos el resultado. La forma más sencilla, es cometer un error a propósito. Por ejemplo, escribimos: **http://aula9.hileras.net/indexe** (en lugar de: **index**). Existen muchos más módulos para hacer otras cosas. Se utilizan de la misma forma.

Bien. Ya tenemos un servidor Apache funcionando, hemos alojado varios sitios Web, instalado módulos. Ahora, necesitamos saber si nuestro servidor está dando un buen servicio. ¿Es lo bastante rápido? ¿Soporta bien un número alto de conexiones simultaneas? ¿Sufren los usuarios demoras a la hora de visualizar los sitios que alojamos? Para poder responder a estas preguntas, es necesario instalar algunos programas especializados en test de rendimiento. Un buen programa es: **httperf** Se utiliza así:

```
httperf -hog -server aula9.hileras.net -num-conn 5000 # El número del final es opcional.
En este caso 5000 conexiones. Podemos poner lo que queramos hasta hundir el Servidor.
```

Apache, no es fácil de hundir. Es un Servidor muy robusto. Veamos otro programa, esta vez en modo gráfico: **Kdesystemguard** (es decir, el guardian de **Kde**). Puede mostrar mediante varios tipos de gráficas, el estado de la máquina en tiempo real. Es decir, que se va actualizando sobre la marcha. Las conexiones simuladas lanzadas por **httperf**, han tenido una repercusión en en el rendimiento del ordenador: procesador, memoria, etc. Con **Kdesystemguard**, lo podemos ver. Es buena idea, ejecutar al mismo tiempo **httperf** en una consola, cambiando el número de conexiones y ver como esto afecta al sistema. El parámetro **-timeout número**, comprueba el tiempo de respuesta. Mediante este método, emulamos conexiones al puerto 80

(**http**), si añadimos el parámetro **--port443**, hace lo mismo en el puerto **443**. El parámetro **--ssl** es para utilizar encriptación en la simulación. Este último parámetro, obliga a Apache a un esfuerzo notablemente superior. Veamos una posibilidad más sofisticada:

--wssess=24,6,4

Esto quiere decir: simula 24 conexiones, en bloques de 6, cada 4 segundos.

Este parámetro rompe la linealidad. Con anterioridad, el número de conexiones, ya fueran un número muy alto o muy bajo, se llevaban a cabo de una en una. **--wssess** permite conexiones simultaneas, lo cual es muy interesante, pues podremos comprobar la capacidad de nuestro servidor ante una situación, mucho más cercana a la realidad. De este modo, a medida que aumentemos el número, podremos saber los límites de nuestra máquina. Estas pruebas, son esenciales, ya que de lo contrario no podríamos configurar **Apache** de una forma apropiada. Es imprescindible saber lo que nuestra máquina aguanta, para editar el fichero: **httpd.conf** en función de nuestras necesidades específicas.

Con **Kdesystemguard**, podíamos ver unas gráficas de rendimiento de la máquina. Esto esta bien. Pero seria mucho más interesante disponer de unos informes, con la información concreta de cada conexión (hora, usuario, etc). Para esto se utilizan estos dos programas:

webalizer
analog

Webalizer y **analog**, nos informan de las conexiones recibidas. Los datos de estos programas suelen ser: IP / hora / navegador. **Webalizer** los presenta en formato html, en color, con gráficos y demás. **analog** posee un frontend llamado: **fanalog**.

Los accesos que nos van a aparecer en los informes de **webalicer** y **analog**, son las peticiones que hicimos previamente con **httperf**.

9. PLANIFICACION DE TAREAS CON PERIODICIDAD (CRONTAB).

El demonio Crontab, es un proceso o servicio del sistema encargado de ejecutar tareas cuando se lo indiquemos. Puede, por ejemplo, hacer una copia de seguridad, todos los días a las cuatro de la mañana o enviar un correo al root cada minuto. Veamos a continuación, los comandos que se suelen utilizar:

- cron / crontab** # demonio para periodicidad de tareas.
- crontab -l** # ver tareas del usuario.
- crontab -u usuario -l** # ver tareas del usuario que se desee.
- crontab -e** # editar fichero de configuración de tareas.

Estructura del fichero de configuracion de cron.

CUANDO *QUE COSA*> programa, script, comando...

	min	Horas	Dia	mes	dia (semana)
	0-59	0-23	1-31	-12	0-7 (7 y 0 es lo mismo)
ejemplo	0	12	3	*	*
ejemplo	0,3	0,4,5	*	*	*

* significa cualquiera o mejor dicho todos.

Realmente, lo único que hay que hacer es escribir en una consola de texto: **crontab -e**, y editar el fichero. Se hace siempre de la misma manera. Primero cinco valores numéricos que se corresponden con los minutos, horas, días, meses y día de la semana. Si escribimos asteriscos, quiere decirse que se realice la tarea. Finalmente, solo nos queda decirle que cosa queremos que se haga periódicamente. Si es un script, hay que escribir su ruta. **Cron**, puede ejecutar programas, comandos, etc. Lo más lógico, es utilizar scripts ya que permiten un alto nivel de complejidad y seguridad. Podemos guardarlos ordenadamente y comprobar previamente su correcto funcionamiento. **Cron**, cuenta con un fronted gráfico. Se encuentra en Kde:

menú K / sistema / planificador de tareas

10. ACCESO A DISPOSITIVOS USB.

Vamos a hacer un acceso directo a un dispositivo hardware de tipo **usb**. Puede ser una cámara de fotos, un disco duro externo, etc. Para ello, tenemos que modificar el fichero: **fstab**, que se encuentra en: **/etc**. Lo primero, creamos una copia de seguridad. Luego lo editamos mediante **Vim**. Veamos cual es su aspecto:

/dev/hdc3	/	reiserfs	acl,user_xattr	1 1
/dev/hdc1	/boot	ext2	acl,user_xattr	1 2
/dev/hdc2	swap	swap	pri=42	0 0
/dev/hdd1	/DEBIAN	ext3	ac1,user_xattr	1 1
devpts	/dev/pts	devpts	mode=0620,gid=5	0 0
proc	/proc	proc	defaults	0 0
usbfs	/proc/bus/usb	usbfs	noauto	0 0
sysfs	/sys	sysfs	noauto	0 0

Este, es un caso concreto (mi ordenador). Cada sistema, en función de su hardware, y de como se haya configurado, tendrá unas entradas u otras. La sintaxis es bastante clara. En la primera columna, se encuentran las rutas al hardware propiamente dicho. En Linux, todo lo que tenga que ver con el hardware se encuentra en: **/dev**. dev es abreviatura de: devices, que significa algo así como dispositivos. La segunda columna, es el punto de montaje. La tercera columna es el sistema de archivos que se va a utilizar (ext2, ext3, reiserfs, fat32, etc) La cuarta columna, es la más complicada. Sirve, entre otras cosas, para que un dispositivo se monte automáticamente en el inicio del ordenador o no. **noauto**, significa que no lo haga. La quinta columna son los permisos.

Bien. Vamos a incluir una nueva línea, para un dispositivo de almacenamiento móvil, tipo **USB**. La podemos incluir al final del documento, o intercalarla entre el resto. Es fundamental respetar la sintaxis. Los espacios entre las columnas son tabuladores. No utilizar la barra espaciadora. La línea que tenemos que introducir es:

/dev/sda1	/memoria_usb	vfat	defaults,user,noauto	0 0
-----------	--------------	------	----------------------	-----

El punto de montaje se llama: memoria usb. Previamente, tenemos que crear este directorio. **vfat**, es el sistema de ficheros de **windows**. Estos dispositivos, vienen ya formateados así por el fabricante. Hemos introducido el parámetro: **noauto**, para que el dispositivo se monte cuando nosotros lo insertemos en un puerto **usb**, y no antes. Finalmente, guardamos el fichero **fstab** con cambios, y realizamos un acceso directo en el escritorio al directorio: **/memoria_usb**. Para hacer el acceso directo en Kde, basta con arrastrar la carpeta sobre el escritorio y elegir la opción: enlazar, en el menú que aparece.

11. COMPILACION DEL KERNEL.

En **Linux**, la compilación de un nuevo Kernel, es una acción relativamente corriente. En otros sistemas operativos comerciales, este tipo de cosas solamente las hace el fabricante. Dos son las razones fundamentales para compilar un nuevo kernel: obtener un mejor rendimiento de la máquina, y lograr nuevas prestaciones (normalmente, soporte de hardware nuevo). Lo primero que hay que hacer, es descargar el kernel que nos interese desde internet: (**www.kernel.org**). Suele esta comprimido en **.tar.gz** o en **.bz2**. Hay que descomprimirlo en el directorio: **/usr/src**. Creamos una carpeta llamada: **Linux**, y copiamos dentro de ella el kernel

descomprimido. Es buena idea hacer un enlace simbólico (acceso directo) a esta carpeta. Más que nada, para cambiar el nombre de linux, por algo más concreto en el enlace.

Para la configuración previa a la compilación, se pueden utilizar varios programas. Todos tienen las mismas capacidades. No son unos mejores que otros. Lo que cambia, es la interface. Los más utilizados son:

xconfig	# Método gráfico (precisa las librerías QT).
menuconfig	# Método gráfico basado en menús (precisa nurses)
gconfig	# Modo texto (precisa Gtk).

Para saber que kernel estamos utilizando, escribimos el comando: **uname -a** , para saber cual es nuestro hardware: **lspci**. La información sobre la máquina se obtiene mediante:

```
cat /proc/cpuinfo
```

Comencemos. Elegimos uno de los programas de configuración, por ejemplo: **menuconfig**. Para ejecutar el programa, desde: **/usr/src/Linux**, escribimos:

```
make menuconfig
```

La cantidad de parámetros que se pueden modificar es enorme. Conviene saber, que unos apartados se relacionan automáticamente con otros, de tal forma que para poder realizar una acción, es preciso haber realizado otra previamente. Por ejemplo, si queremos activar soporte para particiones de macintosh, primero tendremos que activar un menú llamado sistemas de ficheros especiales, o algo similar. En general, lo que se hace es ir seleccionando nuestro hardware: tarjeta de red, tarjeta de video, etc. Es el momento ideal de comprobar si el nuevo kernel soporta algún dispositivo, que no pudiesemos hacer funcionar con el kernel anterior. Es

imposible dar una receta precisa de lo que hay que activar o no. Esto, va a depender de cada ordenador concreto. No hay que tener miedo. Instalar un nuevo kernel no significa destruir el antiguo. Finalmente dispondremos de los dos, y escogeremos en el **Lilo** arrancar con uno o con otro.

Si salimos de **menuconfig**, nos va a preguntar si queremos guardar los cambios. Le decimos que sí. Ya tenemos la configuración preparada. Ahora vamos a hacer algunas comprobaciones antes de compilar. Escribimos: **make dep**, Este comando sirve para comprobar las dependencias.

Ahora: **make clean**, para borrar posibles restos de otras compilaciones anteriores. Finalmente:

```
make bzImage2430
```

Este último comando es el que compila. El kernel compilado se va a llamar: **bzImage2430**, y lo encontraremos en la siguiente ruta: **/usr/src/linux-2.4.30/arch/i386/lib**. Una vez que la compilación del kernel se ha completado, hay que compilar también los módulos. Estos le añaden funcionalidades. Los comandos necesarios son, en primer lugar: **make modules**, y luego: **make modules_install**. Hecho esto, el kernel está completamente compilado. Pero para poder utilizarlo hay que enviarlo al directorio: **/boot**, que es donde se guardan los kernels con capacidad de arrancar el sistema. También tenemos que copiar el fichero: **System.map** a **/boot**.

Por último, ya solo nos queda añadir una entrada en el **Lilo** al nuevo kernel. Para ello abrimos el fichero de configuración de **Lilo** en: **/etc/lilo/lilo.conf**, mediante un editor de texto como **Vim**.

Lo único que hay que hacer es añadir al final del documento dos líneas:

```
image=/boot/bzImage2430
```

```
label=linux2430
```

Guardamos los cambios, y ejecutamos el comando: **lilo**, que actualiza la configuración. Y ya estamos listos para reiniciar el ordenador. En la pantalla del **lilo** escogemos nuestro flamante kernel nuevo y cruzamos los dedos. Normalmente, suelen ser capaces de arrancar el sistema. Otra cosa es la conexión a internet, que solo funcionará si se ha configurado correctamente la

tarjeta de red. Si algún apartado no responde a nuestras expectativas, se vuelve a ejecutar el programa de configuración (**menuconfig**) y volvemos a compilar.

Antonio Becerro Martinez.

littledog@es.gnu.org

Alcobendas. 2005.
