

CONFIGURACIÓN DE UN SERVIDOR GNU/LINUX

Nombre: Daniel Clemente Laboreo

Tutor: Joan Canudas

Curso: 2º BACH. C

Centro: IES Bruguers (Gavà)

Fecha: Enero 2003

<http://www.danielclemente.com/servidor/>

Índice

1 INTRODUCCIÓN	1
1.1 Contexto	1
1.2 Justificación	2
1.3 Objetivos	3
2 CONFIGURACIÓN DEL SERVIDOR	4
2.1 Preparación	4
2.1.1 Estado inicial de los ordenadores	4
2.1.2 Método a seguir	5
2.1.3 Elección del ordenador	6
2.2 Instalación de Linux	7
2.2.1 Elección de la distribución	7
2.2.2 Proceso de instalación	12
2.2.3 Configuraciones básicas	15
2.3 Servicios básicos	18
2.3.1 Integración con la red de Windows	18
2.3.2 Servidor web para la Intranet	25
2.3.3 Servidor de FTP	28
2.3.4 Configuración del proxy	32
2.4 Servicios secundarios	39
2.4.1 Soporte de PHP en la web	39
2.4.2 Soporte de CGIs en la web	42
2.4.3 Estadísticas de acceso a la web	44
2.4.4 Conexiones por Secure Shell	46
2.4.5 Web para Internet	48
2.4.6 Servidor DNS	52
2.4.7 Firewall	56
3 CONCLUSIONES	64
3.1 Evaluación general	64
3.2 Otras utilidades	66
4 ANEXOS	67
4.1 Comandos importantes	67
4.2 Seguridad	70
4.3 Mantenimiento	72
4.4 Glosario de términos	73
4.5 Licencia FDL de la GNU	77
5 BIBLIOGRAFÍA	78

1 INTRODUCCIÓN

1.1 Contexto

Soy un alumno del instituto IES Bruguers, de Gavà (Barcelona), usuario habitual de Linux, programador y webmaster. Mi Trabajo de Investigación consistirá en el reciclaje de un ordenador viejo de una de las aulas de informática para convertirlo en un ordenador central del instituto que ofrezca diferentes servicios a alumnos y profesores; y sólo utilizando Linux y otro software libre de código abierto.

El ordenador pondrá al alcance de los alumnos y visitantes una página web con contenidos dinámicos, diferentes formas de compartir archivos entre los departamentos, servicio de proxy para acelerar la navegación, y otras características avanzadas como los dominios DNS, las conexiones SSH al ordenador, el filtrado de paquetes, el PHP y los CGIs.

Todo esto se hará sin tener que comprar programas profesionales, ya que todo el software usado es gratuito y legal. Además, hará el instituto mucho más profesional al utilizar un sistema operativo del nivel de Linux.

El trabajo está orientado a lectores que ya tengan una cierta experiencia en cualquier distribución de Linux utilizando la consola de comandos, o, al menos, que tengan interés por aprender (esto es lo más importante). No se explicarán las órdenes básicas ya que hay muchos tutoriales más apropiados en Internet.

También supondremos que el lector entiende algo de inglés (el inglés relacionado con la informática), porque así los conceptos quedarán más claros.

1.2 Justificación

He escogido este tema por muchos motivos::

En primer lugar, me gusta mucho Linux -el único sistema operativo que uso- y pienso que todos deberían saber que existe y que hay alternativas a Windows; claro está que cada uno usa el sistema operativo que quiere, pero mucha gente no quiere saber nada sólo porque piensan que “eso es sólo para expertos”. Este trabajo no servirá para demostrar que sea muy fácil de usar, porque precisamente usaremos una de las distribuciones más complicadas: Debian. Mandrake o Knoppix son más adecuadas para los principiantes, pero no para la finalidad de este trabajo.

También he hecho el servidor porque he visto que hacía falta un ordenador central en una red cada vez más grande; hacía falta un lugar común que sirviese para compartir archivos entre los diferentes departamentos, y para simplificar las tareas de mantenimiento (instalación de un nuevo programa en cada máquina, actualización del antivirus, etc). Además, algunas instalaciones lo requerían (por ejemplo, para alcanzar la velocidad máxima de la línea ADSL había que montar un proxy).

Además, creo que poner un sistema operativo como Linux, para el que existen miles de aplicaciones profesionales desde hace tiempo, elevará el nivel de las instalaciones informáticas del centro, ya que abrirá las puertas a muchísimas posibilidades para los alumnos y profesores: se podrán hacer prácticas de programación en cualquier lenguaje, comprobar los conocimientos sobre redes y telecomunicaciones, instalar filtros e instrumentos de protección como en los ordenadores de las grandes empresas, monitorizar el estado de toda la red, y cualquier otra cosa que podamos imaginar. El ordenador será del mismo nivel que los que se pueden encontrar en las universidades de informática.

El trabajo también servirá para contribuir a la documentación de Linux en catalán (su idioma original), pero esta vez con un caso práctico hecho por apartados, y con las opiniones, problemas, ideas y errores que han aparecido durante su realización.

Finalmente, este trabajo también lo he hecho con intención de ayudar a los futuros informáticos que están aprendiendo y que quieren conocer cosas nuevas. Lo hago porque a mí me hubiera gustado enterarme antes de la existencia de Linux.

1.3 Objetivos

¿Qué se quiere conseguir con el trabajo? No sólo es montar un ordenador que lo haga todo, sino que...:

- se quiere mejorar la calidad de las instalaciones del centro, utilizando programas profesionales
- se quiere promover el uso de Linux y del software libre, siguiendo el ejemplo de Debian (la distribución menos comercial de todas)
- se quieren aprovechar los ordenadores viejos que no funcionan con otros sistemas operativos
- y, claro, se quiere montar un ordenador con las siguientes características:
 - que sea fácil de usar
 - con servidor web: uno para Internet y otro para la red interna
 - que el servidor web tenga soporte para scripts CGI (contadores, por ejemplo) y lenguaje PHP
 - que también muestre gráficamente las estadísticas de acceso a la página
 - con servidor de DNS para no tener que recordar IPs
 - con servidor de FTP para poder compartir archivos
 - que esté integrado en la red de Windows y que pueda compartir carpetas
 - que acepte conexiones remotas para administrarlo desde cualquier lugar
 - que esté protegido contra alumnos traviesos...

2 CONFIGURACIÓN DEL SERVIDOR

2.1 Preparación

2.1.1 Estado inicial de los ordenadores

La red del instituto Bruguers consta de dos aulas, una en cada piso, más los ordenadores de los diferentes departamentos que hay en ambos pisos. El servidor se montará en una de las dos aulas de informática.

Los ordenadores no son todos iguales; hay diferentes marcas y modelos: algunos tienen mejor hardware que otros o no tienen determinados periféricos (por ejemplo, no todos tienen tarjeta de sonido). Todos utilizan el sistema operativo Windows 98, pero incluso el idioma es el catalán en algunos y el castellano en otros.

Hay un ordenador que no usa Windows 98 sino Windows NT 4 y que gestiona la red y los diferentes grupos de trabajo en los que se clasifican los ordenadores. Cuando integremos nuestro servidor en esta red, lo tendremos en cuenta y lo pondremos junto a este servidor NT.

Una nota curiosa es que en la red no sólo hay ordenadores sino que también hay otros periféricos conectados, por ejemplo un JetDirect y una fotocopidora (con IP y página de configuración).

Todo esto es sólo para ver que, aunque los ordenadores y dispositivos son muy diferentes entre ellos, todos tienen algo en común: forman parte de la misma red. Por tanto, se comportarán de manera idéntica al acceder al servidor.

Esta red es de clase C (255 ordenadores como máximo), y su dirección IP es 192.168.0.0./24 (o sea, que la máscara de subred es 255.255.255.0).

Toda la red está conectada a Internet mediante 2 líneas ADSL de 2 Mbits/s cada una, pero una de ellas está desactivada. La que funciona está controlada por un router Cisco al que no podemos modificar la configuración porque pertenece a la xtec (Xarxa Telemàtica Educativa de Catalunya). Por lo tanto, para hacer un servidor web donde se pueda entrar desde Internet habrá que pedir permiso a esta entidad.

2.1.2 Método a seguir

El trabajo práctico está dividido en tres bloques: instalación, servicios básicos y servicios secundarios.

- **Instalación** : escogeremos la distribución más apropiada, borraremos todo lo que haya en el ordenador y pondremos Linux con la configuración mínima. Después empezaremos a cambiar algún parámetro y a preparar el sistema para la parte difícil.
- **Servicios básicos** : haremos que el ordenador haga las tareas más importantes según las necesidades de la red. Lo integraremos en la red de Windows con el paquete Samba, haremos que sea un servidor web (para la red interna) con Apache, pondremos un servidor FTP con pure-ftpd y haremos que funcione como proxy para poder controlar las conexiones y además acelerar la navegación.
- **Servicios secundarios** : ampliaremos los de la primera parte y añadiremos otros. Por ejemplo, haremos que el servidor web acepte conexiones tanto de la red interna como de Internet y que las trate de forma distinta. También lo mejoraremos dándole soporte para PHP y CGI's y subiremos una página a la que podremos entrar para ver estadísticas sobre las visitas al servidor (páginas más visitas, número de visitas al día, etc). Haremos que el servidor acepte conexiones por SSH (parecido a Telnet pero más seguro) con el fin de administrar el PC desde cualquier lugar. Otros puntos muy importantes de esta parte son la configuración del servicio DNS (traductor de nombres de dominio a las IPs adecuadas) y el cortafuegos (que protegerá el servidor de los curiosos).

También pensaremos en el mantenimiento del ordenador una vez acabada toda la instalación, y en cómo actuar cuando empieza a quedarse anticuado o surjan problemas.

2.1.3 Elección del ordenador

No hace falta un ordenador muy potente para que haga de servidor de una red pequeña.

Aunque podamos ver anuncios de servidores con varios GigaBytes de memoria RAM, dos o más procesadores que funcionan a la vez, y mucho más disco duro que un ordenador estándar, también es cierto que muchos particulares y pequeñas empresas compran especialmente un ordenador de segunda mano para hacer un servidor.

¿Cómo se puede explicar esta diferencia tan importante? No es muy difícil: todo depende del uso que le queramos dar al servidor. Las grandes empresas reciben cada día miles de visitas a su página web, y tienen que controlar el uso que hacen su red los cientos de ordenadores que acceden constantemente a los servicios de impresión, proxy, firewall, correo electrónico, etc.

En cambio, nuestro servidor no siempre estará usándose, y cuando tenga que responder a alguna conexión, servir páginas web o ejecutar procesos de los usuarios conectados difícilmente llegará al 100% de uso. Además, Linux -a diferencia de otros sistemas operativos muy usados- aprovecha perfectamente los recursos del ordenadores, pudiendo ejecutarse hasta en 386 con 4 Mb de RAM y menos de 100 Mb de disco duro.

En nuestro centro disponemos de un ordenador de las siguientes características:

- Procesador Pentium I a 100 MHz
- 48 Mb de memoria RAM
- Disco duro de 1'2 Gb
- CD-ROM y disquetera.
- Tarjeta de red Realtek 8139
- Pantalla a color, teclado y ratón estándares.



El ordenador asignado

Este ordenador tiene todo lo que necesitamos y más para funcionar como servidor de un instituto. No usaremos el CD-ROM ni el ratón (aunque se puede), porque con pantalla, teclado y la red ya lo podemos hacer todo.

2.2 Instalación de Linux

2.2.1 Elección de la distribución











Como Linux es software libre que puede ser modificado y adaptado por todos, se han creado muchas versiones distintas del sistema operativo, tantos que ni se pueden contar. Hay distribuciones especializadas en aspectos muy concretos, y también las hay dirigidas al público general.

En principio, para el uso que le daremos, tendrá que cumplir las siguientes condiciones:

- **Seguridad:** ¡muy importante! Ningún sistema operativo es 100% seguro (tampoco Linux) y hemos de estar seguros de que ningún hacker pueda acceder a nuestro servidor. Para mantener la seguridad habrá que hacer actualizaciones del sistema muy a menudo.
- **Fácil de actualizar:** tendremos que tener siempre las últimas versiones de los programas para corregir todos los posibles errores.
- **Estabilidad:** no queremos que se 'cuelgue': como es el ordenador central, de él depende toda la red interna y algunos servicios externos. Tenemos que tener en cuenta que puede tardar mucho en encenderse.
- **Simplicidad:** no queremos nada del otro mundo: para lo que queremos no hace falta ni usar el modo gráfico. Usaremos sólo órdenes desde la terminal. Así nos ahorraremos los problemas que da la configuración de la tarjeta gráfica.

Como no tenemos ninguna necesidad extremadamente especial, compararemos sólo las 10 distribuciones más utilizadas. Las versiones analizadas quedan anticuadas en pocos meses, pero la filosofía de los programadores y el tipo de distribución de cada una seguirá siendo el mismo.

Las 10 distribuciones Linux más usadas¹

										
Distribución	Mandrake	Red Hat	Debian	Gentoo	SuSE	Slackware	Lycoris	Beehive	TurboLinux	Caldera
	8.2 Download	7.3 Standard	3.0r0 Woody	1.2 Linux	8.0 Personal	8.1 Linux	Amethyst2 Desktop/LX	0.5.0 Linux	8.0 Workstation	3.1.1 Workstation
Precio (US\$)	25	60	-	-	40	40	20	-	124	99
Origen	Francia	USA	-	USA	Alemania	USA	USA	USA	Japón	USA
Soporte	Soporte web 30 días para la instalación	Soporte web 30 días para la instalación	Listas de correo	Listas de correo, foros	Soporte web 60 días para la instalación	En la instalación, y soporte técnico limitado	Soporte por e-mail 60 días	Listas de correo	Para TurboTools; ilimitado para la instalación	Soporte web 60 días para la instalación
CDs	3	7	7	1	3	4	3	1	7	6
Versión del kernel	2.4.18	2.4.18	2.2.20	2.4.19	2.4.18	2.4.18	2.4.18	2.4.18	2.4.18	2.4.13
Instalación	Gráfica	Gráfica	Texto	Texto	Gráfica	Texto	Gráfica	Texto	Gráfica	Gráfica
Gestor por defecto	KDE	Gnome	-	-	KDE	KDE	KDE	KDE	KDE	KDE
Tipo de paquetes	rpm	rpm	deb	src	rpm	tar.gz	rpm	tar.gz	rpm	rpm

Es curioso, pero, de estas diez, sólo las cinco primeras son las realmente conocidas -especialmente Red Hat, Suse y Mandrake- por su facilidad de instalación y uso. Es bueno que usemos una distribución conocida porque así nos costará poco encontrar manuales, ayuda o soporte técnico.

No obstante, vamos a compararlas punto por punto:

- **Precio:** los precios que vemos en la tabla son sólo para las distribuciones compradas en tiendas. Estas versiones incluyen un gran

¹ Según las estadísticas de la web <http://www.distrowatch.com/top.php> en agosto de 2002

número de CDs con todos los programas que puedan hacer falta, toda la documentación disponible, y soporte técnico. La alternativa a comprar estos paquetes es bajarse los CDs de Internet. Esta vía es exactamente igual de legal, con la diferencia de que no obtenemos documentación impresa ni soporte técnico. Por tanto, el precio lo podemos ignorar.

- **SopORTE:** el soporte técnico sólo está disponibles para los que compren el paquete entero en una tienda. Como podemos encontrar toda la información en Internet, no nos hará falta soporte.
- **Número de CDs:** incluso en las distribuciones de 7 CDs y más, sólo son imprescindibles los dos o tres primeros. Además, como tenemos una conexión a Internet rápida y queremos estar actualizados, siempre que necesitemos un programa lo bajaremos directamente. Por tanto, es otro criterio que podemos eliminar.
- **Versión del kernel:** la versión del núcleo del sistema operativo es importante sobre todo cuando tenemos problemas de hardware. Necesitaremos una 2.4.18 o superior. Si no es el caso, hemos de pasar por el tedioso proceso de bajar un kernel nuevo, compilarlo y probarlo hasta que funcione. Hay que destacar la excepción de que sistemas como Debian tengan kernels precompilados listos para ser bajados e instalados sin problemas.
- **Tipo de instalación:** como utilizaremos el Linux en modo texto (consola), la instalación también será así.
- **Gestor de ventanas:** como no usaremos el modo gráfico no nos hace falta. En caso de que más tarde quisiéramos poner uno, uno sencillo sería suficiente.
- **Tipo de paquetes:** para instalar los programas que nos bajemos, lo podemos hacer de diferentes formas:
 - **tar.gz:** estos ficheros comprimidos -llamados Tarball- contienen el código fuente del programa en lenguaje C o C++. Para instalarlo lo tendremos que compilar para nuestro modelo de ordenador, proceso que puede tardar varias horas dependiendo del tamaño del programa, y que también puede dar algunos problemas. Funciona en todas las distribuciones Linux, por eso todos los programas que busquemos estarán como mínimo en este formato.
 - **RPM:** formato de ficheros originariamente para Red Hat, pero que se ha adaptado a otras distribuciones. Es rápido y fácil de usar, pero tiene los inconvenientes de que hemos de encontrar los paquetes para nuestro modelo en concreto de ordenador. Además, suelen dar problemas de dependencias.
 - **DEB:** formato propio de Debian. Similar al RPM, pero más seguro. La gran ventaja que presenta es el llamado apt, que sirve para gestionar los paquetes instalados y añadir nuevos o quitar otros sin prácticamente ningún esfuerzo.
- **Filosofía:** algunas distribuciones (sobre todo Suse, Red Hat y

Mandrake) son muy comerciales: se anuncian por Internet, sacar a la venta packs y nuevos productos, ediciones especiales o ofrecen más soporte técnico. Por otra parte, distribuciones como Debian siguen la filosofía del software libre: no hay ninguna empresa detrás, sino que está formada por voluntarios de todo el mundo que trabajan y se organizan conjuntamente. Como no hay presión de comercial, no están obligados a sacar nuevas versiones, y se dedican más a comprobar que las que sacan funcionen perfectamente.

- **Otras características:** cada distribución está orientada a un tipo de usuario en concreto. Por ejemplo, Red Hat está orientada a grandes empresas, Mandrake a principiantes y Gentoo a profesionales. Esta última no es apropiada para nuestro caso porque hay que compilar cada uno de los programas que se instalan, y eso nos haría perder muchas horas.

Después de haber hecho esta comparación, creo que la distribución más apropiada para el instituto es Debian, porque, además de representar al software libre, es simple y muy estable. Es una de las distribuciones más aptas para hacer de servidor. Aparte de ésta, también se usa mucho FreeBSD, que es un sistema basado en UNIX (como Linux).

No utilizamos Mandrake, Red Hat ni Suse por ser demasiado orientadas a usuarios domésticos y principiantes. Gentoo es demasiado complicada y difícil de configurar, y Slackware, Lycoris, Beehive, Turbolinux y Caldera son poco conocidas (si tuviésemos un problema sería difícil encontrar medios para solucionarlo).

Debian desarrolla a la vez tres ramas de su sistema operativo: la versión estable, la inestable y la 'en pruebas'. Las diferencias entre versiones son:

- **Estable (“stable”)**: la más recomendada. Está muy probada y teóricamente no debería fallar nada.
- **Inestable (“unstable”)**: la que van mejorando los programadores cada día. No es seguro que funcione perfectamente. Cuando pasa un tiempo, se dedican a probarla a fondo (se convierte en 'en pruebas') hasta que llega a ser 'stable'.
- **En pruebas (“testing”)**: la versión inestable bloqueada, a la cual no añaden nada más y sólo se dedican a probarla a fondo. Cuando, después de unos meses, ven que puede salir al público, la convierten en 'stable'.

Cada versión lleva un nombre clave (Potato, Buzz, Rex, Bo, Slink, etc), que, por cierto, son los nombres de los personajes de la película Toy Story. En el momento de escribir esto (septiembre de 2002), la versión estable es la Woody (es la versión 3.0), la 'en pruebas' se llama Sarge (será la versión 3.1) y la inestable se llama Sid.

La que usaremos para el servidor será la versión estable; no podemos arriesgarnos probando una inestable o una 'testing' porque pueden tener bugs y otros problemas con la seguridad. Igualmente, no hemos de olvidar actualizar bastante a

menudo el sistema.

Por tanto, nos decidimos por una Debian estable.

2.2.2 Proceso de instalación

Podemos instalar Debian de diversas maneras, de las cuales las más comunes son los CDs con las imágenes y la instalación por Internet. También lo podemos comprar en alguna tienda por un precio muy reducido. Toda la información la encontraremos en <http://www.debian.org>, (en el pie de página podemos cambiar el idioma a castellano o catalán).

Como no nos hace falta más que los programas básicos, es más que rentable hacer la instalación por Internet, ya que si nos tuviésemos que bajar un CD entero (650 Mb) no lo aprovecharíamos. Además, no nos hará falta usar una grabadora de CDs, sino sólo unos cuantos disquets vacíos.

El proceso es sencillo:

1. Bajamos las imágenes de los disquets del mismo FTP de Debian. Las imágenes de tamaño disquet para la versión estable y un procesador x86 las encontraremos en:

<ftp://ftp.debian.org/debian/dists/stable/main/disks-i386/current/images-1.44>

En este directorio hay muchos ficheros .bin de 1'4 Mb cada uno. Para empezar necesitamos bajar los siguiente; [driver-1.bin](#), [driver-2.bin](#), [driver-3.bin](#), [driver-4.bin](#), [rescue.bin](#) y [root.bin](#).

NOTA: Si podemos usar otro ordenador a la vez, no hace falta usar un disco distinto para cada fichero BIN: podemos hacerlo sólo con dos si los vamos turnando de manera que mientras uno se está leyendo el otro se esté grabando, y viceversa.

2. Grabamos las imágenes en disquetes de 1'4 Mb. Esto lo podemos hacer desde Linux con el comando `dd if=nombre-de-la-imagen.bin of=/dev/fd0` o desde Windows con programas como rawrite2 u otros que transfieran el archivo byte por byte hacia el disquete.
3. Insertamos el disquete correspondiente a [rescue.bin](#) y encendemos el ordenador. Cuando aparezca `boot:` pulsamos Intro. Al cabo de un tiempo nos pedirá el disquete correspondiente a [root.bin](#) y continuará la instalación.
4. Vamos siguiendo la instalación de forma normal, fijándonos especialmente en los siguientes puntos:
 - Particiones: son necesarias dos: una swap (tipo 82) no muy grande (más o menos el mismo número de Mb que la RAM del ordenador, aunque un

valor entre 50 y 100 Mb. ya va bien), y el resto del disco destinado a la otra partición, de tipo Linux Native (número 83), con punto de montaje en la raíz (“ / “). Si es posible, mejor crear primero la Native y luego la swap para poder llamarlas después `/dev/hda1` y `/dev/hda2` respectivamente. También hay que marcar la Native como bootable.

- Cuando pregunte dónde está el kernel, le diremos que en la disquetera: `/dev/fd0` y vamos metiendo los disquets que nos pida.
 - Módulos del kernel: como mínimo hay que añadir ahora los drivers para la tarjeta de red. Si no lo hacemos, no podremos instalar nada desde Internet. En nuestro caso añadiremos el módulo `rtl8139` de la categoría `net`. Si queremos añadir alguno más (para usb, sistemas de archivos, impresoras, dispositivos especiales, etc) lo podemos hacer ahora, aunque una vez instalado también es muy fácil añadir y quitarlos con `modconf`.
 - Nombre de host: lo podemos cambiar en cualquier momento, pero mejor decidir uno ahora y no cambiarlo. Por ejemplo, `bruguers`.
 - IP del ordenador: normalmente se pone la IP de la red, acabada en un número pequeño. Por ejemplo, en nuestro caso podemos poner 192.168.0.2 (192.168.0.1 es para el router).
 - Lilo: no hace falta porque sólo tenemos un sistema operativo, pero tampoco pasa nada por instalarlo. Irá bien si queremos poder arrancar con diferentes kernels.
 - Disco de arranque: no hace falta, pero siempre va bien tener uno.
5. Después de reiniciar, continuaremos con la personalización del sistema. Seleccionaremos las opciones recomendadas, asegurándonos de activar la opción de contraseñas shadow.
6. A la hora de crear usuarios, tendremos que escribir la contraseña de root., el usuario más importante del sistema. Es extremadamente importante escoger una buena contraseña. Además, por seguridad no trabajaremos siempre con el usuario root (es peligroso) sino que nos crearemos otra cuenta de usuario normal (con nuestro nombre o nick).
7. Cuando nos pregunte desde dónde instalar los paquetes, le decimos que por ftp y escogemos uno de la lista (por ejemplo, ftp.debian.org). Entonces, Debian utilizará `apt` para bajar las últimas versiones de cada programa. Tendremos que decidir qué grupos de programas decidimos instalar. Seguiremos estas indicaciones:
- No nos harán falta las X (para el modo gráfico), pues todo lo necesario se puede hacer desde consola.
 - Juegos y programas de ocio tampoco; con la configuración del Linux tendremos entretenimiento para un buen rato.

- Servidores web, DNS, mail, etc. los instalaremos individualmente en los diferentes apartados, o sea que tampoco deben ser marcados.
- Herramientas C y C++: son básicas para compilar los programas. Se tienen que instalar.
- Entorno en español: opcional. Si lo marcamos, algunas páginas de ayuda saldrán en español, pero quizás no estén tan actualizadas.

Marcando pocos paquetes nos aseguramos de que no quede instalado nada que no nos haga falta. Si no lo hacemos así, al acabar la instalación serían tantos los servicios disponibles que alguna persona se podría aprovechar y entrar al servidor sin permiso.

8. Cuando todo esté listo, aparecerá la lista de paquetes a bajar y lo que ocupan. Le decimos que continúe y dejamos el ordenador encendido mientras baja cada paquete junto con sus dependencias. Cuando acabe, tendremos que configurar algunos mientras que otros se descomprimarán e instalarán automáticamente. Si pregunta cómo configurar el correo, le diremos que no lo queremos configurar; así evitaremos muchos bugs innecesarios.
9. Al final acabaremos en la pantalla de login, que aparecerá cada vez que encendamos el ordenador para pedirnos el nombre de usuario y la contraseña.

2.2.3 Configuraciones básicas

Ahora que tenemos el sistema operativo instalado hay que hacer unas configuraciones sencillas, que deben hacerse antes de empezar a poner programas y demonios. Lo primero que hace falta es identificarse como root (con la contraseña que pusimos en la instalación).

Lo que haremos será:

- **Configuración de apt**: apt es el sistema de control de paquetes exclusivo de Debian. Cada vez que haga falta instalar un programa, sólo habrá que escribir su nombre y apt se conectará al FTP de Debian, bajará la última versión y sus dependencias, haciendo la instalación sin ningún problema.

Por defecto ya tenemos una buena configuración de los servidores en el fichero `/etc/apt/sources.list`, pero podemos ejecutar `apt-setup` (como root) y decidir según nuestras preferencias si queremos tener software totalmente libre (estilo Debian) o aceptamos cualquier programa, aunque tenga partes de código cerrado. También hemos de escoger un servidor. Preferiblemente usaremos el de Estados Unidos `ftp.debian.org` (el central) porque los situados en España suelen ser más lentos.
- **Actualización de paquetes**: una vez definidos los servidores de apt, haremos `apt-get update` para actualizar la información sobre los nuevos paquetes con el servidor. No se instalará nada aún; sólo se sincronizan.

Para actualizar los paquetes instalados con las nuevas versiones disponibles hay que hacer un `apt-get upgrade -u` (el `-u` es para mostrar los nombres). Después de mostrar la lista de paquetes a bajar, respondemos (Y/n) y sólo habrá que esperar a que acabe. Puede que mientras los instale aparezca alguna pregunta, pero normalmente da suficiente información para saber qué hay que responder.

Si queremos algún paquete adicional sólo hay que hacer `apt-get install nombre`. Podemos instalar programas en cualquier momento, a medida que los necesitemos. En el anexo hay una lista de programas recomendados. También es recomendable instalar ahora `gpm` para poder utilizar el ratón en la consola (se configura con `gpmconfig`).
- **Locales**: podemos decirle a Debian que preferimos los programas y la ayuda en español haciendo un `dpkg-reconfigure locales` y seleccionando `es_ES` y `es_ES@euro`. Así podremos usar acentos y otros símbolos especiales como el del euro, además de ver la mayoría de

mensajes y programas en español y no en inglés.

- **Hora del sistema** : no es sólo para poder consultarla y no tener que mirar el reloj: en Linux hay muchas acciones que dependen del tiempo. Por ejemplo, `at` y `crontab` ejecutan tareas programadas puntuales o periódicamente. Podemos ver la hora y la fecha con `date`. Recordamos que nuestra zona horaria es la CET, y que tiene un desfase de +1 h. respecto de la UTC (que podemos ver con `date -u` y que debe ser de una hora menos que nuestra hora).

Para cambiar el día y la hora podemos poner comandos como: `date -s "25/12/2003"` o `date -s "13:15:42"`

- **Tareas programadas** : hay acciones que se ejecutan cada día, cada semana o cada mes; el problema es que quizás el ordenador no esté encendido a las horas que hay puestas por defecto (6:25, 6:47 y 6:52). Sólo hay que cambiar en `/etc/crontab` el segundo número de cada línea -que representa la hora de una acción programada- por una más apropiada que a las 6 de la mañana; por ejemplo, las 10 de la mañana.
- **Hagamos más cómoda la shell** : al principio de `/etc/profile` podemos poner las líneas que se ejecutarán cada vez que iniciemos sesión. Algunos comandos útiles que podemos poner son `alias ls="ls --color"` para ver el listado de ficheros en colores sólo escribiendo `ls`, o `setleds +num` para activar NumLock. Podemos definir muchos más alias, como `alias "cd.."="cd .."`, `alias i="apt-get install"`, `alias u="apt-get update && apt-get upgrade -u"`, y muchos más. Para órdenes más largas, mejor hacer un script y colocarlo en el PATH (por ejemplo en `/usr/bin`).
- **Protección de algunos archivos importantes** : hay archivos que un usuario normal no tiene que necesitar ver. Por eso quitaremos los permisos de lectura, escritura o ejecución de cada uno de ellos, usando el comando `chmod`.
`chmod o-x /etc/passwd /etc/exports /etc/*netd.conf`
(`/etc/passwd` tiene información sobre los usuarios del sistema y sus privilegios, `/etc/exports` dice a qué directorios se puede acceder remotamente, y `/etc/inetd.conf` o `xinetd.conf` dice qué servicios se cargan cada vez que se enciende el PC).
- **Instalación de un kernel nuevo, si hay** : no es necesario si no tenemos problemas con el actual, pero es recomendable porque soluciona problemas de seguridad que afectan a todo el sistema. Además, algunas cosas cambian, como por ejemplo la forma de implementar las reglas del

cortafuegos que montaremos (con el 2.4). Podemos ver la versión del kernel actual con `uname -a`. Los nuevos kernels se encuentran precompilados, o sea, que sólo hace falta bajarlos con `apt-get`. Para saber el nombre y tipo de kernel que necesitamos, podemos buscar en la base de datos local de paquetes con `apt-cache search kernel-image` y fijarnos en todos los disponibles. La versión ha de ser superior al actual y la plataforma ha de ser la de nuestro sistema (p. ej. 386 para un 80386, 586 para Pentium I, 686 para Pentium 2, 3, 4 y Celerons, k6/k7 para AMD, etc.). No hay que bajar una versión SMP ya que estas siglas son de “Symmetric Multiprocessor”, cosa que no tenemos.

Pero antes hay que hacer una pequeña modificación en `/etc/lilo.conf` (si no la hacemos ahora nos lo recordará al bajar el kernel). Se trata de buscar la línea donde pone `image=/vmlinuz` y añadir justo debajo (o en el mismo párrafo) la opción `initrd=/initrd.img`. Entonces ya podemos hacer un `apt-get install kernel-image-versión-arquitectura` (ej. `apt-get install kernel-image-2.4.18-686`) y pedirle que nos cree el enlace simbólico `initrd.img` cuando lo pregunte.

A partir de ahora el Lilo (gestor de arranque) nos pedirá si queremos entrar en Linux o en LinuxOld, la versión antigua del kernel. Si no nos funciona una podemos entrar usando la otra. Es recomendable reiniciar ahora y probar el nuevo kernel; si no funcionase habría que hacer predeterminado al otro modificando `/etc/lilo.conf`

2.3 Servicios básicos

2.3.1 Integración con la red de Windows

Como en el instituto los ordenadores están conectados mediante una red y un servidor Windows NT, sería interesante incluir también nuestro servidor Linux para poder compartir carpetas o acceder a las de otros ordenadores. También pondremos al alcance del servidor las impresoras compartidas en la red, aunque habrá que configurarlas en Linux.

Todo esto lo haremos fácilmente y de forma segura con las herramientas que nos ofrece Samba. Samba es una implementación para Linux del protocolo SMB (Server Message Block), creado por IBM en 1985, redefinido después por Microsoft, y presente en otros sistemas operativos. Con Samba podremos acceder (por TCP/IP) a servidores SMB como cliente, o montar un servidor SMB propio.

Como siempre, hacemos un `apt-get install samba` y decimos que sí queremos que nos haga unas preguntas para adaptar el fichero de configuración `smb.conf`, aunque después los revisaremos.

Nos preguntará las siguientes opciones:

- Grupo de trabajo: hay que poner el mismo que el de las otras máquinas Windows. Para evitar problemas, mejor lo ponemos tal como sale en los demás ordenadores (probablemente esté todo en mayúsculas). En nuestro caso es `99PIENT22.DOM`
- ¿Utilizar contraseñas cifradas? ¡Sí! Si no lo hacemos, aparecerán problemas extraños cuando intentemos entrar a recursos compartidos de un Windows NT. Además, es obvio que las contraseñas cifradas dan más seguridad que las estándar.
- ¿Cómo queremos que se ejecuten los procesos de Samba, como demonios o como una parte del demonio `inetd`? Es mejor que se ejecuten como demonios (como los otros servidores), ya que así se podrán controlar mejor.
- Crearemos el fichero de contraseñas cifradas tal como nos pide.

Después de esto ya tenemos los dos procesos de Samba funcionando: `smbd` y `nmbd` (hacen referencia a los demonios de SMB y Netbios respectivamente). Si queremos que Debian nos vuelva a preguntar todo lo anterior podemos hacer un `dpkg-reconfigure samba`

Con Samba podemos hacer básicamente cuatro operaciones:

- Compartir una unidad Linux con máquinas Windows

- Compartir una unidad Windows con máquinas Linux
- Compartir una impresora Linux con máquinas Windows
- Compartir una impresora Windows con máquinas Linux

Hay que tener muy claro qué queremos que haga el servidor, y actuar en consecuencia. Si ponemos configuraciones por defecto o cambiamos cosas sin saber qué son, no funcionará exactamente como queremos, y arreglarlo puede costar mucho.

En nuestro caso, las decisiones que hemos tomado son:

- Hemos de compartir sólo una carpeta del servidor (con sus directorios) a todos los usuarios de Windows de la red. Otra alternativa es hacer que cada usuario de Windows pueda acceder a su carpeta personal del servidor, pero como hay muchos preferimos compartir todo en un lugar común que permita interactuar a todos los usuarios, y dejar el FTP para necesidades más concretas.
- El servidor estará preparado para acceder a los recursos compartidos de cualquier ordenador, y para montarlos como si se tratara de un disquete o un CD.
- No tenemos ninguna impresora conectada al servidor, por tanto Linux no tiene que compartir ninguna.
- Lo que sí que podemos hacer, si tenemos tiempo, es que una impresora conectada a la red de Windows se pueda usar también en Linux, por si acaso hace falta imprimir algo importante directamente desde un programa no disponible para otros sistemas operativos. El problema principal por el que no lo haremos es que hay que configurar la impresora en Linux.

Ya avisamos que es muy probable que salga un problema (muy común) si usamos tanto máquinas Windows NT/2000 como Windows 9x a la vez, debido a su forma de enviar las contraseñas y a los requisitos de cada implementación. Para entender estas complicaciones hay que ver unas cuantas diferencias entre el SMB de Win98, WinNT y Samba:

- Los Windows 98 envían contraseñas encriptadas por defecto, mientras que Samba las suele recibir en formato de texto normal. Esto tiene solución fácil añadiendo la directiva `encrypt passwords = yes` al fichero de configuración.
- Los Windows NT y Samba son muchísimo más configurables que el SMB de Windows 98 (que casi no tiene opciones). Entre otras cosas, con WinNT y Samba podemos especificar el nombre de usuario y contraseña con los que queremos acceder a un recurso compartido. Esto quiere decir que con Windows 98, la única forma de acceder con un nombre de usuario en concreto es ser ese usuario (de todas maneras, crear un nuevo usuario cuesta poco).

- Samba y Windows NT no aceptan 'invitados' (usuarios no autenticados) por defecto. Los podemos activar, pero provocan muchos problemas de seguridad (por eso se han desactivado). Este es un tema muy delicado en Windows NT, donde es muy fácil encontrar sistemas a los que se puede entrar usando `Invitado` (o `guest`) como nombre de usuario y dejando en blanco la contraseña ...

Bien, pues podemos empezar: lo primero de todo será crear un usuario en el servidor que sea el único que pueda tener acceso a los recursos compartidos. Es el usuario que usarán todos los ordenadores que accedan, y, tal como hemos dicho antes, es necesario que este usuario esté creado en todos los Windows y tenga la misma contraseña. Podemos hacer una excepción con los Windows NT ya que, como hemos explicado antes, permiten escribir un nombre de usuario y contraseña al acceder, independientemente del nombre del usuario conectado. Nota: no hace falta que vayamos ordenador por ordenador creando este usuario, ya que se creará automáticamente cuando alguien ponga su nombre y contraseña en la pantalla de inicio de sesión

En nuestro caso a este usuario lo llamaremos `bruguers` y le daremos una contraseña pública y fácil de memorizar (¡pero no de adivinar!). La contraseña es opcional, pero el no poner representa un agujero de seguridad muy importante (sobre todo cuando hablamos de sistemas Microsoft).

Añadimos el usuario al servidor con `adduser bruguers` (no es necesario escribir ningún dato más), y lo añadimos a la lista de usuarios de Samba -que se encuentra en `/etc/samba/smbpasswd`- con el comando `smbpasswd -a bruguers`, y poniendo la misma contraseña.

Ahora hemos de editar el fichero de configuración `/etc/samba/smb.conf`. Al principio encontraremos las opciones que pusimos al instalar Samba; mejor comprobamos que haya un `workgroup = NOMBRE_DEL_GRUPO_DE_TRABAJO`. Además, la siguiente opción, `server string`, nos permite poner la descripción del ordenador que se verá al navegar por la red. Por defecto pone `server string = %h server (Samba %v)`, donde `%h` es el nombre de host del equipo y `%v` la versión de Samba.

Habrà una línea comentada con el carácter `;` que pone:

```
; guest account = nobody
```

Es bueno saber que si la cambiamos a `guest account = bruguers` haremos que dejar el nombre de usuario y contraseña en blanco equivalga a entrar con el usuario `bruguers` (sin contraseña). Como hemos decidido que había que poner contraseña, dejaremos la línea comentada, pero si vemos que es más fácil sin, sólo hay que modificar eso.

En esta sección también hay que añadir la siguiente línea para evitar uno de los métodos de intrusión más usados desde hace muchos años: el de los recursos compartidos.


```
hosts allow = 192.168.0. localhost
```

Y también haremos que sólo permita el acceso al usuario `bruguers`. Si se crean más habrá que ponerlos aquí también.

```
valid users = bruguers
```

Ahora localizamos la sección titulada `Share Definitions` y, en concreto, este fragmento:

```
[homes]
    comment = Home Directories
    browseable = no

# By default, the home directories are exported read-only. Change next
# parameter to 'yes' if you want to be able to write to them.
    writable = no

# File creation mask is set to 0700 for security reasons. If you want to
# create files with group=rw permissions, set next parameter to 0775.
    create mask = 0700

# Directory creation mask is set to 0700 for security reasons. If you
# want to
# create dirs. with group=rw permissions, set next parameter to 0775.
    directory mask = 0700
```

Según nuestros intereses, lo cambiaremos a (omitimos los comentarios):

```
[homes]
    comment = Home Directories
    browseable = yes
    writable = yes
    create mask = 0777
    directory mask = 0777
```

Todo ese fragmento hace que se compartan las carpetas personales de los usuarios (`[homes]`). En nuestro caso sólo hay un usuario, que se llama `bruguers`. Éste tiene permiso para escribir en su directorio (`writable=yes`) y todo lo que se cree podrá ser leído, modificado o ejecutado por cualquier usuario del sistema (por eso ponemos los permisos a `777`, o, lo que es lo mismo, `u=rwx g=rwx o=rwx`). El `browseable=yes` hace que el recurso se pueda ver al navegar por los recursos compartidos del servidor.

El siguiente párrafo, que empieza por `[printers]`, lo comentaremos entero

con un carácter # en cada línea, porque no tenemos impresoras para compartir conectadas al servidor.

Reiniciamos el servicio con `/etc/init.d/samba restart` y probamos a acceder desde un ordenador Windows explorando la red (en 'Entorno de red' o 'Mis sitios de red'). Veremos un ordenador llamado 'Bruguers', y dentro, la carpeta compartida, donde podemos entrar y dejar archivos. También veremos el icono para gestionar las impresoras.

Probablemente no funcione a la primera debido a las diferencias entre cada Windows. Los problemas que hay entre Samba (Linux) y Windows son los mismos que los que Windows 98 y Windows NT se ocasionan mutuamente. Básicamente, se resumen en: “se puede acceder desde 98 a NT pero desde 98 a NT no”. A continuación presentamos los más típicos, junto con algunas posibles soluciones:

- En Windows NT/2000, no deja entrar en el servidor ('Acceso denegado').

Probablemente esté desactivada la opción de utilizar contraseñas encriptadas. Hay que añadir `encrypt passwords = yes` al fichero de configuración. No deberíamos tener ningún problema más con Windows NT/2000, ya que es el sistema operativo que mejor ha desarrollado el protocolo SMB.

- En Windows 98, no se ve ningún ordenador de la red, y al entrar en “Toda la red” da un mensaje de error.

La causa es que al entrar en Windows, cuando pedía nombre de usuario y contraseña, hemos escogido 'Cancelar', y por tanto no hemos iniciado sesión dentro de la red Microsoft. Hay que entrar como un usuario con nombre (y contraseña opcional). Sólo entrando ya se habrá creado el usuario, pero los podemos gestionar desde 'Mi PC' -> 'Panel de control' -> 'Usuarios'.

- En Windows 98, se ve el servidor, pero al intentar entrar pide la contraseña del recurso `IPC$`



El famoso cuadro IPC\$

No hay que poner ninguna contraseña especial; ésta es la forma que tiene Windows de decir que estamos intentando entrar como un usuario no permitido en el servidor. La solución es crear en la máquina Windows un usuario con el mismo nombre y contraseña que el que se permite en el servidor, o hacerlo a la inversa: añadir al servidor el usuario

que se conectará desde Windows.

Otra solución es activar el usuario invitado, llamado 'guest' o 'Invitado' y

con contraseña en blanco; pero hemos de evitar hacer esto a toda costa ya que pondría en peligro las unidades y carpetas compartidas.

Sin duda es más fácil crear el usuario en Windows 98: sólo hay que escribir `bruguers` en vez de cualquier otra cosa en el cuadro de inicio de sesión y poner la contraseña correcta para crear este usuario.

Por tanto, lo dejamos de forma que sólo los usuarios autenticados puedan acceder a los recursos compartidos del servidor; si alguien no pone nombre de usuario al iniciar sesión o se inventa otro es que no tiene permiso para entrar.

Otras utilidades que nos irán bien son `testparm` para comprobar la configuración y `smbstatus` para ver quién está conectado.

Nos queda explicar cómo acceder a la red de Windows desde Linux. No es difícil, pero hay muchas opciones de todo tipo. Por ejemplo, hay programas gráficos como `komba` y `xfamba`, y también se puede hacer desde consola. Algunos comandos son:

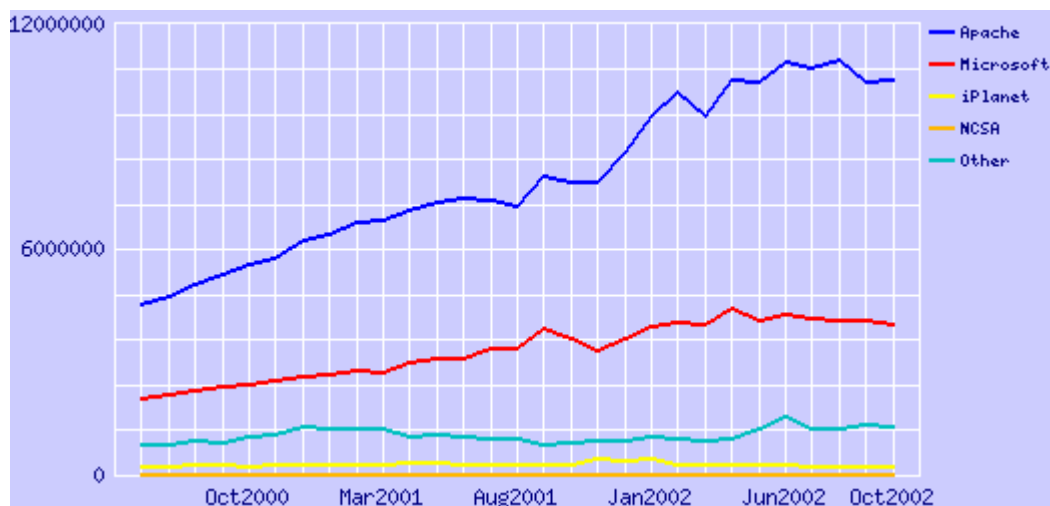
- `smbclient -L host` mostrará los recursos compartidos del equipo `host`. Podemos especificar el usuario (la contraseña la preguntará) con `smbclient -L host -U NombreUsuario`
- `smbmount //host/nombredelrecurso /mnt/samba` monta la carpeta o unidad compartida especificada en el directorio local que se le indique (que ha de existir), como si fuese un disquete. Después podremos acceder de manera normal, y copiar archivos, borrar, crearlos, cambiar los permisos, etc. Nota: para especificar el nombre de usuario hay que usar `smbmount //host/nombredelrecurso /mnt/samba -o username=NombreUsuario`
- `smbumount /mnt/samba` desmonta el recurso. Hay que hacerlo antes de que se apague el ordenador Windows porque sinó saldrán mensajes de error.
- `nmblookup host` nos da la IP del equipo `host`, presente en la red.
- `nbtscan 192.168.0.0/24` escanea toda la red (de tipo C, con máscara 255.255.255.0) y muestra los equipos que comparten recursos.

Pero, como hemos dicho, programas como `komba` o algunos navegadores de archivos con soporte SMB hacen esta tarea mucho más fácil. Eso sí, requieren usar el modo gráfico, pero no es necesario que se ejecuten en el servidor con Debian; como estamos trabajando en red, ésta se podrá explorar desde cualquier ordenador.

2.3.2 Servidor web para la Intranet

El servidor web es un demonio que escucha en el puerto de HTTP (el 80 TCP) y responde las peticiones de documentos HTML (u otros formatos). En el mercado hay muchos, y en concreto que funcionen bajo Linux también (Jigsaw, GoAhead, Roxen, Stronghold, Zeus, Abyss, Apache, ...). Incluso podemos programar uno sencillo con Netcat, haciendo que escuche en el puerto 80 y devuelva cada página pedida. Pero en Internet los servidores más usados son claramente dos: Apache y Microsoft IIS (Internet Information Server). Obviamente, IIS es sólo para Windows, así que para nuestro ordenador usaremos Apache para Linux.

¿Es una buena elección? Podemos consultar las estadísticas sobre los servidores más utilizados mundialmente en una importante página dedicada sólo a este tema: de <http://www.netcraft.com/survey> obtenemos este gráfico:



Número de servidores (todos los dominios) Junio 2000 - Octubre 2002

Hay que añadir que “Microsoft” incluye todos los servidores web de esta marca (no sólo IIS), y que iPlanet es el conjunto de todos los servidores Netscape e iPlanet. Esto deja bastante claro que Apache es el mejor servidor y que no nos hemos equivocado en la decisión.

De momento la web la haremos accesible sólo a la red interna; desde Internet no se podrá acceder porque el router no tiene abierto el puerto 80. Más adelante estudiaremos como poner dos páginas distintas, una para la red interna y otra para Internet.

Integrar Apache en Debian es tan sencillo como hacer un `apt-get install apache` (se supone que la base de datos de apt-get ya está actualizada mediante `apt-get update`) y automáticamente se bajarán todos los paquetes necesarios. Podemos

hacer una pequeña comprobación escribiendo la IP del servidor en el navegador de cualquier ordenador de la red. Si lo hemos hecho bien, aparecerá una página de prueba de Apache.

No es buena costumbre dejar la configuración por defecto, ya que podría no hacer todo lo que queremos o, peor aún, hacer más cosas de las que suponemos. Por lo tanto, vamos a revisar el archivo de configuración del demonio HTTP en [/etc/apache/httpd.conf](#):

En el archivo ya podemos ver algunos parámetros preconfigurados. Por ejemplo:

- La configuración está en [/etc/apache](#)
- El directorio de la web es [/var/www](#)
- Se usa el puerto 80
- Los logs (registros) están en [/var/log/apache/access.log](#)

Lo único que hace falta cambiar es:

- En [ServerAdmin](#) podemos poner (o quitar) el e-mail del administrador, que saldrá cuando haya errores. (Por ejemplo, "Error 404. La página no existe. Si continua teniendo problemas, contacte con el administrador: [email@email.com](#)").
- [ServerName](#): podemos poner el nombre de host o la IP. Todos los hosts se graban en [/etc/hosts](#), pero no hace falta modificar el fichero porque la IP del servidor ya tiene un nombre de host asociado (es el que nos preguntó en la instalación). En nuestro caso es [bruguers](#), por tanto escribiremos [ServerName bruguers](#). Si no lo hacemos, cogerá la IP del ordenador, pero saldrán mensajes de aviso cada vez que se inicie el servidor.

No hay que hacer más cambios, pero antes de reiniciar el servidor borraremos el contenido del directorio donde está la web ([/var/www](#)) y pondremos una sencilla creada por nosotros, con enlaces e imágenes. A la página principal la llamaremos [index.htm](#) para ver si también funciona (la original era [index.html](#)). Por ejemplo, podemos hacer un [index.htm](#) muy sencillo así:

```
<BR><BR>
<CENTER>Has entrado en el <B>nuevo servidor web</B> del
instituto.<BR><BR>
<IMG SRC="obreros_trabajando.jpg" ALT="En construcción">
</CENTER>
```

Esta página ya nos sirve para probar si acepta nombres largos de fichero, si acepta extensiones que no sean HTM/HTML, si se ven las imágenes, si el texto llega como HTML (y no como texto plano) y si un .HTM funciona bien como página principal. Sería interesante darse cuenta de que también es 'Case sensitive', o sea, que [index.HTM](#) es diferente de [index.htm](#). Lógicamente, la razón es que el sistema de

archivos de Linux ya es así.

Después de colocar en `/var/www` el fichero `index.htm` y alguna imagen, reiniciaremos el servidor haciendo un `apachectl restart` y volveremos a comprobar en el navegador si sale la página correcta. correcta.

Y por último, crearemos un usuario llamado `web`, porque después pondremos un servidor FTP, y nos irá muy bien poder administrar los ficheros de la página web desde el FTP. Hay que hacer un `adduser web` como root, poner una buena contraseña, y después modificar el `/etc/passwd` para cambiar el directorio de inicio a `/var/www`, que es donde se encuentran todos los archivos de la web. Podemos borrar el directorio HOME creado por defecto, `/home/www`. También tenemos que hacer suyos todos los archivos de la web, ya que si el directorio pertenece a root ningún usuario lo podrá modificar. Lo hacemos con `chown web.web /var/www -R` (el `R` hace que cambie los permisos de forma recursiva: al directorio y a todo lo que haya dentro).

Este Apache en principio sirve webs a cualquier PC (local o remoto), por tanto habrá que cerrar el puerto 80 del router si queremos que sea accesible sólo a la Intranet o abrirlo si lo queremos hacer público.

2.3.3 Servidor de FTP

El FTP sirve para transferir archivos de forma rápida y sencilla entre ordenadores. Permite compartir -con nombre y contraseña- todos los archivos o sólo algunas carpetas a cada usuario, pudiendo establecer los permisos que tienen sobre cada elemento. Por ejemplo, podemos hacer que sólo puedan bajar cosas, o sólo subir y no cambiar nada.

Servidores de FTP para Linux también tenemos muchísimos. Algunos muy conocidos son `proftpd` y `wu-ftp`, pero como se han encontrado muchos bugs que afectan a características extra normalmente no utilizadas, mucha gente prefiere servidores más sencillos que cumplan únicamente su propósito de compartir archivos.

Por lo tanto, buscaremos un servidor con pocas funciones adicionales, sencillo, y que ocupe poco (cuando más pequeño sea, menos fallos puede tener en el código). El que más se adapta a estas condiciones es PureFTPd.

Probamos a hacer `apt-get install pureftpd` (y con `pure-ftpd`), pero ninguno funciona. La razón es que PureFTPd no viene incluido en esta versión de Debian. Lo podemos comprobar con `apt-cache search pure`: veremos que no hay ningún paquete en el que salga “pure”. La forma de instalar el programa es, entonces, ir a la página web (<http://www.pureftpd.org>) y a la sección de Downloads y bajar en formato `.tar.gz` la última versión. Lo que hemos bajado son las fuentes (en inglés, “sources”) del programa, en lenguaje C, por tanto, las tendremos que compilar para crear el ejecutable que funcione en nuestra máquina. Nos hará falta como mínimo el `gcc` (GNU C Compiler) y un poco de tiempo (depende del ordenador, pero es casi seguro que menos de una hora).

Como root, descomprimiremos el archivo en algún sitio (por ejemplo `/usr/src`) con:

```
tar zxvf archivo.tar.gz -C /usr/src
```

Entonces podemos entrar en el directorio creado y leer los ficheros `README` e `INSTALL`, que normalmente están en todos los programas que hay que compilar. Si vemos que no hay que hacer nada especial -como es el caso- seguiremos el procedimiento habitual para compilar:

1. Ejecutar `./configure` para generar un script de compilación adecuado para nuestra máquina en concreto. Si falta algún programa o librería, parará y lo mostrará de forma clara. Podemos desactivar o activar muchas opciones (ver `./configure --help`). En nuestro caso, hemos comprobado que al compilar sin ninguna opción especial, el binario resultante no aceptaba la opción `-n`, que sirve para limitar el espacio de cada usuario. Por lo tanto, es buena idea activar directamente la opción

adecuada con `./configure --with-quotas`.

2. Ejecutar `make`, la verdadera compilación. Este proceso -completamente automatizado- puede tardar segundos, minutos o horas dependiendo del programa y del ordenador. No tiene que fallar; si lo hace será más complicado solucionar el problema ya que probablemente se encuentre dentro del código fuente de algún fichero del proyecto (o sea, que no es culpa nuestra). Eso sí, es probable que aparezcan mensajes de “warning” (advertencias), que podemos ignorar a menos que puedan provocar algún problema grave.
3. Este paso es opcional. `make check` hará unas comprobaciones para ver si se ha compilado bien. Si alguna prueba falla no hay que continuar.
4. `make install` copiará los ejecutables (el resultado de la compilación) a los directorios donde se encuentran todos los demás programas de Linux. Hacen falta privilegios de root.

Hecho esto ya hemos conseguido crear el fichero ejecutable e integrarlo en el sistema; por tanto podemos borrar el directorio que hemos creado en `/usr/src` (a menos que queramos hacer algún cambio al código y volverlo a compilar).

Ahora, cada vez que encendamos el ordenador tendremos que ejecutar `pure-ftpd` (se puede dejar en segundo plano) y el servicio estará activo, con los usuarios actuales del sistema. Para evitar repetir esta acción cada vez, podemos crear un script que arranque el servicio con las opciones más apropiadas (que habremos consultado en la ayuda con `pure-ftpd --help`):

```
/usr/local/sbin/pure-ftpd -B -A -u 100 -C 5 -n 1000:800  
echo "Arrancando el servidor pure-ftpd..."
```

Descripción de las opciones:

- El `-B` es para que se ejecute en segundo plano
- El `-A` hace que todos los usuarios estén en un entorno `chroot()`, o sea, que ven su directorio personal como `/` y no pueden salir de ahí. Por tanto, están 'encerrados' dentro de su directorio personal. Es teóricamente imposible salir de esta jaula, aunque en la práctica se puede hacer (aunque en la del FTP costaría mucho más). No obstante, aunque un usuario consiguiera acceder al exterior no podría estropear muchas cosas porque no sería el propietario. Estaría en la misma situación que un usuario del sistema sin privilegios.
- El `-u 100` es el UID mínimo que hace falta para conectarse. Por lo tanto, los usuarios que tengan un UID menos (que son el administrador y los usuarios especiales y de sistema) no podrán conectarse. Esto es bueno para la seguridad general del ordenador (recordemos que la contraseña de FTP viaja sin encriptar).

- `-C 5` hace que sólo se puedan hacer 5 conexiones por host. Es poco probable que un usuario con buenas intenciones necesite hacer más de una a la vez, pero tampoco hay que limitarlo a una, así que dejaremos cinco.
- El `-n 1000:100` limita cada usuario a 1000 archivos o 800 Mb, más que suficiente para traspasar ficheros grandes. En algunas versiones esta opción requiere compilar con la opción de cuotas activada. Si sólo hemos puesto `./configure` y vemos que después no acepta la opción `-n`, haremos `./configure --with-quotas` y después el `make` y el `make install`.

Grabaremos este script en `/etc/init.d/pure-ftpd`, le daremos permisos de ejecución a todos con `chmod a+x /etc/init.d/pure-ftpd`, y crearemos enlaces dentro de los niveles de ejecución en los que nos interese que corra este proceso. Recordamos los niveles de ejecución (“runlevels”):

- 0 --> Halt (parar el sistema y apagarlo)
- 1 --> Modo monousuario
- 2 --> Modo multiusuario sin soporte de red
- 3 --> Modo multiusuario completo
- 4 --> (Sin usar)
- 5 --> Modo multiusuario completo con login gráfico
- 6 --> Reboot (reiniciar el sistema)

Por lo tanto, los niveles en los que nos interesa que actúe nuestro script son el 1, 2, 3 y 5. Crearemos los enlaces simbólicos con:

```
ln -s /etc/init.d/pure-ftpd /etc/rc1.d/S80pure-ftpd
ln -s /etc/init.d/pure-ftpd /etc/rc2.d/S80pure-ftpd
ln -s /etc/init.d/pure-ftpd /etc/rc3.d/S80pure-ftpd
ln -s /etc/init.d/pure-ftpd /etc/rc5.d/S80pure-ftpd
```

El prefijo S80 tiene su explicación: S significa que es un script 'Startup', que sirve para iniciar algo. En caso contrario llevaría una K ('Kill'). El número indica el orden en el que se ejecutarán los scripts de un mismo directorio rc. Hay que asegurarse de que se inicie en un buen momento, cuando la red ya está preparada y las operaciones importantes ya han acabado; por tanto, 80 ya es un buen número.

Sería correcto añadir en los runlevel 0 y 6 un script que matase al servidor, pero no hace falta porque el propio proceso de apagado ya se encarga de matar todos

los procesos.

Ya sólo queda reiniciar (o hacer un `init 3`) y probar a hacer un FTP desde fuera. Hay que recordar que en el router los puertos 20 y 21 de TCP tienen que estar abiertos (el 20 es para la transmisión de datos).

También es el momento de probar a entrar como usuario `web` desde cualquier cliente de FTP (o con un navegador, con `ftp://web@IP_DEL_SERVIDOR`), poner la contraseña y comprobar que podemos subir y bajar archivos al servidor web.

NOTA: hay que evitar usar cuentas de usuario importantes en el FTP, porque los datos viajan sin encriptar, y un usuario que tuviese privilegios de root podría interceptar los paquetes y ver las contraseñas. Para evitarlo podemos crear cuentas adicionales sólo para el FTP, o usar el programa `sftp` que viene incluido en el [SSH](#).

2.3.4 Configuración del proxy

En cualquier red los usuarios acceden a páginas web prohibidas, ya sea porque no están relacionadas con el trabajo que hay que hacer con el ordenador, o porque son pornográficas o de contenidos ilegales.

Una forma de denegar el acceso a ciertas páginas es configurar cada navegador web de forma que pida contraseña cuando se entra en unas direcciones web o IPs preestablecidas. También podemos utilizar software de filtrado de contenidos que analice las palabras o incluso las imágenes.

El problema que presentan estos métodos (aparte del elevado precio) es que requieren manipular cada uno de los ordenadores de la red individualmente, cuando lo ideal sería tener que crear la 'lista negra' sólo en un ordenador. Es aquí donde intervienen los proxy: un proxy es un ordenador que recoge todas las peticiones de la red, busca las páginas correspondientes en Internet y devuelve cada página al ordenador que la ha pedido. De esta manera, es sólo este ordenador el que se conecta a Internet, y por lo tanto es fácil prohibir o admitir IPs.

Puede parecer que el proxy hace más lenta la navegación ya que un sólo ordenador tiene que hacer el trabajo de muchos, pero de hecho el proxy también está diseñado para agilizarla actuando como caché. 'Caché' hace referencia a una zona de la memoria o del disco, que almacena información durante un tiempo y que va vaciándose a medida que llegan nuevos datos para llenarla.

Por lo tanto, un proxy-caché funciona así:

1. Recibe una petición HTTP de un ordenador.
2. Consulta en su caché buscando la página pedida.
3. Si no tiene la página en la caché, la baja de Internet y queda grabada en la caché. En cambio, si ya la tenía no es necesario que se vuelva a conectar a Internet.
4. Devuelve la página de la caché al ordenador que la había pedido.

Periódicamente hace comprobaciones para evitar que la caché se llene, borrando los datos más antiguos o los menos pedidos. Además, registra los accesos en los ficheros de log, como casi todos los demás servidores que hemos visto.

En conclusión, un proxy-caché ayuda a reducir mucho el ancho de banda usado en una red (debido principalmente a las consultas web). No es ningún problema la velocidad de transmisión; será la adecuada porque disponemos de tarjetas de red 10/100 Mbits/s para enviar datos de un ordenador a otro (por la Intranet), mientras

que la conexión a Internet llega como mucho a los 2 Mbits/s si es una línea ADSL. Además, hay que recordar que casi todos los navegadores actuales tienen su propia caché web local, y por tanto se evitarán peticiones web a Internet, o en este caso al ordenador proxy.

El proxy-caché para Linux más conocido y usado es Squid. Tiene soporte para HTTP, FTP, SSL, SNMP, DNS, control de acceso (ACL), jerarquías de proxies (protocolo ICP), y muchas opciones más. Su web es <http://www.squid-cache.org>

Una de las características más interesantes de Squid es que puede actuar como proxy transparente: funciona de manera parecida a la explicada anteriormente, pero el cliente (el ordenador que pide páginas web) no se entera de que está pasando por un proxy ya que éste último actúa como si realmente fuera el servidor web de Internet. El inconveniente de este método es que el proxy tiene que actuar como servidor web en el puerto 80, y nosotros ya tenemos un servidor Apache en este puerto; por lo tanto el proxy que pondremos será estándar.

Haremos un `apt-get install squid` para bajarlo e instalarlo, pero tenemos que modificar la configuración por defecto. Hay que cambiar, por ejemplo, el puerto usado (ahora es el 3128, pero en Europa se usa más el 8080). Antes de modificar el fichero de configuración (`/etc/squid.conf`), comprobamos que `squid` no esté funcionando con privilegios de root (eso sería un problema de seguridad):

```
bruguers:~# ps axu | grep squid
root      336  0.0  1.1  3824 1124 ?        S    12:38   0:00 /
usr/sbin/squid -D -sYC
proxy     338  0.9  5.5  8556 5308 ?        S    12:38   0:19 (squid)
-D -sYC
root      374  0.0  0.7  1748   716 pts/0    S    13:11   0:00 grep
squid
bruguers:~#
```

Podemos ver que, aunque lo llame root, el proceso está con el conocido como “setuid proxy”, o sea, que se ejecuta con los privilegios del usuario “`proxy`”. Es mejor que esté así porque en versiones anteriores se ejecutaba como root, y eso comportaba mucho más peligro delante de cualquier bug que se encontrase, ya que un usuario que usase un exploit conseguiría una shell de root. Con la configuración actual, si un usuario aprovecha un bug (por ejemplo un buffer overflow, los más comunes) podría acceder al sistema como si fuera un usuario normal.

Ahora vamos a por la configuración del proxy Squid, que se encuentra en `/etc/squid.conf`. Tendremos que cambiar los siguientes parámetros:

```
http_port 3128
http_port 8080
```

Este es el puerto usado por el proxy (por defecto 3128). Hemos añadido el 8080, pero tampoco es necesario quitar el 3128; Squid puede recibir peticiones por ambos puertos. Lo dejamos por compatibilidad por otros programas.

Además, si hacemos escaneos de puertos cuando todo funcione, veremos que también usa el 3130 de UDP.

```
cache_peer proxy.xtec.es parent 8080 0 no-query no-digest
default
```

Esto hace que nuestro proxy tenga un 'padre' y trabaje en jerarquía con él: todo lo que los usuarios de la red pidan al servidor se buscará en su caché, y si no se encuentra, la petición pasará al proxy padre. Si este padre tampoco lo encuentra en su caché, lo buscará en Internet, por tanto evitamos todo lo posible las peticiones innecesarias.

El instituto usa el proxy proxy.xtec.es por el puerto 8080 (el habitual de los proxy-cachés). Hemos desactivado las opciones ICP (protocolo usado por los proxys) poniendo los parámetros 0 como puerto ICP y `no-query`. El `no-digest` lo ponemos porque el fichero de digests es innecesario cuando hay un sólo padre, y el `default` hace que el servidor use por defecto este padre para encontrar todo lo que no tenga en la caché.

Obviamente, si no disponemos de proxy padre (o sea, si queremos tener Internet como padre) no hay que poner esta línea.

```
# cache_mem 8 MB
```

Dejamos la opción por defecto de 8 Mb. para la memoria RAM usada como caché. Se recomienda dejar una cuarta parte de la RAM total del ordenador, pero como nuestro servidor tiene poca (48 Mb) y además se tiene que encargar de muchas más cosas, dejaremos este valor, que de todas maneras es lo bastante grande como para grabar los objetos más pedidos (entendiendo objeto como una imagen, una página, un vídeo, o cualquier otro elemento web). Hay que tener en cuenta que aparte de la RAM usada como caché, el proceso de Squid necesita más RAM para poder ejecutarse (como cualquier otro proceso), por lo tanto no hay que poner valores muy grandes.

```
cache_dir ufs /var/spool/squid 150 16 256
```

Es aquí donde decimos a Squid que use el directorio `/var/spool/squid` como caché. `ufs` es el sistema de archivos de Squid. En los parámetros le indicamos que puede usar 150 Mb. y que organice los objetos creando 16 directorios, cada uno con 256 subdirectorios y con los ficheros dentro.

Hemos asignado 150 Mb. porque el disco duro del ordenador puede llegar a ser más lento que Internet si hay que buscar un archivo entre miles de ellos, teniendo en cuenta también que 150 Mb. son suficientes para guardar casi todas las páginas que se usarán a menudo en el instituto.

Un detalle: tenemos que asegurarnos de que el propietario del directorio usado como caché sea el mismo que ejecuta el proceso `squid`, sinó no podrá acceder. Es

poco probable que no sea así ya que se ha creado durante la instalación.

```
# ftp_passive on
```

Dependiendo del router detrás del cual estemos, habrá que hacer que Squid se conecte a los FTPs en modo pasivo o no. La mejor manera de saberlo es probar la configuración por defecto y si no funciona poner la otra opción.

En el mismo fichero hay que definir las listas de control de acceso (ACL). Aún no prohíben nada; sólo sirven para definir unos grupos de usuarios. Añadiremos las siguientes:

Definimos los ordenadores con la IP 192.168.0.x (la red local del instituto). Después haremos que sean sólo éstos los que tengan acceso al proxy:

```
acl red src 192.168.0.0/255.255.255.0
```

Añadiremos más listas de control de acceso para prohibir ciertas páginas. Hacen falta tres ficheros, que crearemos con cualquier editor y grabaremos en `/etc/squid` (si no existe el directorio lo creamos con `mkdir /etc/squid`). El nombre de cada archivo es de libre elección:

- `/etc/squid/dom_no.txt`: aquí escribiremos los dominios o hosts de Internet a los que no se podrá acceder, uno en cada línea. Por ejemplo, podemos poner `playboy.com`, `penthouselive.com`, `mercabanner.com`, `doubleclick.com`, `doubleclick.net`, `linkexchange.com`, `tradedoubler.com`, `realmedia.com`. Estos últimos son páginas que se dedican al intercambio de banners; por tanto si los prohibimos ya no veremos más.

Nota: en vez de poner decenas de dominios como 'cibersexo.com', 'supersexo.com', 'sexogratis.com', 'quierosexo.com', a continuación crearemos una regla para prohibir todos los dominios que contengan la palabra 'sex'.

- `/etc/squid/url_no.txt`: pondremos palabras (una por línea) que las URLs no puedan tener. Por ejemplo, `sex`, `porno`. Eso nos ahorrará mucho trabajo.
- `/etc/squid/url_yes.txt`: aquí pondremos las excepciones a la regla anterior: palabras que, aún teniendo la palabra prohibida, son aceptadas. Por ejemplo: `sexuali` (para que se acepten “sexualidad” y “sexualitat”, en catalán).

Definimos estas tres listas. Hemos puesto el `-i` en cada opción para que la lista de palabras sea 'case insensitive', o sea, que no tenga en cuenta las mayúsculas y minúsculas:

```
# Dominios prohibidos:
```

```
acl dom_no dstdom_regex -i "/etc/squid/dom_no.txt"  
# Palabras prohibidas:  
acl url_no url_regex -i "/etc/squid/url_no.txt"  
# Palabras autorizadas:  
acl url_yes url_regex -i "/etc/squid/url_yes.txt"
```

Ya hemos definido cuatro listas (la de IPs y estas tres). Ahora tenemos que especificar qué privilegios tienen estos grupos de usuarios:

```
http_access deny !red  
http_access deny dom_no  
http_access allow url_yes  
http_access deny url_no  
http_access allow all
```

Para entender esto hay que tener claro lo siguiente: deny = 'denegar', allow = 'permitir', !condición = "no-condición" (NOT lógico).

Es muy importante que estas líneas estén en un orden lógico ya que Squid hace las comprobaciones comenzando por el principio, y cuando una de las condiciones se cumple, no sigue leyendo y hace la acción asociada ([allow](#) o [deny](#)). Tal como las hemos puesto, los filtros por los que pasa una petición que llega al proxy son:

1. ¿No es de la red local? Pues si no es de la red local, deniego el acceso; si es de la red, continuo comprobando.
2. ¿Quiere entrar a un dominio prohibido? Si es así, deniego el acceso; sinó, sigo comprobando.
3. ¿Contiene la URL una palabra permitida? (Una palabra que parece prohibida pero que el administrador ha considerado que puede ser utilizada). Si la tiene, no sigo comprobando y dejo pasar la petición directamente. Si no la tiene, sigo comprobando.
4. ¿Contiene la URL una palabra prohibida? Si la tiene, deniego el acceso.
5. Ya ha pasado por todos los filtros: es de la red local y no contiene palabras prohibidas ni quiere acceder a un dominio prohibido. Permito el acceso al proxy.

Las ACL (listas de control de acceso) ya están definidas. Continuamos con la configuración restante:

```
cache_mgr instituto@dominio.com
```

Si hay algún problema con la caché del proxy se mostrará un mensaje de error con esta dirección de e-mail. Por defecto pone “webmaster”, lo podemos dejar así si no queremos mostrar nuestra dirección de correo.

Si también queremos que Squid actúe como acelerador de `httpd` (demonio HTTP), tendremos que añadir estas líneas. Así conseguiremos que la caché también se use en las peticiones web que se hagan a nuestro servidor Apache, y agilizará más su funcionamiento.

```
httpd_accel_host virtual
httpd_accel_port 0
httpd_accel_with_proxy on
```

Por último, irá muy bien cambiar el idioma de Squid a español sobre todo porque los usuarios de la red comprenderán mejor los mensajes de error que puedan salir. Lo hacemos con:

```
error_directory /usr/lib/squid/errors/Spanish
```

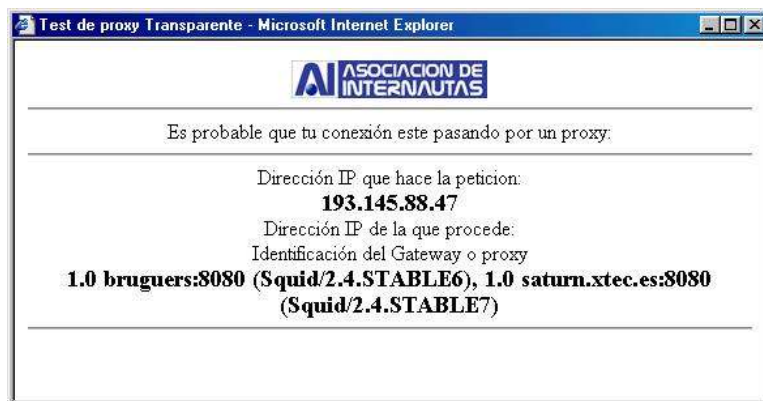
Ya hemos acabado con el fichero de configuración. No hace falta reiniciar el proceso `squid`; sólo hay que ejecutar la siguiente orden para que lea la nueva configuración:

```
squid -k reconfigure
```

Ahora queda la mejor parte: probarlo y comprobar que funciona todo a la primera. Sólo tenemos que ir a otro ordenador de la red, abrir el navegador (sea cual sea) y poner en la configuración de 'Proxy' la IP del servidor y el puerto 8080. Podemos crear un host en cada ordenador para no tener que escribir la IP; de todas maneras dejaremos este paso para más adelante porque cuando tengamos servidor de DNS, el nombre de host se traducirá directamente a la IP del servidor (como si fuera una dirección de Internet).

Para probarlo del todo tendremos que entrar en páginas web, en FTPs, en sitios que requieran conexiones seguras (SSL); entraremos en páginas lentas para ver si tarda menos en volver a entrar, entraremos a páginas inexistentes para ver los mensajes de error, entraremos desde IPs no permitidas, etc. En nuestro caso hemos comprobado una mejora de la velocidad apreciable en todos los sentidos: las páginas web cargan antes, los FTPs van más rápido, los errores que tienen que salir salen antes, y nada falla.

Ahora que ya funciona, tendremos que ver cómo evoluciona su estado con el paso del tiempo. Los ficheros de log irán creciendo, pero no nos tenemos que preocupar porque en `/etc/cron.daily/` (listado de las tareas que se ejecutan cada día) hay un script llamado `logrotate`, que está configurado en `/etc/logrotate.conf` para rotar los logs de, entre otros, el Squid (en `/etc/logrotate.d/squid`).



Comprobando que funciona el proxy y el padre de la xtec

'Rotar los logs' se refiere a comprimir el fichero de logs (de accesos registrados), cambiarle el nombre, y volver a crear uno pero vacío. Por tanto, cuando veamos ficheros como `/var/log/squid/access.log.7.gz`, sabremos que se trata de un log bastante antiguo (más antiguo cuanto más grande sea el número), y también sabremos dónde encontrar el log de ese mismo momento: en su sitio, `/var/log/squid/access.log` (sin comprimir ni cambiar de nombre). La operación de rotación de logs de todos los servidores la controla `logrotate`.

Para acabar, hay que tener en cuenta que si queremos que los usuarios no puedan acceder a Internet por otro medio que no sea el proxy, se puede crear un filtro en el router, o se puede desconfigurar cada ordenador cliente para que no funcionen los DNS del proveedor de Internet contratado, ya que es el proxy también quien se encarga de resolver nombres de host. Esto lo podremos perfeccionar cuando hagamos el servidor de DNS propio.

2.4 Servicios secundarios

2.4.1 Soporte de PHP en la web

Podremos aprovechar mucho más el servidor si usamos páginas web que incluyan scripts en PHP. Además, puede ser una práctica muy útil para los alumnos que han acabado de estudiar HTML y quieren orientarse a la programación de webs de comercio electrónico y similares.

El PHP es un lenguaje muy potente que se ejecuta en el servidor cuando alguien pide una página, y que genera un código HTML que incluye en la página de retorno. Para ver qué nos proporciona este lenguaje, haremos un ejemplo que después nos servirá para probar si se ha instalado correctamente en el servidor.

Supongamos que subimos al servidor un archivo llamado `prueba.php` con el siguiente contenido:

```
<BODY BGCOLOR=YELLOW>
<BR><BR>Esto es <B>HTML</B> normal.<BR>
<?
echo "Aquí comienza el código PHP<br>";
for ($size=1; $size<=7; $size++) {
    echo "<font size=$size color=#".(2+$size)."90000>";
    echo "Tamaño $size</font><br>\n";
}
?> <BR> Esto vuelve a ser HTML; el PHP ya se ha acabado.
</BODY>
```

Si cargásemos este fichero en el navegador (en la dirección http://IP_DEL_SERVIDOR/prueba.php), el código que recibiríamos sería:

```
<body BGCOLOR=YELLOW>
<br><br>Esto es <b>HTML</b> normal.<br>
Aquí comienza el código PHP<br><font size=1 color=#390000>Tamaño
1</font><br>
<font size=2 color=#490000>Tamaño 2</font><br>
<font size=3 color=#590000>Tamaño 3</font><br>
<font size=4 color=#690000>Tamaño 4</font><br>
<font size=5 color=#790000>Tamaño 5</font><br>
<font size=6 color=#890000>Tamaño 6</font><br>
<font size=7 color=#990000>Tamaño 7</font><br>
<br> Esto vuelve a ser HTML; el PHP ya se ha acabado.
</body>
```

Hay que destacar que nada del código PHP que había escrito lo podrá ver nunca el usuario (aunque intente descargar el `.php`), ya que el servidor genera y envía sólo código HTML.

Por tanto, hemos hecho un código que automatiza una operación mediante un bucle `for`; en este caso lo que hace es mostrar diferentes textos de tamaños cada vez más grandes, y de distinto color. Pero las posibilidades son inagotables: se pueden crear dinámicamente páginas que pide el usuario, se pueden usar cookies, contraseñas, sesiones que expiran al cabo de un tiempo, contadores, se puede acceder a bases de datos con el motor MySQL, y muchas cosas más. Hay muchas páginas importantes hechas con lenguaje PHP, como los 'blog' ('web log', redes de páginas interrelacionadas mediante enlaces), o portales como Barrapunto (www.barrapunto.com) y similares. También hay herramientas especiales, como PHPNuke, que don las bases para crear uno de estos portales sin tener que aprender a programar en PHP.

Para nuestro servidor, nos ocuparemos de instalar PHP, integrarlo en Apache, y comprobar que funcione con el ejemplo anterior (el de las fuentes). La instalación de MySQL, su configuración, y el proceso para instalar utilidades como PHPNuke no lo describiremos aquí, pero no es muy difícil: sólo hay que hacerlo como con los demás programas (usando `apt-get`) y seguir las indicaciones de los ficheros `INSTALL` y/o `README`.

Un `apt-cache search php` nos revela que el paquete más apropiado se llama `php4`; por lo tanto hacemos un `apt-get install php4` y se bajará junto con sus dependencias. Nos saldrá un mensaje parecido a éste:

```
I see you have apache webserver installed and so far you haven't used
the apache module version of php4 in your apache. If you want to use it,
you should reconfigure the apache webserver and select to load the php
module. I can call the apacheconfig script now for you to do it, or you
can insert the following line into /etc/apache/httpd.conf manually:
LoadModule php4_module /usr/lib/apache/1.3/libphp4.so
Do you want me to run the apacheconfig script now [y/N] ?
```

Esto quiere decir que ha detectado la configuración de Apache y que se nos ofrece para integrarse como un módulo automáticamente. Como nos fiamos, decimos que sí (`y`) y continuamos con la configuración (nos hará reiniciar el Apache). Cuando acabe, probamos si funciona: ponemos en `/var/www` el fichero `prueba.php` comentado anteriormente e intentamos abrirlo con el navegador. No funciona; en vez de mostrar nada cree que nos queremos bajar el archivo.

Tendremos que ir a la web www.php.net, donde se encuentra toda la información sobre el lenguaje, y leer las instrucciones de instalación. En la web también encontraremos muchas notas de los usuarios sobre todos y cada uno de los temas de la documentación, por tanto es poco probable que no podamos resolver los problemas que nos salgan.

En este caso, vemos en la web que hay que añadir unas líneas al `/`

`etc/apache/httpd.conf` para que entienda el formato PHP. Vamos al fichero y encontramos estas líneas, comentadas:

```
#AddType application/x-httpd-php .php
#AddType application/x-httpd-php-source .phps
```

Sólo hay que descomentarlas (quitando el `#` del principio) para añadir el tipo de archivo PHP. También buscamos la línea que ha añadido el `apt-get` y le quitamos el `#` si lo tiene. Ha de quedar algo parecido a:

```
LoadModule php4_module /usr/lib/apache/1.3/libphp4.so
```

También tenemos que decirle a Apache que la página a cargar por defecto si no se pone nada ya no es sólo `index.htm`, sino que también sirve `index.php`. Buscamos un fragmento donde pone:

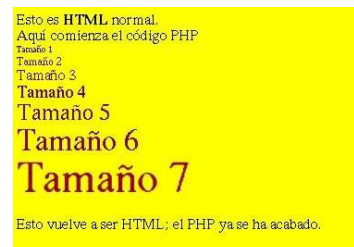
```
<IfModule mod_dir.c>
    DirectoryIndex index.html index.htm index.shtml index.cgi
</IfModule>
```

Y añadimos `index.php` al final, de forma que queda:

```
<IfModule mod_dir.c>
    DirectoryIndex index.html index.htm index.shtml index.cgi index.php
</IfModule>
```

Así, intentará cargar primero `index.html`, si no lo encuentra `index.htm`, después `index.shtml` y así sucesivamente. Podemos ponerlo en el orden que queramos, aunque no debería haber más de un fichero `index` en el directorio.

Ahora reiniciamos Apache (`apachectl restart`) y volvemos a cargar la página PHP en el navegador. Esta vez tiene que funcionar, y veremos el código HTML mencionado antes.



Página PHP de prueba

Como práctica para los alumnos, se puede aprovechar el PHP para hacer un contador de visitas para la web, de forma que grabe en un archivo los datos de los usuarios que entran (IP, referer, navegador, resolución, etc.).

Hay que remarcar que el PHP es uno de los lenguajes mejor documentados de Internet; encontraremos muchos scripts ya hechos, tutoriales, foros y grupos de usuarios; sólo hace falta buscar.

2.4.2 Soporte de CGIs en la web

CGI hace referencia a Common Gateway Interface, y es un estándar para poder ejecutar scripts (pequeños programas) en las páginas web. Es muy parecido a PHP, con la diferencia de que un CGI puede estar escrito en muchos lenguajes de programación, entre ellos:

- Perl
- Shells de Linux
- C/C++
- Fortran
- TCL
- Visual Basic
- AppleScript

Sin embargo, la mayoría están escritos en Perl, y gran parte de los restantes son scripts para la shell de Linux. ¿Y por qué Perl? Hay muchas razones que hacen de Perl el lenguaje más apropiado:

- Es muy portable (multiplataforma).
- Tiene muchos operadores relacionados con las cadenas de texto, y también con los datos en formato binario. Por tanto, puede trabajar bien con la información.
- Es simple y preciso, aunque según como puede parecer algo “críptico” debido a la gran cantidad de símbolos.
- Puede llamar a comandos del shell directamente.
- Hay muchas extensiones para Perl que amplían sus posibilidades y hacen que pueda incluso acceder a muchos formatos de bases de datos.

Veremos un programa CGI más adelante, después de haber preparado el soporte en Apache. No hay que instalar paquetes nuevos; sólo hay que comprobar que tengamos instalado el Perl con `perl --version` y después editaremos el fichero de configuración del Apache `/etc/apache/httpd.conf`:

Vemos que la línea `LoadModule cgi_module /usr/lib/apache/1.3/mod_cgi.so` está descomentada. Perfecto. También leemos cerca de la línea `ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/` que los CGIs no están en `/var/www/cgi-bin/`, sino que esta dirección está redirigida a `/usr/lib/cgi-bin/`, probablemente para evitar que en el directorio web haya programas ejecutables, dejando de esta forma sólo los elementos web. Más adelante encontramos las siguientes líneas comentadas:

```
# To use CGI scripts:
#AddHandler cgi-script .cgi .sh .pl
```

Hay que quitar el signo de comentario en la línea de `AddHandler` para hacer que entienda los `.cgi` como programas y no como ficheros de texto normales.

Grabamos, y reiniciamos el Apache con `apachectl restart`

Para probar si funciona, creamos un fichero con el siguiente contenido:

```
#!/usr/bin/perl
print "Content-type: text/plain","\n\n";
print "; Este script CGI hecho en Perl funciona !", "\n";
$uptime = `/usr/bin/uptime` ;
($load_average) = ($uptime =~ /average: ([^,]*)/);
print "Carga de la CPU: ", $load_average, ".", "\n";
print "Actualiza la página y verás cómo cambia.", "\n";
exit (0);
```

Hemos de guardar este CGI en `/usr/lib/cgi-bin/primer.cgi` (¡no en `/var/www/cgi-bin/primer.cgi`!, ya que Apache tiene el `alias` que hemos comentado antes), y hemos de darle permiso de ejecución con `chmod a+x /usr/lib/cgi-bin/*` (este es un paso muy importante; si no lo hacemos saldrán errores en la web). Cuando esté hecho, abrimos un navegador y entramos en `http://IP_DEL_SERVIDOR/cgi-bin/primer.cgi`. Y ahora sí que nos tenemos que referir al CGI como si realmente estuviera en el directorio `/var/www/cgi-bin`; el `alias` sabrá dónde encontrar el verdadero archivo.

Si todo funciona bien, veremos el mensaje en formato texto (sin HTML ni otros códigos por en medio). Si no funcionase, habrá que comprobar si podemos ejecutar el script localmente (`perl /usr/lib/cgi-bin/primer.cgi`), si tiene permiso de ejecución, si existen los módulos de Perl necesarios, si nos hemos dejado algo relacionado con `cgi` en la configuración del Apache, y si lo hemos reiniciado. En todo caso, podemos consultar `/var/log/apache/error.log` para ver los mensajes de error detallados.

Con los CGI podemos crear herramientas muy potentes que interactúen con el sistema y que nos permitan configurarlo remotamente desde la misma página web. De hecho, programas de este tipo ya los hay; el más usado es Webmin (<http://www.webmin.com>).

Pero también es importante saber que muchas técnicas para acceder a servidores web y sacar información privada aprovechan CGIs poco usados o mal programados. Aunque casi todos estos errores afectan a otros sistemas operativos, conviene no abusar de los CGI.

2.4.3 Estadísticas de acceso a la web

Para ver cómo va nuestra página -uno de los servicios más importantes del ordenador- no podemos estar revisando los logs del Apache continuamente, ya que la información que muestran es muy extensa y detallada, probablemente más de lo que queremos.

Aprovecharemos que hemos añadido el soporte para ejecutar programas CGI y pondremos un analizador de ficheros de registro de accesos. Lo único que hacen es interpretar el fichero `/var/log/apache/access.log` y generar una página HTML con estadísticas y gráficas sobre los documentos más pedidos, la procedencia de los visitantes y las horas en las que entran. Así es muchas más fácil saber si todo funciona según lo previsto.

De analizadores de logs también hay muchos (gratuitos y comerciales); se diferencian sobre todo en la velocidad con la que trabajan con los logs, que pueden ocupar muchos gigabytes. El que usaremos nosotros es AWStats, uno gratuito, muy rápido, configurable y de uso sencillo. Está escrito en Perl, y tiene soporte para muchos idiomas, entre ellos el español y el catalán, por tanto es apto para que también los visitantes puedan ver y entender las estadísticas.

Un `apt-cache search awstats` nos revela que Debian incorpora este paquete en su distribución de Woody, pero un `apt-cache show awstats` nos muestra que es la versión 4.0 Como en su web <http://awstats.sourceforge.net> dice que la versión actual es -en el momento de escribir esto- la 5.3, vale la pena bajar el programa de la web e instalarlo nosotros. Podemos consultar el ChangeLog para ver que ha habido muchos cambios importantes de la versión 4.0 a la 5.3.

Si lo hemos instalado con `apt-get` y hemos descubierto la nueva versión después, podemos desinstalar el programa tan fácilmente como lo hemos instalado:
`apt-get remove awstats`

Por tanto, hay que ir a la web <http://awstats.sourceforge.net> y en la sección de 'Download' bajar el `.tar.gz` (o `.tgz`) que hay en el apartado 'Last Stable'. Lo descomprimos y seguimos las indicaciones de la misma web (en el fichero que hemos bajado también están, pero puede que no estén actualizadas del todo):

- Comprobamos que el fichero de logs de Apache esté en formato 'combinado'. Si lo miramos, en `/var/log/apache/access.log`, tiene que haber líneas como:

```
192.168.0.2 - - [04/Jan/2003:19:44:36 +0100] "GET / HTTP/1.1" 304 - "-"  
"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.3b) Gecko/20030103"
```

- Entramos en el directorio `wwwroot` que se ha creado al descomprimir el fichero, y movemos todo el contenido de la carpeta `cgi-bin`

(`awstats.model.conf`, `awstats.pl` y los directorios `lang`, `lib` y `plugins`) al directorio de `cgi-bin` de Apache, que es `/usr/lib/cgi-bin`

- Movemos el directorio `icon` a `/var/www/icon`
- En `/usr/lib/cgi-bin/`, copiamos el `awstats.model.conf` al mismo directorio, con el nombre `awstats.bruguers.conf`
- Editamos este nuevo fichero, y nos fijamos en lo siguiente:
 - `LogFile` ha de apuntar al fichero de logs `/var/log/apache/access.log`
 - `DirIcons` ha de tener `/icon`, porque es la ruta (relativa a `/var/www`) donde hemos copiado el directorio `icon`
 - En `SiteDomain` pondremos `"bruguers"` (el nombre de host del servidor).
 - `PurgeLogFile=0` porque de los logs ya se ocupa el `logrotate`, no hace falta que sea `awstats` quien los borre periódicamente.
 - `Lang="es"` o `Lang="es_cat"` para poner toda la página en español o catalán.
 - `ShowFlagLinks="en es es_cat"` para ver en la página las banderas de cambiar el idioma.
- Actualizaremos las estadísticas por primera vez (o sea, que las crearemos) con el comando `./awstats.pl -config=bruguers -update`. Nos informará de cuántos registros tenía, cuántos ha añadido, cuántos ha descartado por no ser peticiones HTTP, y cuántos son erróneos.
- Programaremos el sistema mediante `crontab` para hacer que las estadísticas se actualicen cada día. Aprovecharemos el `cron.daily` que hay en `/etc/crontab`. Creamos un script en `/etc/cron.daily/awstats` con el siguiente contenido (el comando es el que acabamos de ejecutar):

```
#!/bin/sh
/usr/lib/cgi-bin/awstats.pl -config=bruguers -update
```

- Le daremos permiso de ejecución con `chmod a+x /etc/cron.daily/awstats`, y ya habremos hecho que cada día, a la hora especificada en `/etc/crontab` (que cambiamos después de la instalación de Debian), se actualicen las estadísticas.

Ya está todo preparado; sólo queda probarlo entrando en la página `http://IP_DEL_SERVIDOR/cgi-bin/awstats.pl`, donde veremos unas completas estadísticas sobre las visitas a nuestro servidor. Podemos quitar información, si queremos, editando el fichero de configuración anterior `/usr/lib/cgi-bin/awstats.bruguers.conf`

2.4.4 Conexiones por Secure Shell

Queremos que los usuarios normales y los administradores se puedan conectar al PC remotamente y utilizarlo como si estuvieran delante: que puedan abrir una consola, identificarse con el nombre y contraseña, y tener acceso a todos los comandos, programas y ficheros que les pertenecen. Si un usuario con más privilegios se conecta, puede hacerlo absolutamente todo: crear nuevos usuarios, desconectar a otros, apagar el ordenador, poner música o incluso abrir y cerrar el CD.

La forma tradicional de hacer esto es montando un servidor de Telnet, y después conectando desde otra máquina con `telnet IP`. Este sistema sólo tiene un inconveniente: toda la información viaja por la red en formato texto, o sea, sin encriptar. Por tanto, cualquier persona que pueda interceptar paquetes (hacen falta privilegios de root) podría ver todo lo que envía y recibe cada uno de los usuarios que hay conectados, incluso las contraseñas.

La solución a este peligroso sistema consiste en utilizar el protocolo SSH en vez de Telnet. SSH (siglas de Secure Shell) es un tipo de conexión idéntica a Telnet, pero encriptada con claves RSA. Además, tiene muchos otros sistemas de seguridad que evitan ataques que pueden hacerse con Telnet, como por ejemplo el IP spoofing. Usa el puerto 22 de TCP, a diferencia del Telnet, que usa el 23.

Para el servidor necesitamos la versión libre y multiplataforma del protocolo SSH, llamada OpenSSH. Como es un programa muy utilizado en máquinas Linux profesionales, probablemente ya lo tenemos instalado y con el demonio funcionando; lo podemos comprobar mirando si el proceso `sshd` se está ejecutando, con `ps axu | grep sshd`, o también podemos mirar si tenemos el puerto 22 de TCP abierto, con `nmap bruguers -p 22`

En caso de que no lo tengamos instalado, sólo hay que hacer un `apt-get install ssh` y responder las preguntas que hará. Siempre podemos volver a hacer esta configuración guiada por preguntas con un `dpkg-reconfigure ssh`.

Una vez instalado y en funcionamiento, tendremos que comprobar que vaya bien: lo primero será hacer un `ssh` desde la misma máquina. Estando como cualquier usuario, hacemos un `ssh bruguers` y nos aparecerá un mensaje diciendo que no somos un usuario autorizado. Los usuarios autorizados son los que no tienen que escribir su contraseña para acceder. Por seguridad, no crearemos ninguno de estos. Decimos `yes` para continuar con la conexión, y el mensaje no volverá a salir más en esta máquina.

Siempre que conectemos a otra máquina, intentará acceder con el nombre del usuario que está realizando la conexión. Si queremos especificar el usuario, hay que poner `ssh máquina -l usuario` (l=login). De cualquier forma, nos pedirá una contraseña. Obviamente, hay que poner la contraseña de la cuenta que tiene el usuario en el ordenador al que se conecta (no en el propio ordenador). Después de

escribir la contraseña, entraremos al sistema de forma normal. Podemos salir de la shell con el comando `exit`.

Si hemos probado a conectar de forma local y funciona, es el momento de comprobar si conecta desde otro ordenador (de la misma red o de Internet). Necesitamos cualquier programa que soporte SSH (no se puede conectar directamente al puerto 22). Son fáciles de encontrar: por ejemplo, tenemos el ssh para Linux, el Putty para Windows y el NiftyTelnet para Macintosh. En cualquiera de ellos hemos de poner la IP del servidor, dejar el 22 como puerto, y conectar. Nos preguntará nombre de usuario y contraseña, y entraremos de forma habitual mediante una conexión segura que encripta toda la información que circula, cosa que nos permite no preocuparnos tanto por si alguien intercepta la comunicación.

Cada nuevo host que conecte al servidor hará aparecer el mensaje anterior de “Máquina desconocida; ahora la he añadido a la lista de máquinas conocidas.”

Si no funciona, hay que comprobar que el router filtre el puerto 22 TCP y lo redirija hacia el servidor.

Finalmente, hay que decir que no se puede estar tranquilo del todo sólo por haber instalado SSH; tenemos que vigilar que las máquinas que usemos para hacer la conexión no tengan ningún programa que pueda registrar las contraseñas (como por ejemplo un keylogger).

2.4.5 Web para Internet

Ya hemos puesto anteriormente el programa Apache sirviendo una web a los ordenadores de la red interna. A esta web no se puede acceder desde Internet poniendo la IP de la línea ADSL porque toda la red del instituto está detrás de un router.

Ahora queremos que haya dos webs diferentes, una con información general del instituto para los visitantes de Internet y otra sólo para los alumnos y profesores (usuarios de la red interna) donde puedan ver información que no está al alcance del público general, como por ejemplo notas, horarios de guardia de los profesores, fechas de las evaluaciones, etc.

Hay muchos métodos para servir páginas diferentes a redes diferentes, pero absolutamente todos requieren que se abra el puerto 80 en el router redirigido hacia un puerto privado del servidor (normalmente, también el 80). Algunas de las ideas que podemos implementar son:

- Una solución absurda y poco factible: poner dos Apache en el servidor, cada uno escuchando en un puerto diferente (ej. 80 y 81), y utilizar el 80 para Internet y el 81 para la red interna. Como hemos dicho, es absurdo porque daría conflictos en la configuración, consumiría el doble de memoria, y haría que se tuviese que escribir el número de puerto en los navegadores web.
- Poner un Apache y otro servidor web en el mismo ordenador: es igual de absurdo, porque lo que queremos es un programa que sirva páginas web, y eso ya lo hace Apache. Por tanto, el otro servidor no aportaría nada nuevo.
- Poner otra máquina con otro Apache (los dos escuchando en el puerto 80) y usar uno para Internet y otro para la red. No es inteligente, porque estaríamos desaprovechando los dos ordenadores. Sólo hace falta uno para hacer de servidor de una red sencilla.
- Utilizar el mismo ordenador, pero poniendo dos tarjetas de red para tener dos direcciones IP. Es demasiado complicado, pero funcionaría; Apache lo permite.
- Utilizar sólo una tarjeta de red, pero con IPs virtuales. Es más simple que la solución anterior, pero sigue siendo demasiado complicado. Tanto este método como el anterior están soportados por Apache y se conocen con el nombre de 'IP-based Virtual Host Support'. Lo que hace es mirar qué IP se ha pedido y mostrar una página u otra.
- Añadir alias al servidor DNS que montaremos en el siguiente apartado para hacer que dos nombres, por ejemplo interna.com y externa.com,

apunten a la misma IP, la del servidor web; y configurar Apache para que distinga qué se ha puesto al acceder, y así mostrar una página o la otra. Esto se conoce en Apache como 'Name-based Virtual Host Support'.

- Utilizar sólo una página, pero hacer que el contenido cambie dependiendo de la procedencia del visitante.
- Utilizar sólo una página y poner un link a una 'zona restringida' que pida contraseña. Como es muy complicado de cara a los usuarios, tampoco es el mejor método.

Las soluciones más sencillas y efectivas son las de los 'Virtual Hosts' y la de la página dinámica, pero la primera posibilidad la tendremos que eliminar porque, leyendo toda la documentación oficial en <http://httpd.apache.org/docs/vhosts>, vemos que sólo funciona en navegadores que lo soporten. Por tanto, usando un navegador sencillo o haciendo nosotros mismos la petición HTTP, ¿conseguiríamos ver cualquiera de las dos páginas! La solución que nos da Apache es la de ejecutar dos demonios httpd, pero esto requiere el doble de memoria, y es mucho más difícil de configurar: dos procesos, dos configuraciones, dos registros de log, etc. Y el doble de problemas. (No obstante, es apto para las webs de dos empresas que no quieran estar juntas de ninguna manera porque quieren configuraciones diferentes).

Por tanto, lo mejor es dejar Apache tal como está y utilizar el PHP y CGIs que ya hemos configurado para detectar si el visitante viene de la red interna o de Internet. Este método presenta otras ventajas:

- Se pueden compartir recursos con mucha facilidad ya que todo se encuentra en el mismo directorio, `/var/www`. Por lo tanto, podemos tener zonas comunes (cosa que con los otros métodos no podríamos hacer). También podemos hacer que los usuarios de una red lo puedan ver absolutamente todo y los otros sólo una parte.
- Los visitantes no se dan cuenta de que están siendo clasificados en dos grupos, ya que no ven nada extraño (a menos que pongamos mensajes de información en la web). Esto es bueno porque si un hacker ve desde Internet que se le ha clasificado como usuario no autorizado, le entrarán más ganas de acceder como usuario privilegiado.
- Si lo hacemos con PHP, podremos poner páginas dinámicas (con información no constante) o incluso que accedan a una base de datos MySQL para hacer listados, consultas, añadir registros, ...
- Es muy sencillo y es fácil de personalizar.

Pero este método también tiene algunos inconvenientes, básicamente porque la autenticación por IP nunca da mucha seguridad: hay muchas técnicas para hacer creer a un ordenador que somos una IP determinada cuando en realidad no lo somos. El IP spoofing y el ARP poison son las más comunes, pero evolucionan constantemente para crear ataques cada vez más sofisticados, hasta el punto en que casi es posible convertirse en el ordenador deseado y aprovechar sus privilegios.

Para evitar problemas, a la zona privada sólo dejaremos acceder a ordenadores de la red interna (con IP 192.168.0.x) exceptuando 192.168.0.1 (router), 192.168.0.0, 192.168.0.255. El router NO puede acceder a la página, aunque sea de nuestra red. Hay que tenerlo en cuenta porque el router es el elemento que separa Internet de la red interna, por tanto si alguien consigue entrar en el router (192.168.0.1) podría acceder a los sitios donde se permiten las IPs que empiezan por 192.168.0. Las direcciones acabadas en 0 y en 255 son especiales y no las pueden utilizar los ordenadores como IP, o sea que mejor hacer que tampoco puedan entrar, por si acaso.

Este es el ejemplo de un [fichero.php](#) protegido por IP (sólo pueden acceder los 192.168.0.x excepto .0, .1 y .255)

Esta página requiere una IP autorizada...


```
<?
$ip = GetHostByName($REMOTE_ADDR); // Capturamos la IP

echo "<BR> Accedes con la IP $ip <BR>";

$aut=strpos($ip,"192.168.0."); // Si la IP contiene el texto 192.168.0.
es autorizada

if ($aut === false || $ip=="192.168.0.1" || $ip=="192.168.0.0" ||
$ip=="192.168.0.255") { // Nota: sí, son TRES signos = al principio
    exit(";No estás autorizado!"); // Cierra la conexión y no sigue
leyendo el resto de la página
}

?>

<!-- Aquí sólo se puede llegar siendo autorizado (sinó, se habría
ejecutado el exit() ) -->
Perfecto, estás autorizado.<BR>
<U>Estos contenidos son para ti.</U><BR>
```

En vez de poner esto en cada página, podemos hacer lo mismo creando un fichero [aut.php](#) con:

```
<?
$ip = GetHostByName($REMOTE_ADDR); // Capturamos la IP
```

```
echo "<BR> Accedes con la IP $ip <BR>";
```

```
$aut=strpos($ip,"192.168.0."); // Si la IP contiene el texto 192.168.0.  
es autorizada
```

```
if ($aut === false || $ip=="192.168.0.1" || $ip=="192.168.0.0" ||  
$ip=="192.168.0.255") { // Nota: sí, son TRES signos = al principio  
    exit("¡No estás autorizado!"); // Cierra la conexión y no sigue  
leyendo el resto de la página
```

```
}
```

```
?>
```

Y después simplemente poner en `prueba.php`:

```
Esta página requiere una IP autorizada... <BR>
```

```
<?
```

```
include 'aut.php';
```

```
?>
```

```
Perfecto, estás autorizado.<BR>
```

```
<U>Estos contenidos son para ti.</U><BR>
```

De esta forma sólo hay que incluir una línea más en cada página que requiera comprobación de IP.

Este sistema ya es bastante seguro para los usuarios normales; de todas maneras, no tendremos que poner información privada ya que PHP también puede tener bugs. Si es necesario, pondremos una contraseña en algunas páginas, o usaremos lo conocido como 'sesiones' (<http://www.php.net/manual/es/ref.session.php>).

2.4.6 Servidor DNS

Para evitar tener que escribir la IP del servidor cada vez que queramos acceder, podemos escribir su nombre de host y hacer que un DNS (Domain Name Server) lo traduzca a la IP adecuada (ejemplo: tecnoblog.net a 192.168.0.53). Como los servidores DNS que nos da el ISP (Internet Service Provider) no los podemos modificar, montaremos nuestro propio servidor DNS. Es un servicio que utiliza el puerto 53 de TCP y el 53 de UDP.

De servidores DNS hay de dos tipos:

- **Para Internet**: son públicos y traducen básicamente nombres de dominios (pagina.es, subdominio.dominio.com, etc.). Estos servidores se han de comunicar entre ellos para conocer todos los dominios y las IPs correspondientes. Están controlados por 13 'root servers', situados en los Estados Unidos y en Japón, aunque recientemente se ha incorporado un 14º 'root server' en Madrid. Toda esta información se puede consultar en <http://root-servers.org/>
- **Para una red local**: estos sólo resuelven los nombres de host a sus IPs privadas. No aceptan peticiones del exterior. Es éste el tipo de servidor DNS que pondremos, y haremos que también pueda conectarse a los DNS de nuestro proveedor de Internet para resolver nombres de dominios.

Utilizaremos la versión del ISC (Internet Software Consortium, www.isc.org) del protocolo DNS, que se llama BIND (Berkeley Internet Name Domain). Consta del servidor de nombres de dominio (`named`) y de utilidades relacionadas con DNS.

La versión actual es la 9, que tiene soporte entre otras cosas para hacer caché de los DNS (un mismo dominio lo resolverá mucho más rápido la segunda vez que la primera). De todas maneras, conviene instalar siempre la última versión de `bind`, porque se han encontrado bugs importantes en versiones anteriores.

Haremos un `apt-get install bind9 bind9-doc dnsutils`, aunque el paquete `dnsutils` probablemente ya esté instalado. El `bind9-doc` es opcional, pero como la configuración es difícil nos irá bien la documentación (sobre todo la que se encuentra en Internet).

Ahora ya tenemos el demonio `named` funcionando a la perfección y listo para resolver hosts; pero lo mejor es revisar la configuración de `/etc/bind/named.conf`. Nota: en los ficheros de configuración hemos de respetar los signos y la tabulación (se utilizan sobre todo tabuladores y puntos).

Comenzaremos añadiendo las IPs de los servidores DNS que nos ha dado nuestro ISP. Se utilizarán para resolver peticiones de nombres de dominio que no sean de la Intranet.

```
forwarders {
    193.145.88.16;
    193.145.88.18;
};
```

Después añadiremos el nombre del dominio, que será `bruguers`. Coincide con el nombre de host del servidor, pero no pasa nada ya que definiremos otros, como `www` o `proxy`, para no liarse. Añadimos las siguientes líneas:

```
zone "bruguers" {
    type master;
    file "/etc/bind/bruguers.hosts";
};
```

Hemos de crear otra zona para hacer la operación inversa (traducir una IP a un nombre de dominio). Para esto crearemos la zona llamada `0.168.192.in-addr.arpa`, que es la IP `192.168.0.*` invertida (todos los PCs de la red tienen IP `192.168.0.cualquier número` ya que su máscara de subred es `255.255.255.0`). Para esta zona ponemos:

```
zone "0.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/192.168.0.rev";
};
```

Y ahora, claro, hay que crear `/etc/bind/bruguers.hosts` y `/etc/bind/192.168.0.rev`. Empezaremos por el primero:

```
bruguers. IN      SOA      linux. root (
                        1          ; Serial
                        8H         ; Refresh
                        2H         ; Retry
                        4W         ; Expire
                        1D )       ; Minimum TTL

bruguers.         IN      NS       linux.
router.bruguers. IN      A        192.168.0.1
linux.bruguers.   IN      A        192.168.0.200
proxy.bruguers.   IN      CNAME    linux
www.bruguers.     IN      CNAME    linux
t01.bruguers.     IN      A        192.168.0.131
```


Los parámetros de la primera sección (comentados a partir del carácter “;”) son más útiles para DNS públicos (los que sirven para Internet). Especifican tiempos, que pueden ir en segundos o en múltiplos mayores (ej: 2H = dos horas, 1D = un día, 4W = cuatro semanas). Dejaremos estos valores que hemos encontrado en configuraciones por defecto en páginas de documentación.

Cada nombre de host que queramos definir ha de seguir la misma estructura. Las palabras clave que hay en el fichero son:

IN NS: indica que este equipo es el servidor DNS

IN A: añade un nombre de host y su IP correspondiente

IN CNAME: define un alias para un host ya conocido

El otro fichero, `/etc/bind/192.168.0.rev`, ha de contener lo mismo pero preparado de otra forma:

```
0.168.192.in-addr.arpa.      IN          SOA          linux. root (
                              1            ; Serial
                              8H            ; Refresh
                              2H            ; Retry
                              4W            ; Expire
                              1D )         ; Minimum TTL
0.168.192.in-addr.arpa.      IN          NS          linux.
1.0.168.192.in-addr.arpa.    IN          PTR          router.bruguers.
200.0.168.192.in-addr.arpa.  IN          PTR          linux.bruguers.
131.0.168.192.in-addr.arpa.  IN          PTR          t01.bruguers.
```

Los parámetros son los mismos, y los hosts definidos también. Cuando añadamos más al primer fichero también los tendremos que poner en el segundo, naturalmente siguiendo el mismo modelo (y recordando que hay que usar la tecla TAB para separar los campos).

Hemos acabado la configuración. Lo que haremos ahora es vaciar el fichero `/etc/resolv.conf` del servidor para quitar las IPs del ISP contratado, ya que ahora ya no hacen falta (nuestro servidor DNS se comunica directamente con los root servers). Al dejar el fichero en blanco, Linux intenta resolver los nombres de dominio en la misma máquina.

Reiniciamos `named` ejecutando `rndc reload`, vamos a otra máquina, configuramos el DNS con la IP privada del servidor, reiniciemos si hace falta (depende del sistema operativo), y probamos a hacer un `ping` a `linux.bruguers`

```
PING linux.bruguers (192.168.0.200) from 192.168.0.132 : 56(84) bytes of data.
```

```
64 bytes from linux.bruguers (192.168.0.200): icmp_seq=1 ttl=255 time=0.778 ms
```

```
64 bytes from linux.bruguers (192.168.0.200): icmp_seq=2 ttl=255
time=0.667 ms
```

```
64 bytes from linux.bruguers (192.168.0.200): icmp_seq=3 ttl=255
time=0.574 ms
```

¡Funciona! Ha traducido `linux.bruguers` a la IP correspondiente, 192.168.0.200. También hemos de comprobar que funcionen los alias `www.bruguers` y `proxy.bruguers`, y que cada una de las estaciones que hemos añadido también sea accesible (en el ejemplo hemos definido `t01.bruguers`). No hemos de olvidar probar también a resolver páginas de Internet (ej. `google.com`). Si medimos cuánto tarda la primera vez y cuánto las posteriores comprobaremos la eficacia de su memoria caché.

Podemos hacer pruebas más detalladas con comandos como `dig router.bruguers` para ver si encuentra la IP o `dig -x 192.168.0.1` para ver si también funciona a la inversa (tendría que mostrar `router.bruguers`).

Cuando todo funcione, podemos volver a editar el fichero `/etc/resolv.conf` del servidor para hacer que quede así:

```
domain bruguers
nameserver 192.168.0.200
nameserver 193.145.88.16
nameserver 193.145.88.18
```

Esto hace que las peticiones DNS las resuelva el propio servidor (192.168.0.200), pero que si por alguna razón falla (por ejemplo, si el proceso `named` muere), el DNS funcione de manera estándar pasando las peticiones a los DNS del ISP. El `domain bruguers` hace que si no especificamos nada entienda los nombres como miembros del dominio llamado `bruguers` (por ejemplo, `proxy` haría referencia a `proxy.bruguers`, `tecno` a `tecno.bruguers`, etc.). Esta opción la podemos poner en todos los ordenadores del dominio (se hará de distinta forma dependiendo del sistema operativo).

El servicio de DNS ya funciona; pero nos quedan dos tareas por hacer:

- Pensar en un nombre para cada ordenador de la red. Si puede ser, que sean más descriptivos que `ord01`, `ord02`, `ord03`, ... Buenos nombres son: `tecno`, `biblio1`, `biblio2`, `almaster`, `profes`, etc. También podemos poner nombres más creativos; o -¿por qué no?- diferentes alias para cada ordenador.
- Configurar cada ordenador para hacer que use el servidor como DNS. Ya que hemos de hacer este tedioso proceso, aprovecharemos para hacer las otras configuraciones, como la del proxy. La ventaja es que ahora sólo hay que poner `proxy.bruguers` en vez de la IP entera en cada navegador de cada uno de los PCs.

2.4.7 Firewall

En un ordenador con tanta importancia dentro de la red no nos podemos olvidar de la seguridad: como es un servidor tiene muchos puertos abiertos y ofrece muchas formas de conexión, de entre ellas puede que algunas sean indeseadas.

Para hacer que sólo se acceda utilizando los servicios que hemos configurado, usaremos un cortafuegos ('firewall' en inglés), que acepta, deniega o ignora los paquetes que han de pasar por el ordenador. Hay otra herramienta más compleja y potente que los cortafuegos: los IDS (Intrusion Detection System) que detectan y registran patrones extraños en el uso de la red, como pueden ser el tráfico a altas horas de la madrugada o la llegada de paquetes TCP con cabecera errónea.

Debido a la complejidad de los IDS, pondremos sólo un cortafuegos ya que hace todo lo que necesitamos. Lo haremos a nivel de núcleo (es el kernel quien controla el filtrado de paquetes) utilizando el programa `iptables`. Una curiosidad: como han salido bastantes versiones estables del kernel de Linux con muchas diferencias entre ellas, se llamó de forma diferente al filtrado de paquetes de cada una. Así, tenemos que el programa es `ipfwadm` para kernels 2.0, `ipchains` para 2.2 e `iptables` para 2.4. Nosotros utilizaremos sólo `iptables`, el del kernel 2.4 (ya se ha explicado cómo actualizar el kernel fácilmente en la sección de 'Configuraciones básicas').

Como es una herramienta del núcleo, es probable que ya lo tengamos instalado (`iptables` para probarlo, `apt-get install iptables` para instalarlo). Ahora empezaremos viendo cómo funciona este cortafuegos:

Tenemos tres tablas, `INPUT`, `FORWARD` y `OUTPUT`, para catalogar los paquetes:

- `INPUT`: los paquetes que llegan al servidor desde el exterior o desde la misma máquina pasan por la tabla de `INPUT` ('entrada'). Es de donde vienen todos los ataques.
- `FORWARD`: los paquetes que llegan al servidor para ser enrutados hacia otro sitio pasan por la regla `FORWARD` ('hacer que continúe'). Hay que activar una opción antes para permitir el forwarding.
- `OUTPUT`: los paquetes generados en el mismo ordenador y listos para ser enviados hacia el exterior pasan por la tabla `OUTPUT` ('salida').

Cada tabla consta de una serie lógica de reglas (ordenadas) que deciden el destino del paquete. El paquete acabará básicamente de una de estas formas:

- `ACCEPT`: se deja pasar el paquete; el firewall no actúa para nada.

- **REJECT**: se deniega el paquete. El firewall contesta diciendo que no quiere hacer la conexión, por tanto el emisor del paquete se entera.
- **DROP**: ignora el paquete. Es como si el ordenador estuviera apagado y el paquete se perdiera.

Para entender mejor la diferencia entre **REJECT** y **DROP** (llamado '**DENY**' en versiones anteriores), nos irán bien unos conceptos básicos de TCP/IP.

Sabemos que en IPv4, cada paquete tiene en la cabecera TCP unos flags que determinan la finalidad del paquete. Estos son: URG, ACK, PSH, RST, SYN, FIN. Hay dos más que completan el byte, CWR y ECE, pero no se usan mucho.

Para establecer una conexión a una máquina y un puerto determinados, hay que hacer el conocido como 'three way handshake' (saludo de las tres vías). Consiste en enviarle un paquete con el bit SYN (SYNCHRONIZE) activado. Nos contestará con un paquete con el bit ACK (ACKNOWLEDGEMENT) activado y hará lo mismo que hemos hecho nosotros: enviar un paquete con SYN y esperar un ACK nuestro; entonces ya puede empezar a enviar y recibir datos. Como enviar un ACK y luego un SYN se puede simplificar en sólo un paquete con los bits SYN y ACK activados, el 'three way handshake' queda así:

Nosotros		Destino	Descripción
SYN	----->		"-Eh, ¿me recibes? Quiero conectar contigo."
	<-----	SYN/ACK	"-Sí, te recibo. ¿Comenzamos ya?"
ACK	----->		"-De acuerdo, comencemos."

Pero esto sólo pasa si el puerto está abierto. Si un puerto está cerrado, al enviar el SYN responde con un RST/ACK y finaliza la conexión. `iptables` por defecto contesta de otra forma, con un mensaje ICMP Port Unreachable, pero lo podemos especificar con el parámetro `--reject-with`

Por tanto, volvamos a definir las tres opciones **ACCEPT**, **REJECT** y **DROP**, pero de una manera más técnica. Si enviamos un SYN al ordenador destino, la respuesta que recibiremos dependerá de la regla utilizada:

- **ACCEPT**: recibiremos un SYN/ACK y se establecerá la conexión.
- **REJECT**: recibiremos un ICMP Port Unreachable o un RST/ACK (se puede escoger en la configuración). Por tanto, el servidor corta la conexión.
- **DROP**: ¡no recibiremos nada! Al final será nuestro ordenador el que, cansado de enviar SYNs, acabe la conexión. Esto es muy interesante porque podemos hacer ver que un ordenador está apagado si le decimos

que ignore todos los paquetes no deseados.

Nuestro trabajo si queremos crear un firewall es definir estas reglas para las tres tablas ([INPUT](#), [FORWARD](#) y [OUTPUT](#)), y para hacerlo lo primero es crear un esquema de cada tabla. Por ejemplo, un servidor web podría hacer lo siguiente:

Paquetes recibidos ([INPUT](#)):

1. De la interfaz `lo` (loopback, el propio ordenador): aceptar
2. Viene al puerto 80 TCP: aceptar
3. Sinó, descartar

Paquetes de paso ([FORWARD](#)):

1. Descartar

Paquetes enviados ([OUTPUT](#)):

1. De la interfaz loopback: aceptar
2. Tiene que ir al puerto 80 TCP del ordenador destino: aceptar
3. Sinó, descartar

Esto, traducido a comandos de [iptables](#), se escribe:

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -j DROP

iptables -A FORWARD -j DROP

iptables -A OUTPUT -o lo -j ACCEPT
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -j DROP
```

Para comprender cada comando hay que saber los parámetros del programa. Podemos verlos todos con `man iptables`, pero resumimos los importantes aquí:

```
iptables -A <tabla> -i <interfaz de entrada> -o <interfaz de salida>
-s <IP de origen> -d <IP de destino>
-p <protocolo> --sport <puerto origen> --dport <puerto destino>
-j <acción>
```

Los nombres de las opciones comentadas son las iniciales de las siguientes palabras inglesas: i='input' o='output' s='source' d='destination' p='protocol' j='jump' A='add' F='flush' P='policy' m='module'

Podemos mejorar este ejemplo añadiendo unas líneas que borren las configuraciones anteriores de cada tabla:

```
iptables -F INPUT
iptables -F FORWARD
iptables -F OUTPUT
```

Hemos visto que en cada tabla hay una sentencia final que se cumple para los paquetes que no cumplen ninguna otra regla; a esta regla se le llama política de una tabla, y se ha de destacar utilizando el comando `-P` en vez de `-A`

El ejemplo, por tanto, quedaría así:

```
iptables -F INPUT
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -P INPUT DROP

iptables -F FORWARD
iptables -P FORWARD DROP

iptables -F OUTPUT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
iptables -P OUTPUT DROP
```

Aún podemos hacer una optimización a este ejemplo: sabemos que si una conexión es aceptada, lo será hasta el final, y todos los paquetes que se envíen no se tendrían que contrastar con cada regla. Como podemos distinguir entre una nueva conexión (bit SYN activado) y una conexión establecida (posterior al 'three way handshake'), lo podemos aplicar al cortafuegos. Utilizaremos el módulo `state` de la siguiente manera:

```
iptables -F INPUT
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -m state --state NEW -i lo -j ACCEPT
iptables -A INPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT
iptables -P INPUT DROP

iptables -F FORWARD
iptables -P FORWARD DROP

iptables -F OUTPUT
```

```
iptables -A OUTPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A OUTPUT -m state --state NEW -o lo -j ACCEPT
iptables -A OUTPUT -m state --state NEW -p tcp --sport 80 -j ACCEPT
iptables -P OUTPUT DROP
```

Y éste es el firewall final para el ejemplo del servidor web. Sólo son una serie de órdenes que modifican parámetros del kernel, por tanto cuando se reinicie y se vuelva a cargar Linux (el núcleo del sistema operativo) ya no estarán activados. Es muy fácil solucionar esto; sólo hay que grabar todos los comandos en un fichero, hacerlo ejecutable, ponerlo en el directorio `/etc/init.d/`, y crear los enlaces a los niveles de ejecución que queramos (en `/etc/rc*.d/`) para que se ejecuten cada vez que se encienda el servidor.

Esto es sólo un ejemplo de un servidor muy simple, pero ahora necesitamos hacer el cortafuegos apropiado para nuestra máquina. Como lo que queremos es cerrar puertos, primero hay que saber cuáles tenemos abiertos mediante un escaneado de puertos. Hay muchos programas que lo hacen, pero utilizaremos `nmap`, que es el más usado por los hackers de Linux.

Iremos a otra máquina (las conexiones hay que hacerlas desde fuera), instalamos `nmap` (preferiblemente bajo el sistema Linux), y ejecutamos el siguiente comando como root:

```
nmap -sS IP_DEL_SERVIDOR -p 1-65535
```

Esto hace un escaneo de puertos de tipo SYN (el tipo por defecto cuando lo hace root), y prueba todos los puertos TCP. 65535 son muchos puertos; pero como esto sólo hay que hacerlo una vez, vale la pena esperar un poco más.

Hemos recibido la siguiente salida:

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on bruguers (192.168.0.200):
(The 65520 ports scanned but not shown below are in state: closed)
Port      State      Service
9/tcp     open      discard
13/tcp    open      daytime
21/tcp    open      ftp
22/tcp    open      ssh
25/tcp    open      smtp
37/tcp    open      time
53/tcp    open      domain
80/tcp    open      http
111/tcp   open      sunrpc
```

```
113/tcp    open      auth
139/tcp    open      netbios-ssn
515/tcp    open      printer
1024/tcp   open      kdm
3128/tcp   open      squid-http
8080/tcp   open      http-proxy
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 70 seconds
```

Ahora haremos el escaneo de puertos UDP (normalmente olvidados, pero que pueden traer muchos problemas fáciles de explotar). Como es muchísimo más lento (para probar todos los puertos puede tardar bastantes horas) haremos sólo los puertos por defecto: hacemos un `nmap -sU IP_DEL_SERVIDOR`; y ahora vamos al servidor y, desde él mismo, sí que podemos hacer un `nmap -sU IP_DEL_SERVIDOR -p 1-65535`. Como ahora no interviene la red, irá mucho más rápido y tardará aproximadamente un minuto. La salida que nos muestra el segundo comando (que, por cierto, incluye todo lo que ha salido al hacer el primero) es:

```
Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Interesting ports on bruguers (192.168.0.200):
(The 65526 ports scanned but not shown below are in state: closed)
Port      State      Service
9/udp     open      discard
53/udp    open      domain
111/udp   open      sunrpc
137/udp   open      netbios-ns
138/udp   open      netbios-dgm
1024/udp  open      unknown
1025/udp  open      blackjack
1026/udp  open      unknown
3130/udp  open      squid-ipc
```

```
Nmap run completed -- 1 IP address (1 host up) scanned in 67 seconds
```

Lo que haremos es modificar el ejemplo del cortafuegos para el servidor web pero permitiendo el acceso sólo a los puertos que sabemos qué son: como el `nmap` muestra al lado el nombre y en Internet es muy fácil encontrar a alguien que también lo haya preguntado, no tendremos problemas para saber a cuál hemos de dejar entrar y a cuál no. De puertos para enviar información hacia el exterior, los dejaremos todos ya que no queremos poner ninguna restricción. El forwarding no lo habilitamos.

Para ahorrar líneas, añadiremos el módulo `multiport` y especificaremos los puertos de destino con `--dports` en vez de `--dport` (porque ahora pondremos más de uno). El script resultante, con algún comentario y con la lista de puertos permitidos, es:

```
echo Cargando reglas de iptables...
iptables -F INPUT
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
iptables -A INPUT -m state --state NEW -i lo -j ACCEPT
iptables -A INPUT -m state --state NEW -i eth0 -p tcp -m multiport --
dports 21,22,53,80,139,515,3128,8080 -j ACCEPT
iptables -A INPUT -m state --state NEW -i eth0 -p udp -m multiport --
dports 53,137,138,3130 -j ACCEPT
iptables -P INPUT DROP

iptables -F FORWARD
iptables -P FORWARD DROP

iptables -F OUTPUT
iptables -P OUTPUT ACCEPT
```

Esto lo tenemos que grabar en `/etc/init.d/`, pero vemos que ya hay un script llamado `iptables`, que es mucho más complicado y que permite cargar y guardar configuraciones, reiniciar y parar el servicio, y muchas cosas más. Como nuestro script es mucho más sencillo y fácil de entender y de modificar, cambiaremos de nombre el `iptables` actual (a `iptables_ANTIGUO`, por ejemplo) y pondremos el nuestro con el nombre de `iptables`. Muy importante: no hay que olvidar hacerlo ejecutable con `chmod a+x iptables`.

Hemos de crear los enlaces de los directorios `/etc/rc*.d`, en concreto lo haremos para los niveles 2, 3 y 5, que son los que pueden necesitar cortafuegos (los modos multiusuario). Haremos que se cargue más o menos en último lugar (posición 92 de las 100 posibles), aunque tampoco importa mucho el orden. Ejecutamos:

```
ln -s /etc/init.d/iptables /etc/rc2.d/S92iptables
ln -s /etc/init.d/iptables /etc/rc3.d/S92iptables
ln -s /etc/init.d/iptables /etc/rc5.d/S92iptables
```

Y al reiniciar o cambiar de runlevel (`init NÚMERO_DE_RUNLEVEL`) ya estará funcionando. Lo podemos comprobar volviendo a hacer un `nmap` de la misma manera. Veremos que sólo se ven los puertos que hemos escrito en la lista, y que además es bastante más lento. La lentitud es debida a que no contesta “Este puerto está cerrado” sino que se calla debido al `DROP` que le hemos puesto por defecto. La lentitud es una buena arma para cansar a quienes nos hagan escaneos de puertos a menudo, mientras que a nosotros no nos afecta. No obstante, podemos hacer que conteste “Puerto cerrado” cambiando el `DROP` por un `REJECT`.

Ahora comentaremos un problema práctico que puede haber salido: como hemos dicho al principio, `iptables` es sólo para kernels 2.4, pero la Debian estable actual tiene un kernel 2.2 por defecto; por tanto hemos de actualizar el kernel. Esto no siempre es posible, como por ejemplo en nuestro caso, porque el driver de la tarjeta de red del kernel 2.4.18 no la reconocía (ya que usaba diferentes módulos: el `rtl8139` en el 2.2 y el `8139too` en el 2.4)

En estos casos podemos hacer varias cosas:

- Buscar nuevos drivers, recompilar los antiguos, o recompilar el kernel hasta que funcione. Lo conseguiremos, seguro, pero costará mucho tiempo.
- Volver al kernel 2.2 e implementar el cortafuegos con `ipchains`. Es muy parecido a `iptables` pero se han de hacer algunas adaptaciones a las reglas. Si tenemos tiempo, será fácil (pero poco útil) volver a estudiar cómo se configura un cortafuegos en kernels 2.2
- Dejar el kernel 2.2, como antes, y, como sólo queremos cerrar unos cuantos servicios, ir uno por uno. Por ejemplo, en `/etc/inetd.conf` encontramos la mayoría de los servicios que no queremos, y sólo hay que comentar las líneas con `#` para desactivarlos. Esta es la solución temporal que hemos usado; si más adelante quisiéramos sacar provecho de las ventajas de un cortafuegos tendríamos que solucionar el problema original.

3 CONCLUSIONES

3.1 Evaluación general

Hemos conseguido un ordenador mucho más potente que cualquier otro de la red, incluso cuando sólo funciona a 100 Mhz. y tiene 48 Mb. de RAM. Por lo tanto, hemos aprovechado muy bien los recursos, sobre todo gracias a Linux. Si hubiésemos instalado otro sistema operativo (como Windows) no se habría podido hacer casi nada en este ordenador.

Finalmente, el ordenador dispone de todos los servicios que habíamos propuesto al principio, algunos mejor implementados que otros, pero que funcionan bien.

Hemos elegido siempre la opción correcta al no dejar las configuraciones por defecto y revisarlas, aunque para hacerlo bien hecho sería necesario leerse toda la documentación hasta llegar a entender cada fichero de configuración del sistema. Algunos son muy complicados y ni aún haciendo esto conseguiríamos los resultados que queremos a la primera.

En resumen, el ordenador ha quedado bien, presentable, ya que todo lo que hace lo hace bien: proxy, DNS, SSH; PHP, CGI, FTP y servidor web (con página de estadísticas) funcionan perfectamente; tenemos algunas dudas en cuanto a Samba por el problema comentado de la necesidad de crear el mismo usuario en cada máquina Windows, pero como eso también pasa entre Windows NT y Windows 98, podemos pensar que también funciona bien.

El firewall está comprobado en otros ordenadores, pero en éste en concreto no funcionó por el problema con la tarjeta de red y se tuvo que cerrar cada puerto a mano. El resultado es casi el mismo, pero no queda tan profesional.

Estos pequeños problemas con el hardware y con los otros sistemas operativos los podemos resolver leyendo toda la documentación incluida y buscando más por Internet. Después de probar cosas durante un tiempo, conseguiremos dejar el servidor 'perfecto'. Funcionará casi igual que ahora, pero ya sabremos si es normal que esto pase o no.

También hay que destacar que lo hemos hecho todo con software libre (podemos ejecutar [vrms](#) para asegurarnos) y que no nos hemos gastado nada de dinero, aunque todo es completamente legal. Si lo hubiésemos hecho con otros sistemas operativos propietarios tendríamos que haber pagado cantidades enormes de dólares o

euros y los resultados no serían mejores que los que hemos conseguido con Linux (al contrario, aún tendríamos más problemas que no podríamos resolver nosotros mismos).

3.2 Otras utilidades

Podemos aprovechar mucho un ordenador con Linux instalado, por muy viejo que sea. Como va a estar mucho tiempo encendido sin parar, es ideal para hacer tareas como:

- Descargar archivos grandes y páginas web enteras, dejándolos en el FTP para poder recuperarlos después desde cualquier ordenador de la red. Lo podemos hacer con [wget](#).
- Participar en redes de intercambio de archivos; siempre que sean legales, claro. [mldonkey](#) es una buena opción.
- Hacer cálculos matemáticos complejos. Hay muchos programas matemáticos muy avanzados para Linux. Un ejemplo es [Scilab](#). El problema es que 100 Mhz. no es mucha potencia...
- Trabajar en paralelo con otros ordenadores; o sea montar un clúster Beowulf. Muchas empresas y centros de investigación hacen exactamente esto, aunque parezca extraño. Cogen componentes viejos que la gente no quiere, los juntan para montar PCs (u otras plataformas), les instalan Linux u otro sistema operativo abierto con soporte de red, y los ponen a trabajar en tareas como el renderizado de imágenes 3D, fractales (matemáticas), cálculo de órbitas de planetas (astrodinámica), flujos turbulentos (hidráulica), y muchas cosas más.
- Programar: en Linux están las mejores herramientas para hacer código estándar, y están soportados la mayoría de los lenguajes: C/C++, Perl, Python, MONO::, Tcl/Tk, assembler, etc.
- Programar tareas como, por ejemplo, enviar e-mails el primer día de cada mes a los profesores con información que les interese, modificar datos de alguna página externa periódicamente, etc.

También podemos decidirnos por instalar el modo gráfico y utilizar programas profesionales como [blender](#) (diseño 3D), [gimp](#) (retoque fotográfico), [mozilla](#) (navegador web), [openoffice](#) (suite de aplicaciones), [mplayer](#) (visualizador de vídeos), etc., todo funcionando en un ordenador sencillo pero consiguiendo resultados decentes.

4 ANEXOS

4.1 Comandos importantes

Éstas son algunas de las órdenes necesarias para utilizar Linux. Podemos ver la ayuda completa con la orden `man` seguida del nombre del programa.

- `ls`: hace un listado de los archivos. Es conveniente utilizar `ls -l` para ver más propiedades.
- `pwd`: escribe el nombre del directorio actual
- `cd directorio`: entra en un directorio. Sin parámetros, va al directorio `$HOME`
- `mkdir/rmdir directorio`: crear y borrar directorios vacíos
- `cp origen destino`: copia los archivos de origen al directorio de destino
- `mv origen destino`: mueve los archivos, o renombra (de hecho es lo mismo)
- `rm archivo`: borra un archivo. `rm -r` para directorios (ir con cuidado)
- `touch archivo`: crea un archivo en blanco, o cambia la fecha de modificación si ya existía
- `cat archivo`: muestra el contenido del archivo
- `less archivo`: muestra el contenido del archivo haciendo pausas cuando no cabe en pantalla
- `file archivo`: muestra el tipo de archivo fijándose en su contenido
- `chmod permisos archivo`: cambia los permisos de un archivo
- `chown/chgrp usuario archivo`: cambia el propietario o el grupo de un archivo
- `ln origen destino`: crea un enlace simbólico. Puede ser 'enlace duro' o normal
- `alias orden="orden equivalente"`: nos ahorra el escribir lo mismo
- `echo Texto`: escribe el texto, normalmente por pantalla
- `date`: muestra la hora del sistema. También sirve para cambiarla
- `uptime`: muestra el tiempo que lleva encendido el PC y la carga de la CPU
- `ps`: muestra los procesos activos. Es interesante usar `ps aux` para verlos todos
- `top`: monitoriza los procesos

- `kill pid_del_proceso`: matar un proceso o enviarle una señal
- `who`: muestra qué usuarios están conectados
- `id`: muestra información sobre el usuario actual
- `su usuario`: permite identificarse como otro usuario (por defecto root)
- `reset`: si vemos códigos ASCII extraños en la consola se arreglará ejecutándolo
- `reboot`: reinicia el ordenador
- `halt`: apaga el ordenador

Otros programas útiles (que puede que haya que instalar) son:

- `vim`: versión mejorada de `vi`, editor que está en todos los Linux
- `nano` o `pico`: editores de fácil uso, no tan potentes como `vi` o `vim`
- `elinks`: versión mejorada de `lynx`, navegador de Internet en modo texto
- `wget`: para bajar cosas de Internet
- `ping`: para comprobar si un equipo está activo
- `telnet` y `ssh`: para conectar a un puerto de un ordenador
- `ftp`: para conectar a un FTP
- `nmap`: escaneador de puertos
- `nc`: netcat, para hacer conexiones de forma más avanzada
- `netstat`: muestra las conexiones que mantiene el ordenador
- `gcc`: compilador de C/C++, utilizado para compilar el código fuente de los programas
- `tar` y `gzip`: empaquetador y compresor/descompresor de archivos.
- `cdrecord`: para grabar imágenes de CD creadas, por ejemplo, con `mkisofs`
- `dict`: da la definición (en inglés) de palabras y siglas, por Internet
- `aview`: para ver imágenes en códigos ASCII mediante la librería `AALib`
- `mpg123`: para escuchar MP3 desde consola
- `mplayer`: reproductor de vídeo y DVD. En consola los podemos ver en ASCII
- `fsck`: escanea un sistema de archivos y busca y corrige los errores

Es muy importante saber combinar estos programas con 'pipes' (traducido por 'tuberías') y redireccionadores. Ejemplos (del shell `bash`):

```
apt-cache search php | less    # Pipe que envía la salida del
primer comando al less para poder ver el listado de paquetes por páginas
```

```
ps axu | grep ssh    # Pipe que muestra los procesos activos que
contienen la palabra ssh
```

```
nmap pc > puertos_abiertos    # Graba el listado de puertos
abiertos del host pc en un fichero
```

```
nc -l -p 6000 <fichero    # Escucha en el puerto 6000 TCP y
responde con el contenido del fichero cuando alguien se conecta (útil
para transmitir ficheros entre ordenadores sin FTP)
```

```
cp ~/* carpeta & calendar    # Copia los archivos de nuestro
directorio personal (representado por $HOME o por ~ ) a la carpeta
especificada y, dejando en segundo plano este proceso, nos muestra las
efemérides del día (sin tener que esperar a que acabe de copiar todos
los archivos)
```

```
mv b.bak b; cp b b2    # Hace una acción y después la otra
```

```
apt-get update && apt-get upgrade    # Baja la nueva lista de
paquetes y, si no ha habido ningún problema, actualiza los nuevos
paquetes
```


4.2 Seguridad

Nuestro ordenador ya es muy seguro; es poco probable que algún alumno lo pueda estropear 'por error' o 'sin querer'. Si queremos ser más paranoicos (nunca es mala idea) podemos hacer muchísimas cosas para mejorar la seguridad del servidor y de todos los datos que hay dentro (tanto pensando en intrusos como en accidentes). Por ejemplo:

- Poner contraseña a la BIOS para que el ordenador no se pueda encender sin escribirla. No obstante, para un hacker que tuviese acceso físico al ordenador sería muy fácil entrar: sólo hay que abrir el ordenador y quitar la pila de la placa base durante un rato para borrar la contraseña. Naturalmente, nos daríamos cuenta de que alguien lo ha hecho porque veríamos que ya no la pide.
- Hacer que la BIOS no arranque desde disquet también es muy interesante para proteger el sistema, pero entonces no tenemos que olvidar la contraseña de la BIOS porque sinó también habría que quitar la pila de la placa base.
- Para los más paranoicos de todos: la caja del ordenador se puede cerrar con llave (no la llave que viene junto con el manual de instrucciones, sinó con alguna de más calidad). Si lo hacemos, ya nadie podrá abrir el ordenador sin tener que llevárselo del instituto. Esto, aunque parezca algo extraño, se hace a los ordenadores importantes (Pentágono, Ejército, etc), donde algunos ordenadores tienen incluso una 'armadura' de hierro para resistir los posibles ataques nucleares.
- No trabajar como root para las tareas diarias (navegar, escribir e-mails, programar, etc). Es por diversos motivos: primero; las equivocaciones pueden salirnos más caras, ya que tenemos todo el derecho a borrar cualquier cosa, aunque sea importante; y segundo, algún programa puede tener un bug que dé acceso a un hacker con los privilegios del usuario que lo ejecuta. Si este usuario tiene pocos privilegios, el problema no es tan importante.
- No irse y dejar el ordenador con sesiones abiertas en ninguna terminal. Podemos hacer `exit` para salir o instalar el programa `vlock`, que la bloquea y pide contraseña.
- Tener mucho cuidado con los programas 'setuid root' (con privilegios del usuario root), ya que un bug -por ejemplo, un buffer overflow- podría elevar los privilegios de un usuario normal. Podemos ver cuáles tenemos en el sistema con `find / -user root -perm -4000 -print`
- Poner buenas contraseñas (sobre todo la de root): en ellas no ha de aparecer el nombre de login (ej: para el usuario `pepito`, `pepito23` es una mala contraseña). Han de tener números y letras (si puede ser, intercalando mayúsculas y minúsculas) y han de ser tan absurdas que

nadie pueda imaginárselas nunca. Por la tanto, no han de ser palabras que se puedan encontrar en un diccionario. Podemos comprobar la calidad de las contraseñas intentando crackearlas con el programa líder en este sector; el John the Ripper.

- Actualizar a la versión en pruebas de Debian. La versión 'testing' es la que después se convertirá en estable. No es la misma que la inestable, que es en la que trabajan los desarrolladores de Debian. Si actualizamos es para tener las últimas versiones de todos los paquetes y poder hacer actualizaciones más a menudo. Hay que pensarlo bien antes de dar este paso porque volver hacia atrás costaría mucho más (¡habría que actualizar los paquetes con versiones más viejas!). Para hacerlo sólo hay que editar `/etc/apt/sources.list`, cambiar la palabra `stable` por `testing` y hacer un `apt-get update && apt-get dist-upgrade -u`
- Si alguna vez queremos dejar que alguien se conecte al servidor y trastea un poco para ver qué es Linux y cómo funciona, en vez de crear un nuevo usuario y darle acceso por SSH podemos darle el acceso por Telnet (puerto 23) habiendo configurado antes `ttysnoop`, un programa que permite ver y participar en las conexiones que hacen los usuarios a nuestra máquina.
- Si queremos que un usuario se pueda conectar habitualmente por SSH o Telnet pero tenemos miedo de que tenga conocimientos suficientes para hacer cosas que no queremos; lo primero será crear un entorno idéntico a un sistema de Linux en su directorio y hacer que al conectarse se haga un `chroot` a ese directorio para que se convierta en su directorio raíz ('/'). Así, aunque crackease `/etc/passwd` o consiguiese convertirse en root de otra forma, sólo lo habría hecho dentro de su árbol de ficheros, ya que el fichero `/etc/passwd` que ha usado está situado realmente en `/home/carpeta/etc/passwd`
- De tanto en tanto hemos de comprobar si realmente es necesario todo lo que hay instalado. Un `dpkg -l` muestra todos los paquetes que hay en el sistema. Hay que evitar tener cosas que no sabemos qué son, y borrar las que no usamos más.
- Nos podemos apuntar a las listas de correo de Debian para saber si sale alguna vulnerabilidad importante.

4.3 Mantenimiento

El servidor probablemente continuará funcionando a la perfección durante mucho tiempo, y sin tener que apagarlo, pero con el tiempo se irá quedando anticuado. Algunas de las cosas que hemos de hacer, ordenadas por importancia, son:

- Actualizar el sistema con `apt-get update; apt-get upgrade -u` (el `-u` es sólo para ver el nombre de los paquetes). Esto lo podemos hacer cada semana o cuando sepamos que han salido versiones nuevas de algún programa que corrigen vulnerabilidades.
- Mirar los logs para ver errores de funcionamiento en algún programa.
- Cada vez que salga una nueva versión estable de Debian, hacer un `apt-get dist-upgrade -u` para actualizar toda la distribución.
- Ver si hay suficiente espacio en el disco duro, y si no hay, qué es lo que lo ocupa para borrarlo. Si son logs, podemos hacer que se roten.
- Comprobar de vez en cuando el estado del disco duro para ver si hay sectores erróneos con la utilidad `fsck`. ¡Cuidado!: la unidad tiene que estar desmontada antes de corregir errores (lo podemos hacer arrancando desde disquet).
- Actualizar a IPv6 cuando sea necesario. Poco a poco, todos los programas se van adaptando y muchos ya aceptan direcciones en formato IPv6 (la versión 6 del protocolo IP, ampliado para abastecer a más usuarios de todo el mundo).
- Si queremos dar ejemplo sobre el uso del software libre podemos instalar el `vrms` ('virtual Richard M. Stallman') para que nos avise si tenemos programas no libres en el ordenador...
- El software puede mantener activo un servidor durante mucho tiempo, pero el hardware se va estropeando. Típicamente, los equipos se apagan como mínimo cada nueve meses para dejar descansar la memoria RAM.

4.4 Glosario de términos

Como para usar Linux hay que tener muchos conocimientos teóricos y prácticos, listaremos aquí las palabras que pueden crear confusión. No salen nombres de marcas comerciales ni de programas; sólo de los más importantes.

- **Administrador** : persona encargada de llevar una red y sus elementos, entre ellos el ordenador central. En Linux el administrador suele ser el usuario `root`.
- **Buffer overflow (desbordamiento de buffer)** : bug muy común que consiste en poner más caracteres de los que caben cuando un programa pregunta un dato. Estos caracteres sobrantes pueden escribir partes de la memoria y hacer que se ejecute algún código en concreto (por ejemplo, el programa `/bin/sh`, para conseguir una cuenta en el sistema). Se pueden explotar de cualquier forma, incluso desde programas sencillos como navegadores web.
- **Bug** : problema en un programa que hace que no funcione bien. Frecuentemente son amenazas para la seguridad del sistema, y han de ser corregidos rápidamente.
- **ChangeLog** : literalmente, “registro de los cambios”. Es un documento que explica qué hay nuevo y qué se ha arreglado en cada versión de un programa.
- **Demonios (daemons)** : son programas que se ejecutan en segundo plano al iniciar el ordenador, y que hacen de servidores. Esta es la razón por la que llevan la letra d al final del nombre: `telnetd`, `sshd`, `ftpd`, `httpd`, etc.
- **DoS** : Denial of Service (denegación de servicio): resultado de un ataque a un ordenador que hace que deje de ofrecer los servicios habituales. Un ejemplo de DDoS (Distributed DoS) es cuando centenares de ordenadores se conectan a la vez a uno solo hasta que consiguen que se sobrecargue y no trabaje de forma normal.
- **IDS** : Intrusion Detection System; programa que detecta las acciones no permitidas en una red y actúa en consecuencia. Por ejemplo, si ve que se intenta hacer un ataque por un puerto, lo cierra durante un rato a la IP del atacante y envía un mensaje al móvil del administrador. Son programas muy potentes.
- **DNS** : Domain Name Server, o servidor de nombres de dominio. Es el ordenador encargado de traducir las direcciones de Internet a direcciones IP. A esta operación se le llama 'resolver' un nombre de host. Por ejemplo, `google.com` se resuelve a `216.239.51.100`. Actualmente hay 14 'root servers' en todo el mundo; el último que se creó está en Madrid.

- **Exploit (o xexploit)**: programa sencillo que aprovecha un bug para provocar un error en un programa, haciendo que el usuario o hacker aumente sus privilegios (normalmente la finalidad que se quiere conseguir es el acceso como root). Se pueden encontrar muchos por Internet, todos en lenguaje C.
- **FAQ**: Frequently Asked Questions (preguntas más frecuentes). Lo encontramos sobre todo en la documentación de programas.
- **Firewall (cortafuegos)**: programa que bloquea el acceso a algunos puertos del ordenador para evitar las intrusiones no deseadas.
- **Flag (traducido por 'bandera')**: en programación, variable de tipo booleano (cierto/falso, 1/0, sí/no, etc) que se utiliza para activar o desactivar una opción. Por ejemplo, encontramos flags en la cabecera de los paquetes TCP, o en los ficheros especiales del directorio `/proc/sys`
- **FSF**: Free Software Foundation. Fundación iniciada por Richard Stallman creadora del movimiento del software libre, y que comenzó el sistema operativo GNU.
- **gcc**: GNU C Compiler, el compilador de lenguaje C de la GNU. Aunque parezca que hay muchos compiladores de C y C++, en Linux por defecto está el `gcc` (o `cc`) y el `g++` (o `c++`) para C++. `cc` es un link a `gcc`, y `c++` es un link a `g++`. Por tanto, sólo está el `gcc` y el `g++`
- **GID**: Group ID, parecido al UID pero aplicado a grupos.
- **GNU**: siglas de GNU's Not Unix. Es un proyecto de la Free Software Foundation para crear un UNIX libre.
- **GPL**: General Public License. Una licencia que creó la GNU para proteger el software libre. Debian incorpora únicamente software GPL.
- **Hacker**: un hacker es una persona que disfruta con su trabajo. Los hackers informáticos son expertos en su materia, y les interesan los temas avanzados y difíciles (como por ejemplo demostrar que pueden entrar en un sitio donde teóricamente nadie puede entrar).
- **Host (o hostname)**: nombre de un equipo que lo identifica en la red. Hay algunos predefinidos, como `localhost`, pero en `/etc/hosts` podemos definir más.
- **HOWTO**: manual de instrucciones sobre cómo hacer algo. ('How to... ?' en inglés).
- **ISP**: Internet Service Provider. Empresa que nos da la conexión a Internet.
- **Kernel**: núcleo de un sistema operativo, que hace de intermediario entre el usuario y el hardware. En este trabajo se usa el kernel Linux, pero hay otros como el Hurd (aún en una fase muy primitiva).
- **Keylogger**: tal como dice la palabra, es un programa que queda residente en segundo plano y graba todo lo que se escribe con el teclado. El mejor método para interceptar contraseñas; el inconveniente es que

el hacker ha de tener acceso a la máquina porque tiene que volver para revisar el log.

- **Lilo**: gestor de arranque de los diferentes sistemas operativos instalados. Podemos hacer, por ejemplo, que al encender el ordenador nos pregunte si queremos entrar en Linux 2.4.18, Linux 2.2.20, BSD, BeOS, o cualquier otro kernel o sistema. Todo esto se configura en `/etc/lilo.conf` y ejecutando después `/sbin/lilo` para aplicar los cambios.
- **Linux**: nombre del kernel creado originariamente por Linus Torvalds. “Linux” es el nombre del kernel, y no del sistema operativo, ya que éste se llama GNU. Por eso habría que hablar de GNU/Linux en vez de Linux.
- **Locales**: variables de entorno que controlan el idioma de los programas. Si no se definen, todo saldrá en inglés aunque existan versiones en otros idiomas.
- **Montar y desmontar**: proceso por el cual un sistema de archivos se hace o se deja de hacer accesible a través de un directorio del sistema de archivos actual. Por ejemplo, podemos montar un disquet en `/mnt/floppy`. Se usan las órdenes `mount` y `umount`.
- **Log**: registro. En Linux se registran muchas acciones e incidentes, normalmente en el directorio `/var/log`. Se guarda información sobre los accesos a cada servidor y las operaciones realizadas, los ingresos y salidas de cada usuario (especialmente root), y los problemas que ha habido. Un hacker tiene que conseguir borrar los logs si quiere que nadie sospeche.
- **Loopback (lo)**: interfaz de red que representa nuestro ordenador. Se le asigna la IP 127.0.0.1 (también la 0.0.0.0) y se utiliza en servicios internos.
- **PATH**: la variable `$PATH` contiene los directorios en los que se buscará cada programa que ejecutemos antes de comprobar si está en el directorio actual. Por ejemplo, cuando hacemos un `ls` lo que realmente estamos ejecutando es `/bin/ls`, porque el directorio `/bin` está en el PATH. Un hacker puede cambiar el PATH para que programas mal hechos ejecuten comandos falsificados.
- **PID**: Process ID, o número que tiene cada proceso en ejecución. Lo podemos ver con la orden `ps`. Es necesario para poder matar (finalizar) un proceso.
- **Proxy**: programa instalado en un servidor que se encarga de captar las peticiones HTTP (páginas web), bajarlas de Internet, y servir las a los usuarios. Para evitar tener que bajar la misma varias veces, la puede guardar en caché. También puede denegar el acceso a algunas páginas.
- **Root**: el administrador de un sistema Linux. Se le conoce como superusuario, tiene UID 0, y lo puede hacer absolutamente todo (nadie tiene más privilegios).

- **Setuid** : cuando se dice que un programa tiene el bit setuid activado, se ejecuta con los privilegios de su propietario. Por ejemplo, el programa `passwd`, que un usuario normal debe poder ejecutar para cambiar su contraseña, necesariamente ha de acceder a `/etc/passwd` o `/etc/shadow`, donde sólo puede escribir root. Por lo tanto, la única solución es el 'setuid root'. Hay que tener mucho cuidado con estos programas; si se encuentra un buffer overflow se pueden conseguir privilegios de root. Para más información: el propietario lo podemos cambiar con `chown usuario archivo`, el setuid lo podemos poner y quitar con `chmod a+s archivo`, y podemos ver si un programa lo está con un `ls -l` (veremos una `s` en donde normalmente aparecen las `x`).
- **Shell** : intérprete de comandos de Linux. Hay muchas shells, la mayoría muy parecidas, pero la más utilizada es `bash`. Otras son `sh`, `csch`, `tcsh`, `bsh`, `ash`, `ksh`, `rbash`, ... Cada usuario del sistema puede escoger su shell preferida; sólo hay que modificar la entrada apropiada del `/etc/passwd`
- **Superusuario** : el usuario con más poder del sistema. Se llama root, pero se pueden crear más (sólo hace falta que tengan UID 0).
- **UID** : User ID, o número de identificación de cada usuario del sistema. Se especifica en el tercer parámetro de cada registro de `/etc/passwd`
NOTA: quien tiene UID 0 es superusuario (normalmente llamado root).
- **X**: el X Windows System (implementado por el programa XFree86) es un servidor gráfico: un servidor como todos los otros al que los programas se conectan de forma local (también puede ser remota) y muestran una interfaz con botones, menús, ratón, etc. En nuestro ordenador no lo hemos instalado porque no nos hace falta (el modo gráfico es opcional, no como en otros sistemas operativos).

4.5 Licencia FDL de la GNU

Este trabajo está escrito bajo la Free Documentation License de la GNU. Es parecida a la GPL, que hace que los programas de código abierto lo tengan que continuar siendo siempre, pero aplicada a documentos. Por tanto, cualquiera puede aprovechar todo mi trabajo para mejorarlo, pero siempre tendrá que hacer que éste siga siendo público y, claro, no atribuírselo a sí mismo (ha de conservar el nombre del autor, yo, Daniel Clemente Laboreo).

La licencia FDL, necesaria en el trabajo pero que no incluyo por razones de espacio, está disponible (en castellano) en <http://es.gnu.org/Licencias/fdles.html>

La licencia, en inglés, es <http://www.gnu.org/licenses/fdl.html>

El trabajo original se puede descargar libremente de mi página, en la dirección <http://www.danielclemente.com/servidor>

5 BIBLIOGRAFÍA

- General

Serveis de xarxa amb GNU/Linux, Jordi Bort y Marc Guri. 2002.

<http://www.xtec.es/formacio/curstele/d70/>

Securing Debian Manual, Javier Fernández-Sanguino Peña. 2002.

<http://www.debian.org/doc/manuals/securing-debian-howto>

<http://www.debian.org>

- Instalación

The very verbose Debian 3.0 Installation Walkthrough, Clinton De Young.

2002. http://www.osnews.com/story.php?news_id=2016

<http://www.distrowatch.com>

- Servidor web Apache

<http://www.apache.org>

- Servidor FTP pure-ftpd

<http://www.pure-ftpd.org>

- Proxy-caché Squid

<http://www.squid-cache.org>

- PHP

Adding PHP to Apache on Linux, Ken Coar. 1999.

<http://www.linuxplanet.com/linuxplanet/tutorials/1374/1/>

<http://www.php.net>

- CGI

CGI Programming on the World Wide Web, Shishir Gundavaram. 1996.

O'Reilly. 1ª edición; disponible online en <http://www.oreilly.com/openbook/cgi/>

- DNS

DNS HOWTO, Nicolai Langfeldt, Jamie Norrish y otros. V 9.0, 2001

<http://www.linux.org/docs/ldp/howto/DNS-HOWTO.html>

- Samba

Using Samba, Robert Eckstein, David Collier-Brown, Peter Kelly. 1999.
O'Reilly. Disponible online en http://us2.samba.org/samba/oreilly/using_samba

- Firewall

Un firewall en 10' pero sabiendo lo que se hace, Guillermo Pereyra Irujo.
Gener 2003.

