A todas aquellas personas de ciencia que trabajan pensando que el futuro es una oportunidad para hacer las cosas mejor.

Agradecimientos:

Queremos agradecer la colaboración desinteresada a Rafa por sus dibujos, a Carol por su apoyo e ideas, a Mariví, Laura y Carmen por sus correcciones.

Autores:

Arturo Martín Romero	Juan José Martín Romero
Profesor de Sistemas Informáticos	Profesor de Sistemas Electrónicos
IES Rio Arba de Tauste (Zaragoza)	IES M. B. Cossio de Haro (La Rioja)

Estaremos agradecidos de recibir vuestras consultas, críticas y sugerencias.
Nuestras direcciones de correo son:

amartinr@educa.aragon.es
juanjo@cossio.net

PRÓLOGO

¿Qué calculador[1] de principios del siglo XX, hubiera imaginado los grandes avances tecnológicos que el futuro iba a deparar?. Era una época en la que cientos de personas calculadoras debían permanecer coordinadas para solucionar complejos problemas de cómputo durante largos periodos de tiempo; problemas que en la actualidad pueden realizarse en décimas de segundo con nuestros potentes ordenadores.

IBM, mediante sus equipos de tarjetas perforadas, y posteriormente John Von Neumann a través de su extraordinaria capacidad matemática, iniciaron lo que hoy se conoce como la era de las comunicaciones. Aunque su finalidad inicial[2] no fue muy loable, ahora debemos estar agradecidos por los numerosos aspectos de nuestras vidas que se ven afectados por dichos avances. Mensajería electrónica, gestores de bases de datos, imagen/sonido/video en formato digital, compra de billetes de avión o tren por Internet, consulta del estado del tiempo y las carreteras, etc. son sólo algunos ejemplos.

Todos estos servicios ofrecidos a través de Internet aparentemente son muy sencillos, y en la mayoría de los casos no percibimos la complejidad que hay detrás.

Detrás se encuentran en primer lugar los administradores de los servicios que Internet nos ofrece; y aún más atrás se encuentran los desarrolladores, implementando complejos algoritmos que nos hagan la vida más fácil. Con este libro vas a poder situarte en el lado de los administradores.

Seguramente estarás cansado de tropezar con libros relacionados con servicios de Internet que no concuerdan con las expectativas que esperabas de ellos: demasiada teoría y muy poca práctica, no queda claro como implementar el servidor, se centran reiteradamente en los mismos aspectos, no tratan ciertos temas importantes, no muestran ningún ejemplo práctico resuelto, etc.

Esta obra trata de buscar algo diferente mediante un enfoque teórico-práctico, dando un mayor peso a la implementación práctica de todos los servicios que se presentan. El objetivo de la misma es clarificar, apoyar, iluminar al lector, hacerle disfrutar y vibrar con esa emoción con la que se recibe la explicación que te produce el entendimiento de algo con lo que te has chocado varias veces y parece un muro infranqueable.

¿Acaso no te gustaría comprender en profundidad la forma en que funcionan los distintos servidores que hay en Internet? ¿Aprender a instalar y configurar un servidor de correo Webmail? ¿Ser capaz de llevar a la práctica un robusto servidor Web o FTP? ¿Asegurar tu red mediante el uso de un firewall? ¿Resolver nombres de dominio sin depender constantemente de un DNS externo? ¿Dar servicio a complejos sitios Web dinámicos? ¿Sincronizar tu red con un reloj atómico o satélite GPS? Y todo ello de una manera sencilla.

El libro esta organizado en diferentes capítulos pudiendo encontrar en cada uno de ellos una explicación del funcionamiento de los servicios más comúnmente utilizados, junto con ejemplos que permiten ponerlos en práctica.

- El capítulo 1 comienza justificando la elección del sistema operativo, que será utilizado en las distintas implementaciones prácticas que se proponen a lo largo del libro, y como configurarlo para trabajar en red.
- El capítulo 2 describe como implementar un servidor de nombres de dominio.
- En el capítulo 3 aprenderemos a crear un servidor de páginas Web mediante APACHE.
- En el capítulo 4 aprenderemos a configurar nuestro propio servidor FTP.
- El capítulo 5 esta dedicado a la creación de un servidor de correo electrónico.

[1] Cuando no existían ordenadores los cálculos eran realizados a mano. Las personas encargadas de ello se denominaban calculadores/calculadoras.

[2] Al igual que en otras ramas científicas, esta prospero notablemente a comienzos del siglo pasado gracias a las investigaciones militares asociadas a la segunda guerra mundial (bomba atómica).

- En el capítulo 6 es tratado el problema de la seguridad a través de los firewall. Aprenderás a determinar las reglas que harán segura una red.
- El capítulo 7 y último, muestra lo que es un servidor de tiempo y como implementarlo para que de servicio a una red.

Además de los anteriores puntos clave, en el libro se desarrollan de forma paralela otros muchos conceptos, que permiten que el texto pueda ser utilizado en cualquier tipo de curso que tenga como fin el estudio de las redes informáticas y su administración. Aunque el temario desarrollado coincide en mayor medida con algunos ciclos formativos es importante señalar que no esta destinado a ninguno en particular; sus contenidos se pueden adecuar a varios de los ciclos ofertados por el Ministerio de Educación y Ciencia, como por ejemplo:

- C.F.G.M. de Explotación de sistemas informáticos.
- C.F.G.S. de Administración de Sistemas Informáticos.
- C.F.G.S. de Desarrollo de Aplicaciones Informáticas.
- C.F.G.S. de Sistemas de Telecomunicación e Informáticos.

Si quieres aprender practicando, este es tu libro. Por favor, disfrutad y emocionaros.

Índice general

Capítulo 1

CONOCIMIENTOS BÁSICOS SOBRE REDES EN GNU/Linux

1.1. GNU/Linux o Microsoft Windows

A causa de la fuerte publicidad y el gran monopolio que Microsoft trata de ejercer con sus sistemas operativos Windows[1], para la gran mayoría de los usuarios informáticos GNU/Linux es uno de los grandes desconocidos. Es un sistema operativo de aspecto muy similar al archiconocido entorno Windows, pero con grandes diferencias en muchos sentidos respecto a este último, que hacen de él el sistema operativo de futuro. Algunas de las características más relevantes de GNU/Linux:

1. Es Software Libre. La filosofía del software libre fue iniciada por Richard Stallman en los años 80. En esos años Stallman trabajaba como programador de una empresa americana, y se encontraba con el habitual inconveniente de que tras haber adquirido un paquete software no tenía ninguna posibilidad de ajustarlo a sus necesidades, ya que el código fuente no estaba disponible. Ante esta situación, Stallman comenzó a reivindicar que las empresas de software cedierán el código para poder cambiar los defectos y/o limitaciones que este presentaba. Había pagado por el software ¿por qué no podía adecuarlo a lo que realmente necesitaba?

   Con esta idea Richard Stallman pasa a liderar el proyecto GNU, cuyas siglas significan Gnu No es Unix. El nombre da a entender su desacuerdo con la filosofía de las empresas desarrolladoras de software del sistema operativo que en esos años estaba más enraizado, Unix. Tras la creación del proyecto GNU, y con la colaboración de Linus Torvalds que había desarrollado el kernel de un sistema operativo llamado Linux, surge GNU/Linux, que daría soporte a todo el software de código abierto que defendía Richard Stallman.

   La idea inicial acabo derivando en un conjunto de libertades de las que puede hacer uso cualquier usuario de GNU/Linux, y que como comprobarás serían impensables con Microsoft Windows. Estas son:

   a) `Libertad 0`: Tienes la libertad de usar los programas con cualquier propósito.

   b) `Libertad 1`: Podemos estudiar cómo funciona el programa, y adaptarlo a las necesidades particulares de cada usuario. Puesto que el software es libre, el código fuente del programa esta disponible.

   c) `Libertad 2`: Se pueden hacer tantas copias como uno quiera del software y distribuirlas a quien se desee.[2]

[1] Actualmente, en muchos casos y sin desearlo, al aquirir un equipo informático nos lo venden con Windows Microsoft instalado (no es gratuito) sin opción alguna.

[2] Recuerda que las copias que haces de Windows son ilegales.

d) `Libertad 3`: Puedes mejorar el programa (libertades 0 y 1) y hacer públicas las mejoras al resto de usuarios (libertad 2). Esto permite una mejora continua del software desarollado por la Comunidad GNU/Linux.

Es importante resaltar algo que la mayoría de la gente confunde. Esta muy generalizada la idea de que Software Libre equivale a decir software gratuito, y eso no es cierto. Las libertades no hablan en ningún momento del coste del software, y por tanto es posible que no sea gratuito. En cualquier caso, cierto es que es un poco absurdo pagar por algo que se distribuye libremente (libertad 2). Por lo general, son las empresas las que pagan por GNU/Linux; ya no por su código, sino por el soporte que la empresa suministradora ofrece (servicio de mantenimiento y formación).

2. Como consecuencia de la libre distribución del software (Libertad 2), se deriva una ventaja indiscutible, imposible de encontrar en otros sistemas operativos: El software esta disponible en distintos sitios de la red de Internet (sitios FTP y HTTP), pudiendo ser descargado por cualquier usuario y en cualquier momento. Entre las posibles ventajas destacaremos dos,

   a) Las imágenes (.iso) de los CD de instalación de GNU/Linux se encuentran disponibles en abundantes sitios de la red, lo que nos permite una instalación remota de cualquier distribución de GNU/Linux sin requerir los CD/DVD, siendo necesario únicamente una conexión a Internet.
   Para llevar a cabo este tipo de instalación, ya sea desde un sitio FTP o HTTP, será necesario en primer lugar crear un disquete o CD de arranque para la instalación.

   b) Una buena parte del software de aplicación y de desarrollo que corre bajo este sistema operativo, sigue la filosofía de software libre[3]. Es posible obtener (mediante descarga HTTP, FTP, ...) de forma libre software para la reproducción/grabación de DVD, herramientas ofimáticas, de gestión de bases de datos, para servidores (WEB, FTP, PDC, ...), software para el desarrollo de nuevas aplicaciones, y mucho más.
   Para la descarga y utilización de todo este software, puedes conectarte directamente al sitio de Internet donde se encuentra y descargarlo, o puedes hacerlo de forma desatendida haciendo uso de repositorios. Un repositorio no es más que la ubicación lógica o dirección donde puede encontrarse gran cantidad de software: unidades de CD-ROM, DVD, sitios FTP, sitios HTTP, o cualquier otra ubicación donde puede albergarse software.
   Esto nos permite disponer de cualquier aplicación que no tengamos instalada, sin necesidad de ir acompañados de CD/DVD de instalación. Lo único necesario es una conexión a Internet.

3. Otra característica deseable en todo software, y que en este caso cumple GNU/Linux, es la portabilidad. Esto garantiza que pueda hacerse uso de este, independientemente del hardware que este por debajo. Es decir, lo ideal sería poder disponer de un software que presentase las mismas características independientemente de la máquina sobre la cual se este trabajando. Esta portabilidad se ha conseguido gracias a que GNU/Linux toma las ideas del software de sistemas UNIX, sistema operativo desarrollado entre otros por Ken Thompson y Dennis Ritchie, quienes se esforzaron en crear un sistema operativo lo más portable posible, y por tanto, lo más independiente posible de los fabricantes de hardware (microprocesador del equipo).

   Por su parte, los sistemas operativos de Microsoft Windows (2000, XP y 2003) operan sólo bajo la arquitectura x86 (Intel, y los clones de éste como AMD). GNU/Linux esta disponible para una gran variedad de arquitecturas hardware, entre las que podrían recalcarse: x86, PowerPC, Sparc, UltraSparc, Alpha, PA-RISC, o mainframe (IBM). En la actualidad, grandes empresas del sector informático están apostando abiertamente por GNU/Linux.

[3] No debes pensar que GNU/Linux sólamente corre software libre; muchas de las empresas que desarrollan software privativo para Microsoft Windows también lo hacen para GNU/Linux.

4. Otra característica importante es que para obtener el mismo rendimiento en una máquina haciendo uso de GNU/Linux y de Windows, a nivel de servidor, con GNU/Linux los requerimientos hardware son inferiores. Es decir, Windows necesita unos requisitos hardware más potentes: más memoria RAM, periféricos de mayor velocidad, ... resultando un sistema operativo Windows más `pesado` que un GNU/Linux.

5. Windows presenta unos `agujeros de seguridad` más severos. Se ha comprobado que prácticamente la totalidad de los ataques malintencionados que se producen en Internet, son llevados a cabo sobre máquinas sobre las que corre el sistema operativo Microsoft Windows, o alguno de sus paquetes propietarios (Internet Information Server, Internet Explorer, ...). Para tratar de solventar estos problemas, Windows hace uso de `parches` (Service Pack), consiguiendo únicamente que el sistema operativo Microsoft Windows sea todavía más pesado, y a su vez más ineficiente.

   Relacionado igualmente con la seguridad, podría destacarse que en GNU/Linux existe la disponibilidad de un paquete software que se integra en su kernel que nos permite filtrar las comunicaciones: iptables. Aunque en Windows existen también herramientas propias, como de terceros, que desempeñan una función similar, sigue optándose por GNU/Linux para despeñar funciones de firewall en entornos de red.

6. A diferencia de Windows, GNU/Linux es un sistema operativo que está formado por una estructura en diferentes niveles muy sencilla de comprender.

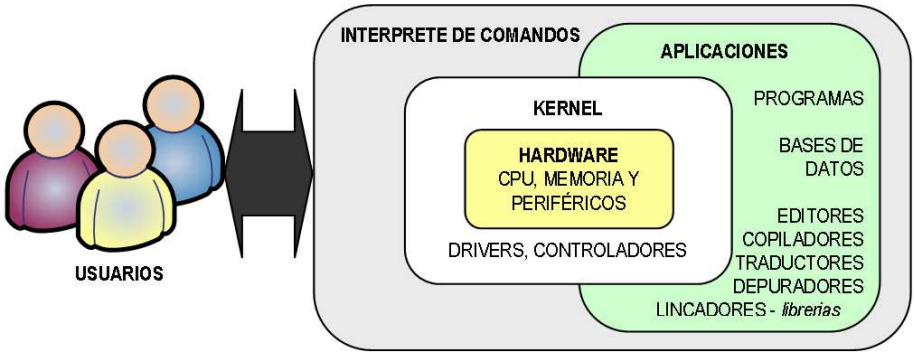


Figura 1.1: Arquitectura del sistema operativo GNU/Linux.

Tal como se puede observar en la figura 1.1, en GNU/Linux se distinguen principalmente dos niveles o capas (el kernel y el intérprete de comandos) que separan a los usuarios del hardware que compone el sistema informático, y que permiten al usuario olvidar la complejidad de la comunicación con éste. En concreto, de las dos capas, la parte del sistema operativo GNU/Linux que se encuentra en contacto directo con el hardware es el kernel o núcleo del sistema.

Este está formado por un conjunto de controladores[4] capaces de interactuar de forma directa con el hardware, y que le ordenan cuales son las tareas que tiene que llevar a cabo (servicios, comandos de usuario, ...).

En el exterior encontramos la capa del sistema operativo GNU/Linux que se encuentra en contacto directo con el usuario: el intérprete de comandos, también denominado la `SHELL` del sistema. La función esencial de la `SHELL` es estar a la escucha de las órdenes que realizan los usuarios del sistema e interpretarlas. Después, son transmitidas al sistema operativo, que las traduce al lenguaje máquina comprensible por el hardware. Por tanto, la función del intérprete de comandos es permitir al usuario ejecutar tareas (reproducir un CD de música, grabar un

[4]Los controladores son programas software a los que también de denomina `drivers`.

DVD, realizar un backup del sistema de ficheros sobre un disco externo, ...) mediante el uso de un lenguaje de alto nivel (lenguaje próximo al lenguaje humano).

El intérprete de comandos tiene una importancia vital en GNU/Linux, ya que nos permite, mediante una sintaxis muy sencilla, la creación poderosos programas para la ejecución de tareas programadas denominados Shell-Scripts. Ejemplos de ello veremos a lo largo de todo el libro.

Entre las dos capas anteriores, kernel e intérprete de comandos, se encuentran el resto de software que es instalado por el usuario y que permiten a este explotar el sistema informático con distintas finalidades: desarrollo de aplicaciones, gestión de bases de datos, ... En el caso de que se haga un uso muy frecuente de alguna de estas aplicaciones, es posible incluso hacer que forme parte del kernel, provocando que su ejecución sea más rápida evitándonos un paso intermedio en el momento de tratar con el hardware. Para ello será necesaria una re-compilación de su núcleo (kernel). Esto hace de GNU/Linux un sistema operativo muy versátil, capaz de adaptar su núcleo o kernel al hardware disponible. Es posible encontrarlo en sistemas de información muy variados, desde sofisticadas agendas electrónicas hasta complejos mainframes.

También sería importante mencionar que el kernel de GNU/Linux está desarrollado casi en su totalidad mediante el conocido lenguaje de programación de alto nivel `C`. Creado por los desarrolladores de UNIX, es uno de los lenguajes de programación más extendidos en el mundo, y con una sintaxis muy sencilla de comprender, lo que facilita su lectura y modificación. Esta es una de las razones fundamentales por las que GNU/Linux es un sistema tan portable.

7. GNU/Linux es un sistema operativo que presenta dos tipos de interfaz para permitir al usuario interactuar con la máquina: una interfaz gráfica (GUI) y una interfaz de línea de comandos (LUI/CUI). La primera presenta las mismas prestaciones que la de Microsoft Windows, con su correspondiente escritorio, accesos directos, barra de herramientas, explorador de archivos, ... En cambio la segunda interfaz, la LUI (Interfaz de Línea de comandos de Usuario) es mucho más completa que la que aporta Windows (el emulador MS-DOS o símbolo del sistema), ofreciendo la opción de poder realizar cualquier tarea del sistema.

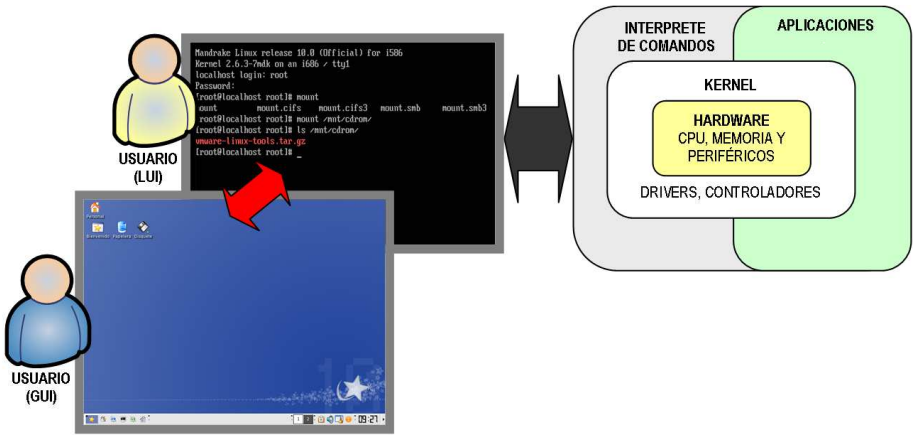


Figura 1.2: GNU/Linux ofrece interfaces de usuario gráfica y de línea de comandos.

Tal como se puede advertir en la figura 1.3, la forma rápida y eficaz de ordenar tareas al equipo informático, es haciendo uso de la línea de comandos (LUI), ya que es la forma más directa de ponerse en contacto con la parte del sistema operativo encargado de gestionarlas: el intérprete de comandos.

Cuando estamos haciendo uso de la interfaz gráfica de usuario (GUI), también llamada X-Window, la ejecución de comandos es más lenta. Esto se debe a que se trata de una aplicación

que funciona sobre el sistema operativo, es decir, se añade una capa más entre nuestra interfaz y el hardware.

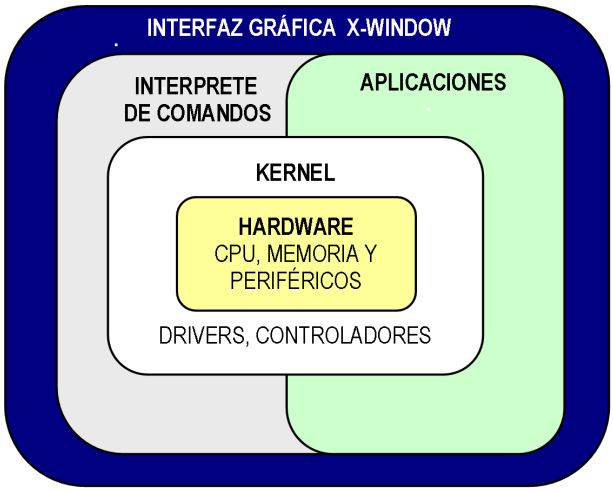


Figura 1.3: La interfaz grafica en GNU/Linux: X Window.

Este es el motivo por el cual es preferible trabajar desde la interfaz LUI, a pesar de que sea menos atractiva, ya que garantiza una eficiencia mucho mayor, y con un menor consumo de recursos que si hiciésemos uso de un entorno de ventanas. Este aspecto hay que tenerlo en cuenta en el momento de seleccionar el sistema operativo que debe instalarse en un equipo que va a funcionar como servidor (servidor WEB, FTP o de correo electrónico). En un servidor interesa que el sistema operativo ponga todo su esfuerzo en dar estos servicios, y no pierda su potencia en otros aspectos, como por ejemplo la interfaz gráfica. Por este motivo, la mayoría de los servidores de Internet son máquinas sobre las cuales esta corriendo un sistema operativo UNIX o GNU/Linux, ya que ambos pueden llevar a cabo las funciones encomendadas sin la necesidad de dicha interfaz.

De la arquitectura software comentada antes debes percibir la independencia de funcionamiento entre sistema operativo y entorno gráfico. Si por cualquier motivo la interfaz gráfica se bloquease, o no nos permitiera seguir trabajando, podríamos prescindir de ella y continuar desde la interfaz de comandos. Esto no puede destacarse en los sistemas operativos Microsoft Windows, donde al contrario que en UNIX/Linux, la interfaz de comandos esta supeditada a la interfaz gráfica, es decir, que en caso de quedársenos *colgado* el entorno gráfico de Windows, nos veríamos obligados a reiniciar la máquina sin la posibilidad de poder seguir trabajando.

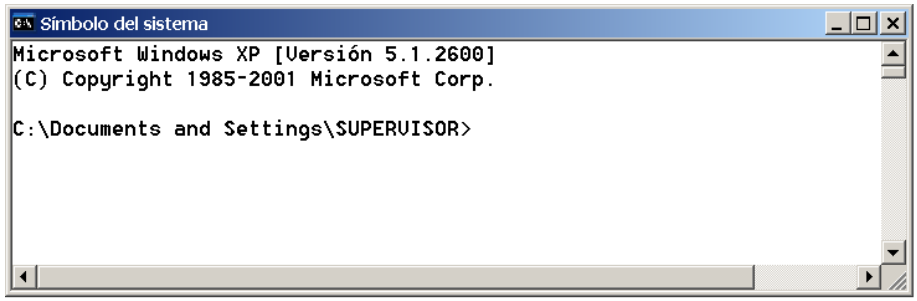


Figura 1.4: Interfaz LUI de Microsoft Windows.

En cuanto a la interfaz LUI de Microsoft Windows, cabría señalar que aunque cada nueva versión de Windows hace un mayor esfuerzo por darle más potencia, sigue estando lejos de las prestaciones ofrecidas por su equivalente en GNU/Linux. A modo de ejemplo, podría

señalarse el comando `dsadd`, utilizado para la creación de usuarios, grupos, equipos o unidades organizativas dentro de un dominio de Microsoft Windows. Apareció como nueva[5] herramienta de administración para la familia de sistemas operativos Microsoft Windows 2003 Server.

A través de los diferentes ejercicios que aparecen en el libro se podrá descubrir la potencia que aporta al usuario el disponer de esta LUI, y la facilidad en la programación de tareas que se nos ofrece mediante el uso de Shell-Scripts.

En el caso de Microsoft Windows, también se pueden realizar estos scripts, haciendo uso de lenguajes de programación como son JavaScript o VisualScript, aunque adolecen de poca potencia en comparación a la ofrecida por UNIX/Linux.

8. Gracias a que UNIX/Linux son sistemas operativos de naturaleza multiusuario, permiten tener abiertas varias sesiones simultáneas, lo que hace posible poder trabajar al mismo tiempo desde una interfaz gráfica y desde una terminal. En concreto se puede trabajar con varias sesiones abiertas simultáneas en modo comando, y por defecto, una en modo gráfico. Para pasar de una a otra, tan sólo es necesario pulsar la combinación de teclas `CONTROL+ALT+F`x (siendo x, 1, 2, 3, ... hasta 8), donde por convenio se accede a la interfaz gráfica desde `CONTROL+ALT+F7`, o `CONTROL+ALT+F8`, y el resto de combinaciones en modo comando. En el caso de no disponer de una sesión gráfica abierta (modo de arranque `init 3`), desde cualquier interfaz en modo comando se puede abrir ejecutando el comando `startx`.

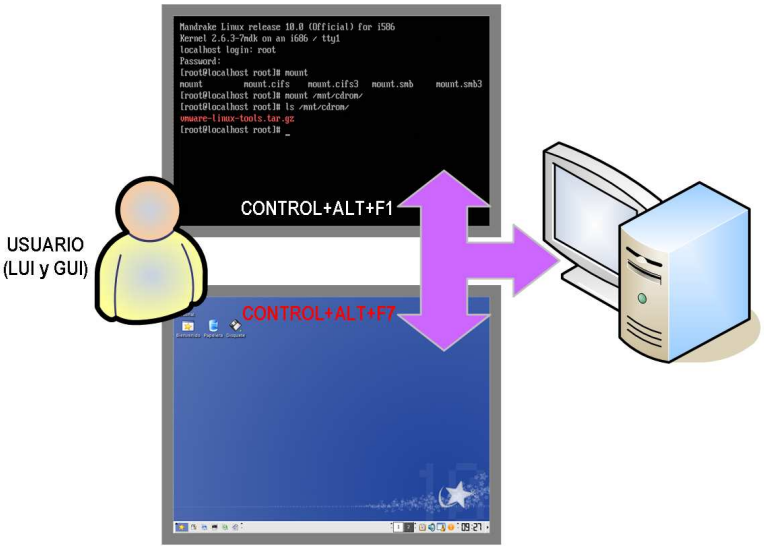


Figura 1.5: GNU/Linux nos permite abrir simultáneamente varias sesiones.

9. En el mundo del software de sistemas[6], puede distinguirse entre sistemas operativos para equipos servidores, y sistemas operativos para equipos clientes. Los hay que están pensados para funcionar solamente bajo servidores, como es el caso de Novell Netware, y sistemas operativos que están pensados para funcionar solamente bajo equipos cliente, como es el caso de Windows 2000 Profesional.

   Microsoft Windows, posee diferentes tipos de sistemas operativos dependiendo del equipo donde va a ser instalado: Windows 2000 Profesional (o Windows XP) para los equipos clientes, y Windows 2000 Server (o Windows 2003) para equipos servidores.

   GNU/Linux es indiferente del equipo donde vaya a ser instalado. Es decir, GNU/Linux es un sistema operativo muy versátil en este sentido, ya que es capaz de acomodarse a las

[5] En GNU/Linux el comando análogo, `useradd`, es uno de los más básicos y antiguos.

[6] El software se puede clasificar en software de sistemas (sistemas operativos), software de aplicación (aplicaciones) y software de desarrollo (entornos de desarrollo de aplicaciones).

funcionalidades del equipo, sea cliente o servidor. Por esta razón, se dice que GNU/Linux es un sistema operativo cliente-servidor.

10. Al contrario que los sistemas operativos de Microsoft, GNU/Linux es un sistema operativo con capacidad para que varios usuarios exploten el sistema de manera simultánea: sistema operativo multiusuario.

    Hoy en día el termino multiusuario esta desvirtuado, ya que se utiliza en muchas situaciones que estrictamente no lo son. Así, un equipo que actúe como servidor de bases de datos, se dice erróneamente, que es multiusuario porque varios usuarios, sobre diferentes sesiones, están realizando múltiples consultas a la base de datos de manera simultánea. Incluso se ha llegado a llamar sistemas multiusuario a aquellos sistemas operativos donde han sido creadas diferentes cuentas de usuario, distinguiéndose entre sí a través del nombre (login) y contraseña (password) introducidos durante el inicio de sesión, pero sin poder trabajar varios simultáneamente.

    Siendo estrictos, un sistema operativo multiusuario, es aquel que permite que un equipo informático, y por tanto, todo su hardware (memoria RAM, disco/s duro/s, unidad/es CD-ROM, impresora/s, conexión a Internet, ...) y todo su software (todo tipo de programas y aplicaciones en el instaladas, incluyendo al sistema operativo) puedan ser explotados simultáneamente por diferentes usuarios. Todo ello está implícito en un sistema que funcione bajo GNU/Linux.

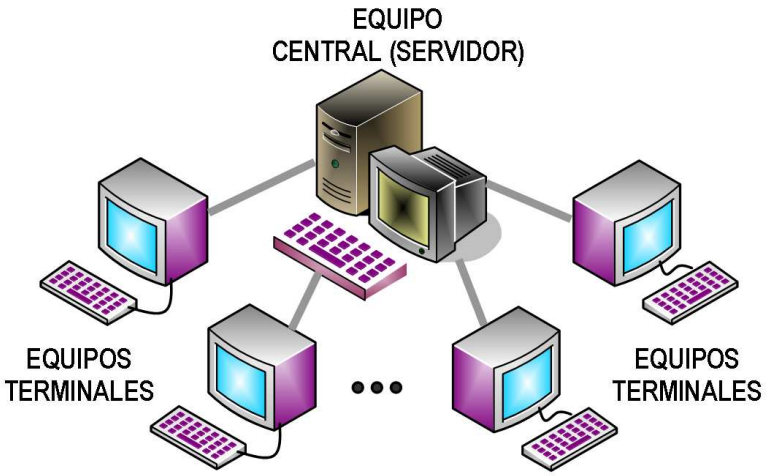


Figura 1.6: GNU/Linux es un sistema multiusuario.

Es cierto, que en sistemas operativos como Windows 2000/XP/2003 existen herramientas software como pueden ser `Windows Terminal Server`, que permite algo similar, pero en realidad es diferente ya que el equipo terminal desde el cual se realiza la conexión a la máquina central, requiere de su propio sistema operativo. En GNU/Linux existe otra herramienta similar a la que presenta Windows, llamada `Linux Terminar Server` o `LTSP`, garantizando una mayor eficiencia del sistema global.

11. Una de las funciones más importantes de un sistema operativo, es la capacidad de gestionar eficientemente la información almacenada. Se debe garantizar que el espacio de almacenamiento sea aprovechado de la mejor forma posible, al mismo tiempo que se nos permite acceder a tal información cuando sea requerido de la forma más rápida. Es decir, de poco nos sirve que el sistema operativo tenga una estrategia óptima para no desperdiciar nada de espacio del disco, si tras una solicitud de datos, emplea demasiado tiempo en recuperarlos.

    En el caso de los sistemas operativos de Microsoft Windows, se hace uso de las estrategias (sistema de gestión de ficheros) denominadas FAT16/32 y NTFS. En cambio GNU/Linux hace uso de un sistema de gestión de ficheros propio denominado `ext3`.

Es importante observar que mientras, Microsoft Windows se aferra en utilizar únicamente sus sistemas de ficheros propietarios, GNU/Linux se abre al resto de opciones o estrategias de gestión, presentando la capacidad de soportar tanto los suyos propios, como los utilizados por Microsoft Windows.

Concretamente GNU/Linux soporta FAT16/32, NTFS[7], ext, ext2, ext3, vfat, xia, minix umsdos, iso9660 y muchos más, entre los cuales cabría señalar por su alto rendimiento JFS, ReiserFS y XFS.

Todo esto permite que GNU/Linux pueda ser instalado en la misma máquina con otros sistemas operativos[8] y compartir información con estos sin ningún tipo de problemas. Esto mismo no se puede decir de los sistemas operativos Microsoft Windows.

12. Con respecto al software de aplicación ocurre algo similar a lo que sucede con los sistemas de gestión de ficheros. GNU/Linux esta esforzándose para que aplicaciones realizadas para entornos Windows puedan ser portables a entornos GNU/Linux. Desde 1993 está siendo desarrollado el proyecto `WINE` (Wine Is Not an Emulator, Wine no es un Emulador de Windows) con la pretensión de que sea posible la ejecución de aplicaciones Windows desde un sistema operativo GNU/Linux. Tal como indican sus siglas, WINE no es un simple emulador de Windows, sino que en realidad es una implementación de la API (Application Programming Interface, Interfaz de Programación de Aplicaciones) de Windows. Con ello se pretende hacer frente a la gran dependencia que ciertos usuarios tienen respecto a determinadas aplicaciones que corren bajo Windows, como pueden ser las Microsoft Office, o Adobe Photoshop. Se desea que puedan afrontar el cambio de sistema operativo, de Windows a GNU/Linux, sin que esto suponga ningún esfuerzo.

13. Siguiendo con las compatibilidades e incompatibilidades entre los distintos sistemas operativos, cabría señalar la facilidad con la que una máquina con sistema operativo GNU/Linux puede integrarse en una red gestionada por un sistema operativo Microsoft Windows, y aprovecharse de los servicios ofrecidos por esta (unidades de almacenamiento compartidas, impresoras compartidas, ...).

    Para ello disponemos de un paquete sofware llamado `samba`, que permite infiltrar a equipos con sistema GNU/Linux en redes Microsoft, pasando estos desapercibidos. De esta forma, un equipo GNU/Linux puede desempeañar las funciones de servidor en una red híbrida formada por equipos clientes bajo diferentes sitemas operativos.

14. Además de todas las características anteriores, señalar que GNU/Linux es un sistema operativo que está especialmente pensado para el trabajo en entornos de red, permitiéndonos ofrecer multitud de servicios: FTP, HTTP, SMTP, NTP, SNMP, POP, IMAP, ... Y todo ello sin coste alguno.

15. Por último, y no menos importante, cabría resaltar la abundante documentación que hay disponible de forma libre para el manejo, explotación, gestión y administración del sistema GNU/Linux. Como consecuencia de las cuatro libertades comentadas al comienzo del presente apartado, junto con el software, es posible obtener multitud de tutoriales, manuales y explicaciones muy útiles que ayudan al administrador. De todas estas ayudas cabría señalar los `HOWTO` (`Cómo`), pequeños tutoriales, normalmente disponibles en formato HTML o postscript, que nos ayudan a configurar y explotar variados aspectos de GNU/Linux.

    En definitiva, GNU/Linux es un sistema operativo muy versátil, robusto y estable. Su filosofía de software libre y el hecho de que existen multitud de personas en el mundo que trabajan desinteresadamente para mejorarlo, hacen de él un sistema operativo con un futuro muy prometedor.

[7] NTFS sólo en modo lectura.

[8] En una máquina puede haber más de un sistema operativo instalado. El gestor de arranque es el encargado de ofrecernos con cual deseamos trabajar.

Consideramos que se haría bien, si desde las universidades, los institutos de enseñanza secundaria y el resto de instituciones públicas se hiciera un mayor esfuerzo para su divulgación.

Por todas las razones expuestas anteriormente, el sistema operativo escogido para la implementación de los distintos servicios de Internet que se proponen en el libro será GNU/Linux. Existen multitud de distribuciones[9], aquí se ha optado por Mandriva porque es una de las más fáciles de instalar y administrar. Esto la hace muy recomendable para los que acaban de entrar en el mundo GNU/Linux. En adelante supondremos que en tu equipo tienes instalada esta distribución obviando el proceso de instalación.

1.2. Contenido del capítulo

Para que una máquina trabaje como servidor debe pertenecer a una red de ordenadores. Este capítulo está enfocado a que aprendas a configurar tu ordenador en red.

En primer lugar veremos como se puede configurar la red desde la línea de comando.

La configuración por línea de comandos no es perpertua, ya que al arrancar el sistema operativo carga la configuración de red desde unos archivos. La modificación de los mismos será nuestro siguiente objetivo.

Por último, veremos como se pueden realizar estas tareas desde el entorno de ventanas; de forma similar a como se haría en Windows.

1.3. Configuración previa de la red

Para hacer las pruebas y ejercicios propuestos en el libro deberás configurar tus interfaces de red. Disponemos de varias opciones:

1. Hacer uso de los comandos `ifconfig` y `route`. El primero de ellos nos permite configurar las interfaces de red de nuestro equipo, y el segundo programar las reglas de enrutamiento que le permitirán comunicar a nuestro equipo con el resto.

2. Modificar mediante un editor de textos (vi, kate, kwrite, ...) los ficheros de configuración de red: `/etc/sysconfig/network` y `/etc/sysconfig/network-scripts/`.

3. Configurar la red mediante un asistente gráfico. En el caso de Mandriva este asistente se invoca mediante el comando `drakconnect`.

1.3.1. Configuración de la red por medio de `ifconfig` y `route`

El comando `ifconfig`, nos permite tanto establecer como comprobar la configuración de las interfaces de red de las que dispone el equipo informático. En el caso de ejecutar dicho comando sin ir acompañado de parámetros nos proporcionará la información relativa a todas las interfaces de red, y en caso de ir acompañado del identificador de una de las interfaces de red (por ejemplo, `ifconfig eth0`), nos informará únicamente de la configuración de la tarjeta especificada. Por otro lado, en el caso de adjuntar dirección IP, máscara de red/subred y dirección de broadcast, estas serán asignadas a la interfaz especificada. La sintaxis es,

```
[root@linux]#  ifconfig ‘interfaz red’ ‘dirección IP’ netmask ‘mascara de
                        red/subred’ broadcast ‘dirección de broadcast’
```

A continuación se hace un pequeña descripción de este comando.

[9] Algunas de las posibles distribuiciones GNU/Linux son: Suse, Debian, Fedora, Mandriva, Ubuntu, ...

ifconfig *'interfaz de red' 'direccion IP / parámetros'*

- **'interfaz red'**

Un equipo informático puede disponer de una o varias tarjetas de red. Para distinguirlas entre sí, y así indicar a cuál de ellas se quiere hacer referencia, se hace uso de un identificador, `eth`x, siendo x el número de interfaz ó tarjeta de red. Este número es el cero para la primera interfaz y se va incrementando a medida que se agregan nuevas interfaces al equipo informático. Por tanto, si nuestro equipo posee una tarjeta de red, para identificarla deberemos hacer alusión a `eth0`, y si insertamos una nueva tarjeta de red esta pasará a ser `eth1`. De igual forma, a una misma interfaz de red se les puede asignar más de una dirección IP haciendo uso del identificador `eth`x`:`y, siendo y un número natural. Posteriormente se muestras varios ejemplos.

- **'dirección IP'**

Si tras el identificador de la interfaz de red, `eth`x, colocamos una dirección IP, esta será asignada a dicha interfaz. En caso de no indicarse máscara de red ó subred, Linux analizará la primera cifra de la dirección IP, determinará la clase de dicha dirección, y le aplicará la máscara y dirección de broadcast por defecto. Por ejemplo, al ejecutar

```
[root@linux]#  ifconfig eth0 192.168.121.15
[root@linux]#  ifconfig eth0:1 192.168.121.16
```

Linux configurará la interfaz eth0 con dirección ip `192.168.121.15`, dirección de broadcast `192.168.121.255`, y máscara de red `255.255.255.0`, al advertir que el `192` (1^{er} octeto de la dirección IP) se corresponde con una `clase C`.

Si deseamos formar subredes será necesario especificar por nuestra parte tanto la máscara de subred como la dirección de broadcast de la subred. En ese caso se hará uso de los siguientes opciones: `netmask` y `broadcast`:

```
[root@linux]#  ifconfig 'interfazEthX' 'direccionIP'
                        netmask 'mascaraRed' broadcast 'dirBroadcast'
```

Por ejemplo, en caso de hacer uso de tres bits para formar subredes y los 5 restantes para identificar equipos y dispositivos dentro de la subred escribiríamos:

```
[root@linux]#  ifconfig eth0 192.168.121.48
                        netmask 255.255.255.224 broadcast 192.168.121.63
```

- **'Parámetros'**

UP/DOWN $\longrightarrow$ Mediante este parámetro podemos habilitar/inhabilitar la interfaz de red. En el caso de inhabilitar una interfaz de red:

```
[root@linux]# ifconfig eth0 down
```

Para volver a *levantar* la interfaz de red, una vez inhabilitada no es suficiente con `up`, sino que deberemos volver a asignar una dirección IP, máscara, ...

MTU ⟶ Permite establecer la cantidad de bytes que se pueden enviar como mucho en un único paquete IP. Es lo que se denomina la `Unidad Máxima de Transferencia`. Por defecto vale 1500. En caso de desear cambiar esta cantidad, por ejemplo, a 1800 bytes:

```
[root@linux]# ifconfig eth0 mtu 1800
```

METRIC ⟶ Si nuestro equipo hace funciones de enrutador, y además este es dinámico, estableceremos el coste o métrica de la ruta a través de este parámetro. Esta métrica suele ser utilizada por el `Protocolo de Información de Enrutamiento (RIP)` y representa el máximo número de routers que se pueden recorrer desde un origen a un destino. Por ejemplo, para establecer una métrica de 10 escribiríamos,

```
[root@linux]# ifconfig eth0 metric 10
```

Por su parte, el comando `route` nos va a permitir establecer las reglas de enrutamiento que serán consultadas por nuestro equipo GNU/Linux para decidir por que interfaz de red enviar un paquete TCP/IP con la finalidad de acabe alcanzando su destino. Para conocer dichas reglas de enrutamiento será necesario ejecutar el comando:

```
[root@linux]# route -n
```

Al ejecutar el comando anterior, podremos advertir que por el mero hecho de haber configurado mediante `ifconfig` las distintas interfaces de red de nuestro equipo, GNU/Linux dispone de una serie de reglas que le permitirán comunicarse con los equipos que forman parte de las distintas redes lógicas a las que pertenecen las direcciones IP asignadas a las interfaces de red.

Por ejemplo, si disponemos de un equipo con tres interfaces de red que le permiten comunicarse con las redes lógicas `192.168.1.0/24`, `12.0.0.0/8` y `172.20.0.0/16`, y deseamos asignarle la dirección IP más baja de cada una de las redes anteriores deberíamos escribir (ver la figura 1.7),

```
[root@linux]#  ifconfig eth0 192.168.1.1
[root@linux]#  ifconfig eth1 12.0.0.1
[root@linux]#  ifconfig eth2 172.20.0.1
```

Automáticamente se habrá configurado la siguiente tabla de enrutamiento en nuestro equipo:

```
[root@linux]# route -n
```

```
Kernel IP routing table
Destination  Gateway     Genmask       Flags  Metric  Ref  Use  Iface
192.168.1.0  0.0.0.0   255.255.255.0     U       0      0    0   eth0
12.0.0.0     0.0.0.0     255.0.0.0       U       0      0    0   eth1
172.20.0.0   0.0.0.0    255.255.0.0      U       0      0    0   eth2
```

La tabla anterior nos advierte a través de su primera columna que destinos son alcanzables por la máquina. Concretamente, cuando desde nuestro equipo deseamos mandar un paquete TCP/IP a un equipo conectado en red se le aplica a la dirección IP de destino la máscara que se encuentra en la tercera columna (columna `genmask`), y si tras aplicarla, se obtiene como resultado la dirección IP de la red lógica de la columna `destination`, dicho paquete es enviado sin necesidad de `gateway`

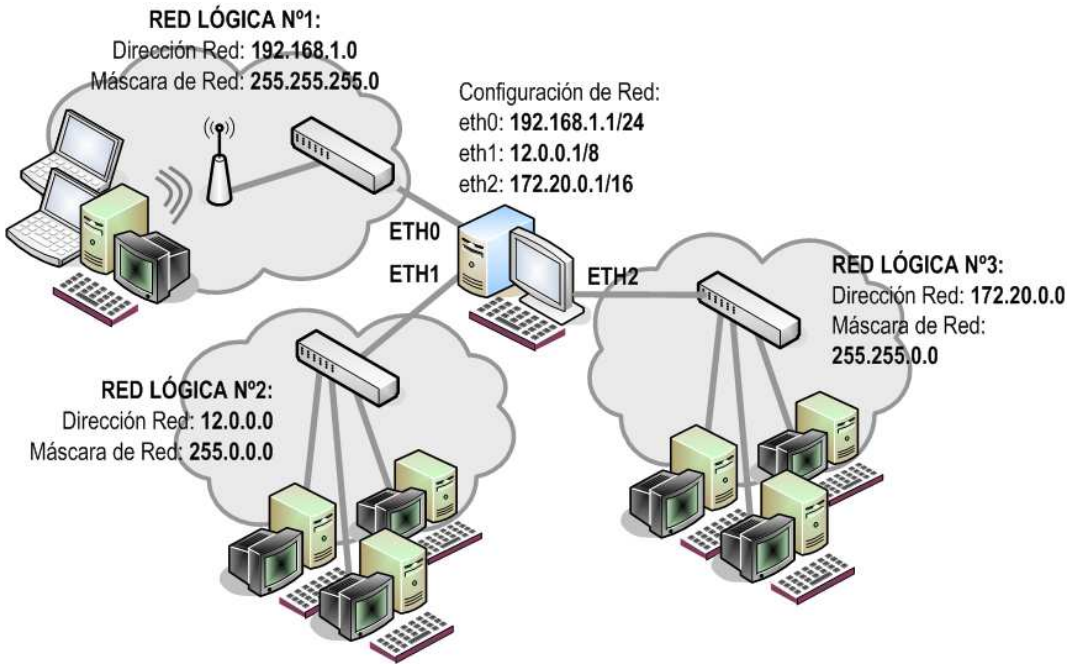


Figura 1.7: Estructura de la red expuesta en el ejemplo.

(columna 2) a su destinatario a través de la interfaz de red que se especifique en la columna `Iface` (columna 8).

Continuemos con el ejemplo. A continuación vamos a ver como quedaría la tabla de enrutamiento en el caso de que a una misma interfaz de red le asignásemos varias direcciones IP, tal como se muestra en la figura 1.8. En primer lugar asignamos las direcciones,

```
[root@linux]#  ifconfig eth0:1 192.168.2.1
[root@linux]#  ifconfig eth0:2 172.16.0.1
[root@linux]#  ifconfig eth1:1 172.25.0.1
```

En la figura 1.8 se puede ver cual sería la estructura de la red descrita en el ejemplo. La ejecución del comando `route`, proporcionaría:

```
[root@linux]# route -n

Kernel IP routing table
Destination  Gateway     Genmask       Flags  Metric  Ref  Use  Iface
192.168.1.0  0.0.0.0   255.255.255.0     U       0      0    0    eth0
12.0.0.0     0.0.0.0     255.0.0.0       U       0      0    0    eth1
172.20.0.0   0.0.0.0    255.255.0.0      U       0      0    0    eth2
192.168.2.0  0.0.0.0   255.255.255.0     U       0      0    0    eth0
172.16.0.0   0.0.0.0    255.255.0.0      U       0      0    0    eth0
172.25.0.0   0.0.0.0    255.255.0.0      U       0      0    0    eth1
```

Al intentar comunicar este equipo con otro se comprobarán una a una cada una de las líneas contenidas en la tabla de enrutamiento. Tras aplicar la correspondiente mascara (columna `genmask`), si el resultado no coincide con la dirección IP de `destination`, el destino se supondrá inalcanzable, y el paquete no saldrá del equipo. Esto significa que cualquier equipo que tenga asignada una dirección IP no perteneciente a las redes lógicas[10] que nos aparezcan allí será inalcanzable[11].

[10] Podría suceder quer en lugar de una dirección de red apareciese en el listado la dirección de un `host` concreto.

[11] Puede comprobarse haciendo un ping a una dirección IP correspondiente a un equipo que no este contemplado en la columna `destination`.

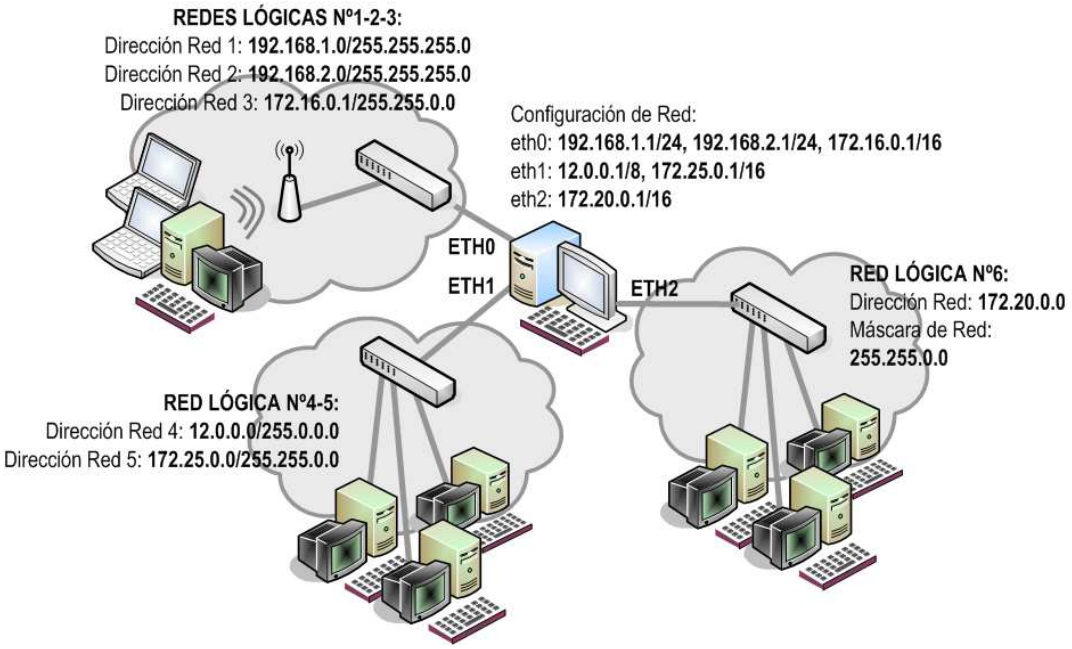


Figura 1.8: Estructura de la red expuesta en el ejemplo tras la primera modificación.

La comunicación con máquinas que no se encuentran en la tabla de enrutamiento se debe realizar a través de lo que se denomina puerta de enlace o `gateway`. A esta puerta de enlace le serán pasados los paquetes TCP/IP para que busque la forma de hacerlos llegar a su destino.

Observando la figura 1.9 puede advertirse que para que nuestro equipo pueda comunicarse con los equipos pertenecientes a la red lógica `172.22.0.0/16` o cualquier otra red externa (o Internet) necesitará al `gateway` con dirección IP `192.168.100.2/24`. Es necesario agregar nuevas reglas de enrutamiento para informar al equipo como acceder a los diferentes destinos. Para ellos usaremos el comando `route` cuya sintaxis se detalla a continuación:

```
[root@linux]#  route add/del -net/host 'direccionIP'
                          netmask 'mascara' gw 'gateway' dev 'interfaz de red'
```

El parámetro `add/del` informa de si nuestro deseo es añadir, `add`, una nueva regla de `routing` o en cambio deseamos eliminarla, `del`. En relación al parámetro `net/host` indica si la regla afecta a un destino que es una red lógica, `net`, o tan sólo a un `host` individual.

Por ejemplo, para informar a nuestro equipo de que para comunicarnos con los equipos de la red lógica `172.22.0.0/16` (`net 172.22.0.0`, `netmask 255.255.0.0`) deberemos mandar los paquetes TCP/IP que formen parte de la comunicación al equipo `192.168.100.2` (`gw 192.168.100.2`) a través de la interfaz de red de nuestro equipo `eth3` (`dev eth3`) escribiríamos,

```
[root@linux]#  route add -net 172.22.0.0 netmask 255.255.0.0
                              gw 192.168.100.2 dev eth3
```

De esta forma, la nueva tabla de enrutamiento de nuestro equipo quedaría de la siguiente forma:

```
[root@linux]# route -n

Kernel IP routing table
Destination     Gateway         Genmask         Flags   Metric  Ref   Use   Iface
192.168.1.0     0.0.0.0       255.255.255.0       U       0       0     0     eth0
12.0.0.0        0.0.0.0         255.0.0.0         U       0       0     0     eth1
172.20.0.0      0.0.0.0       255.255.0.0         U       0       0     0     eth2
172.22.0.0    192.168.100.2   255.255.0.0         U       0       0     0     eth3
```

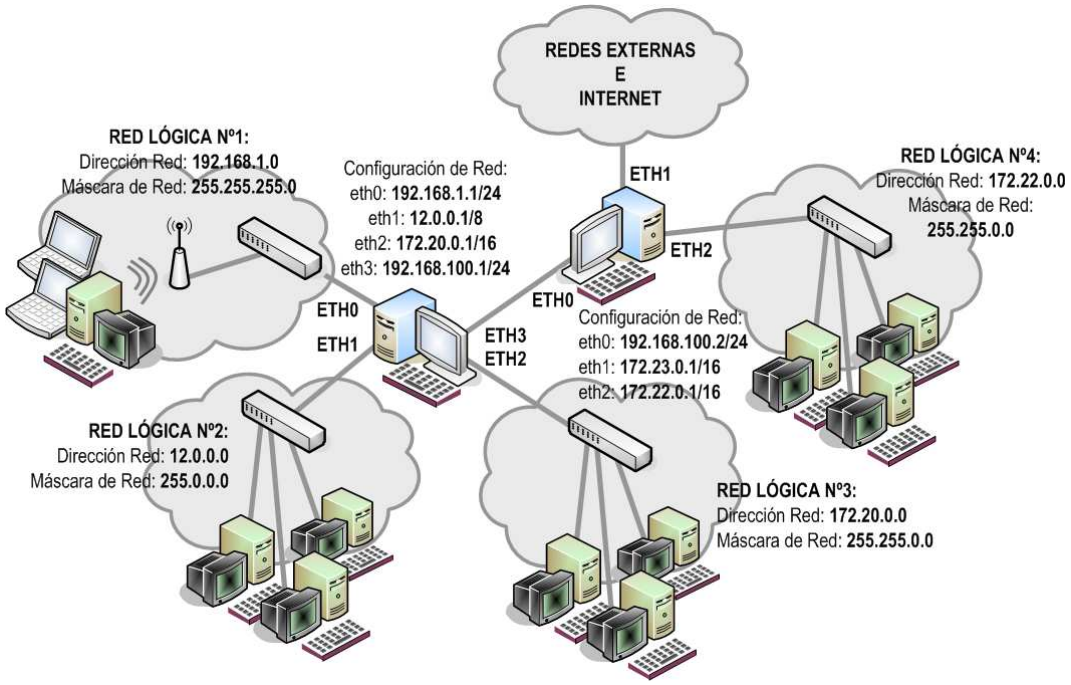


Figura 1.9: Estructura de la red expuesta en el ejemplo tras la segunda modificación.

Si además queremos indicar a nuestro equipo, que para poder acceder a cualquier otra red lógica e Internet (net `0.0.0.0`, netmask `0.0.0.0`) es necesario enviar igualmente los paquetes TCP/IP al equipo que hace de `gateway 192.168.100.2` (gw `192.168.100.2`) ejecutaríamos el siguiente comando:

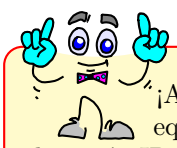
```
[root@linux]#  route add -net 0.0.0.0 netmask 0.0.0.0
                                          gw 192.168.100.2 dev eth3
```

o lo que es lo mismo,

```
[root@linux]#  route add default gw 192.168.100.2 dev eth3
```

La tabla de enrutamiento definitiva de nuestro equipo sería la siguiente:

```
[root@linux]# route -n

Kernel IP routing table
Destination     Gateway         Genmask         Flags   Metric  Ref   Use   Iface
192.168.1.0     0.0.0.0     255.255.255.0       U       0       0     0     eth0
12.0.0.0        0.0.0.0       255.0.0.0         U       0       0     0     eth1
172.20.0.0      0.0.0.0       255.255.0.0       U       0       0     0     eth2
172.22.0.0   192.168.100.2    255.255.0.0       U       0       0     0     eth3
0.0.0.0      192.168.100.2      0.0.0.0         U       0       0     0     eth3
```



¡Aclaración! Las dos últimas reglas de enrutamiento programadas en nuestro equipo equivalen a decir que para poder acceder a cualquier otro destino con dirección IP no perteneciente a las redes lógicas a las que pertenecen nuestras interfaces de red es necesario enviar todo paquete TCP/IP al equipo `192.168.100.2` (`default gw 192.168.100.2`), el cual a su vez habrá sido programado para decidir que hacer con los paquetes que reciba con la finalidad de que finalmente alcancen su destinatario.

Por último, apuntar que para que nuestro equipo pueda salir hacia Internet será necesario que conozca las direcciones IP de los equipos que se encargarán de hacer la resolución de nombres[12]. Para ello debera editarse el fichero `/etc/resolv.conf` y se agregarán tantas líneas como equipos servidores DNS se deseen. La sintaxis de este archivo es,

```
nameserver 'dirección IP equipo servidor DNS'
```

Por ejemplo, si nuestros servidores DNS tuviesen direcciones IP `95.32.178.4` y `95.32.178.5` el fichero `/etc/resolv.conf` podría modificarse ejecutando,

```
[root@linux]#  echo nameserver 95.32.178.4 > /etc/resolv.conf
[root@linux]#  echo nameserver 95.32.178.5 >> /etc/resolv.conf
```

1.4. Los archivos de configuración de red

El problema de configurar nuestro equipo en red a través de los parámetros `ifconfig` y `route` es que la configuración no es persistente. Toda configuración que llevemos a cabo de la forma anterior se perderá al reiniciar el servicio de red,

```
[root@linux]# /etc/init.d/network restart
```

o al reiniciar el propio equipo,

```
[root@linux]# init 6
```

En el caso de desear que la configuración sea persistente es necesario modificar directamente los ficheros de configuración `/etc/sysconfig/network-scripts/*`. Estos ficheros serán observados por nuestro sistema operativo GNU/Linux tanto en el momento de arranque de la máquina, como cuando reiniciemos el servicio de red.

La configuración de las interfaces de red se realiza en los scripts `ifcfg-eth`x (siendo x el identificador de la interfaz). Por ejemplo, en el caso de que dispongamos de tres interfaces de red deberemos disponer de los ficheros `ifcfg-eth0`, `ifcfg-eth1` e `ifcfg-eth2`, y si además quisiéramos asignar a la primera interfaz de red más de una dirección IP, deberemos crear los ficheros `ifcfg-eth0:1`, `ifcfg-eth0:2`, ... El contenido de estos ficheros esta compuesto por un junto de atributos a los cuales se les asigna el valor deseado, entre los cuales podrían destacarse los siguientes:

Atributos de los ficheros de configuración de red:
`/etc/sysconfig/network-scripts/ifcfg-eth`x

- **DEVICE**

Su valor se debe corresponder con el identificador de la interfaz de red que deseamos configurar a través del fichero/script: `eth0`, `eth0:1`, `eth1`, `eth2`, `eth2:5`, ...
Ejemplo:

```
DEVICE=eth0
```

.

- **BOOTPROTO**

[12] En el capítulo 2 se explicará detalladamente qué es un resolutor de nombres, pero básicamente se puede decir que es el encargado de traducir los nombres de dominio de los que hagamos uso (por ejemplo, `www.google.com`, `mail.aragon.es`, ...) a su correspondiente dirección IP.

Forma en que serán asignados a la interfaz de red los valores de configución. Su valor puede ser `dhcp` o `static`. En el caso de que este sea `dhcp`, el único requisito adicional es que en la red exista un servidor `DHCP` el cual se encargará de asignarle a la interfaz de red una dirección IP válida. Por el contrario, si su valor es `static` deberemos introducir en estos ficheros los parámetros de configuración encargados de asignar a nuestra interfaz de red la dirección IP deseada.
Ejemplo:

```
BOOTPROTO=static
```

.

- **IPADDR, NETMASK, NETWORK, BROADCAST**

En el caso en que `BOOTPROTO=static` el valor de los siguientes parámetros configuran sus valores más característicos. `IPADDR` recibe la dirección IP que será asignada a la interfaz definida en `DEVICE`, `NETMASK` la máscara de red y, `NETWORK` y `BROADCAST` las direcciones IP más baja y alta de la red lógica a la que pertenece la interfaz de red. Por definición, la dirección de red debe corresponderse con el resultado de llevar a cabo una operación `AND` lógica entre la dirección IP y la máscara asignada, `NETWORK=IPADDR & NETMASK`. Según esto, aunque se pueden especificar, resulta redundante definir los parámetros `NETWORK` y `BROADCAST` siendo necesario únicamente especificar `IPADDR` y `NETMASK`.

- **PEERDNS, DNS1|2**

Por lo general, nuestro equipo debe conocer direcciones IP de servidores de nombres. Para indicar al equipo esas direcciones IP tenemos dos opciones: modificar manualmente el fichero `/etc/resolv.conf` o que atomáticamente sean registrados estos en `/etc/resolv.conf` en el momento en que se inicie el servicio de red. En el caso en que deseemos que esta configuración sea automática deberemos asignar al atributo `PEERDNS` el valor `yes`. Esta situación suele corresponderse cuando la configuración es dinámica vía `DHCP`, donde el propio servidor `DHCP` informa de quienes son tales servidores. En el caso en que `PEERDNS=yes` y la configuración sea manual `BOOTPROTO=static`, deberemos hacer uso de los atributos `DNS1` y `DNS2` para informar de quiénes son.

- **GATEWAY**

Aunque a través del parámetro `GATEWAY` se nos permite especificar la dirección IP del equipo que nos permite salir hacia el exterior de nuestra red lógica, es más correcto especificar este a través del fichero `/etc/sysconfig/network` como veremos más tarde.

- **ONBOOT**

Este atributo informa de si la configuración de la interfaz de red correspondiente será asignada durante el arranque de la máquina o no: `yes` o `no`.
Ejemplo:

```
ONBOOT=yes
```

- **METRIC**

En el caso de que se haga uso de algoritmos de enrutamiento dinámico determina la métrica a utilizar.
Ejemplo:

```
METRIC=15
```

- **USERCTL**

En el caso en que el valor de este atributo sea `yes` estaremos permitiendo controlar el estado de la interfaz de red (`ifup/ifdown`) a todo usuario sin necesidad de ser `root`.
Ejemplo:

```
USERCTL=no
```

A continuación se muestra el contenido de los ficheros `ifcfg-eth0`, `ifcfg-eth1` e `ifcfg-eth1:1` suponiendo que nuestro equipo dispone de dos interfaces de red (`eth0` y `eth1`). La primera de las interfaces se configura vía `dhcp` y la segunda manualmente (`static`).

```
#Contenido del fichero de configuración de
#                    la interfaz de red eth0: ifcfg-eth0
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
METRIC=10
```

```
#Contenido del fichero de configuración de
#                    la interfaz de red eth0: ifcfg-eth1
DEVICE=eth1
BOOTPROTO=static
IPADDR=192.168.1.101
NETMASK=255.255.255.0
PEERDNS=yes
DNS1=192.168.1.253
DNS2=192.168.1.254
ONBOOT=yes
METRIC=10
```

```
#Contenido del fichero de configuración de
#                    la interfaz de red eth0: ifcfg-eth1:1
DEVICE=eth1:1
BOOTPROTO=static
IPADDR=192.168.2.101
NETMASK=255.255.255.0
PEERDNS=no
ONBOOT=yes
METRIC=10
```

Por último, tan sólo nos falta conocer como indicar a nuestro equipo quién va a ser nuestro `gateway` por defecto sin hacer uso del comando `route`. Para ello, deberemos editar el fichero de

configuración de red `/etc/sysconfig/network` y agregar o modificar los atributos `GATEWAY` y `GATEWAYDEV`:

Atributos del fichero de configuración de red: `/etc/sysconfig/network`

- **NETWORKING**

Permite que la red pueda configurarse como un servicio más (`yes` o `no`).
Ejemplo:

```
NETWORKING=yes
```

- **HOSTNAME**

Nombre cualificado completo asignado a nuestro equipo (FQDN, Fully Qualified Domain Name). Normalmente este nombre es asignado a nuestro equipo a través de los comandos `hostname`, `domainname` y `dnsdomainname`. Este nombre deberá ser validado por el fichero `/etc/hosts` o por nuestro servidor DNS.
Ejemplo:

```
HOSTNAME=equipo1.ingemartin.es
```

- **GATEWAYDEV**

Interfaz de red a través de la cual se alcanza el equipo que nos permite comunicarnos con los equipos que se encuentran fuera de la red o redes lógicas a las que pertenecemos.
Ejemplo:

```
GATEWAYDEV=eth1
```

- **GATEWAY**

Especifica la dirección IP del equipo a través del cual nos comunicaremos a través de `GATEWAYDEV` al cual le enviaremos todos aquellos paquetes TCP/IP que vayan dirigidos hacia equipos de la extranet.
Ejemplo:

```
GATEWAY=192.168.100.250
```

.

Según lo anterior, y para terminar de configurar nuestra red, deberíamos modificar el fichero de configuración de red `/etc/sysconfig/network`, que tendría la siguiente forma:

```
#Contenido del fichero de configuración de red: /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=equipo1.ingemartin.es
GATEWAYDEV=eth1
GATEWAY=192.168.100.250
```

¡¡Importante!! Después de modificar estos archivos y si deseamos que los cambios tengan efecto deberemos reiniciar los servicios de red,

```
[root@linux]# /etc/init.d network restart
```

1.5. Configuración de la red con un asistente gráfico

En el caso de disponer de interfaz gráfica de usuario (GUI), una opción mucho más simple y cómoda que las dos anteriores es hacer uso del asistente gráfico[13] `drakconnect`. Este nos presentará un conjunto de formularios que tras rellenarlos modificará los anteriores ficheros de configuración (`/etc/sysconfig/network` y `/etc/sysconfig/network-scripts/`) por nosotros.

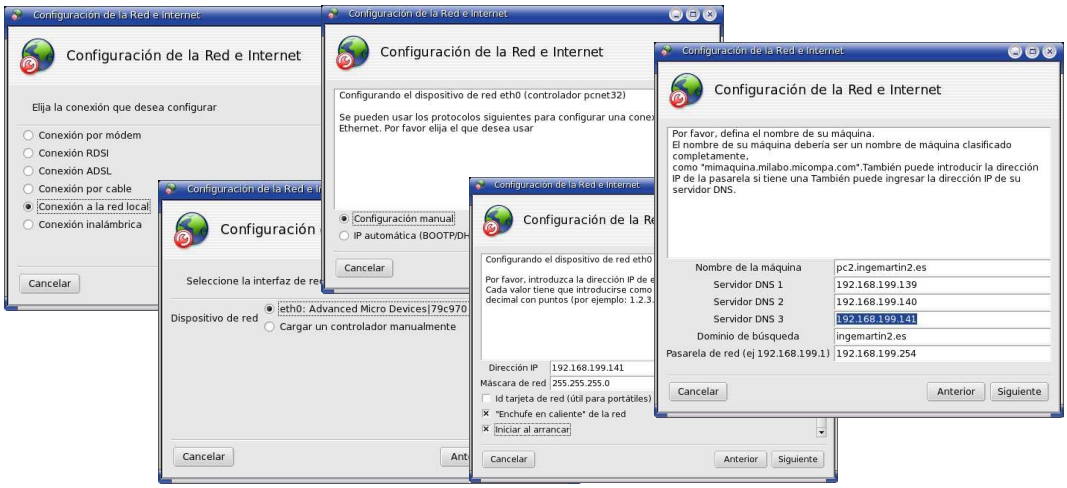


Figura 1.10: Capturas de pantalla de la herramienta `drakconnect` suministrada con Mandriva.

Como se puede observar a través de las capturas de pantalla, `drakconnect` nos interrogará consultándonos todo lo necesario para una adecuada configuración: (1) Tipo de conexión a configurar, (2) interfaz de red de área local a configurar, (3) configuración manual o vía `DHCP`, (4) dirección IP y máscara de red/subred y (5) dirección IP del `gateway` (pasarela de red), de los servidores DNS y el `HOSTNAME`.

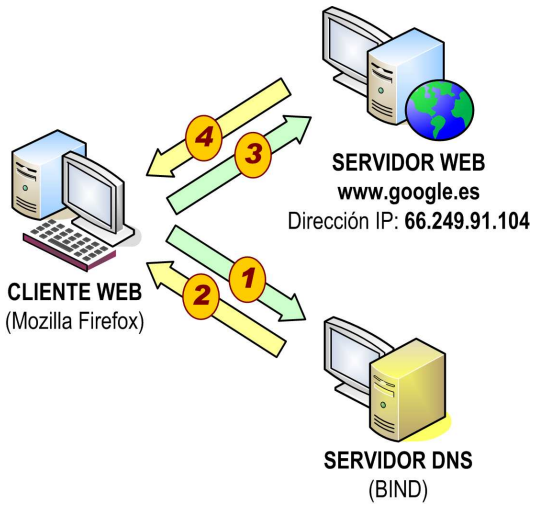
[13] Puesto que para el desarrollo de este libro se ha empleado una distribución de Mandriva, el programa es exclusivo de esta distribución. La distribución que tu utilices tendrá un programa e interfaz gráfico similar.

Capítulo 2

CONFIGURACION DE UN DNS: BIND

2.1. Introducción al concepto de DNS

Ya sabemos que cualquier máquina accesible en nuestra LAN o en la propia Internet con capacidad de comunicarse con las demás tiene asignada, al menos, una dirección IP. El sistema actual[1] de direcciones IP (IPv4) fue estandarizado en 1981 y definió una dirección como la composición de cuatro octetos de ceros y unos (cantidades binarias de 8 dígitos) separados por puntos. Las direcciones IP se catalogan en públicas y privadas. Las públicas son únicas, es decir no se pueden repetir[2], mientras que las privadas son utilizadas reiteradamente por las redes internas de las empresas u otras organizaciones. Un ejemplo de dirección IP privada es la 192.168.1.1. Para entablar una comunicación entre dos equipos es necesario que ambos tengan asignadas sendas direcciones IP. Un buen símil puede ser el de la red telefónica, en la que se necesitan conocer previamente los correspondientes números para establecer una llamada.



Una vez escrito en la barra de direcciones de nuestro cliente Web preferido la `URL` correspondiente al sitio Web a visitar, por ejemplo `http://www.google.es`, tras pulsar `ENTER` se suceden las siguientes acciones:

1. El equipo cliente, para poder dar respuesta a la solicitud http realizada, se pregunta ¿Quién es el equipo que sirve el sitio Web `www.google.es`? Entonces se dirige al servidor `DNS` que tiene configurado para que resuelva el nombre del equipo indicado en la `URL`.
2. El servidor `DNS` nos devuelve la dirección IP del equipo que sirve el sitio Web que deseamos visualizar:
   `www.google.es` $\Longrightarrow$ `66.249.91.104`
3. El equipo cliente solicita vía protocolo `http` al equipo `66.249.91.104` el sitio Web que tiene alojado.
4. El servidor Web nos suministra la página de inicio de `google`.

Figura 2.1: Pasos en la resolución del nombre `www.google.es`.

Para el ser humano es difícil recordar la combinación de dígitos que se corresponde con una dirección IP y por esta razón es normal asignar un nombre a las direcciones. Todos estamos acostumbrados a escribir en nuestro navegador web nombres como `www.google.es`, aunque en realidad

[1] Es de suponer que en poco tiempo se imponga el sistema de direcciones IPv6.

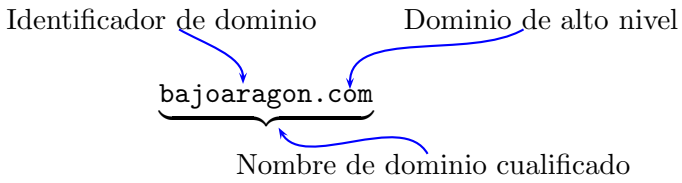
[2] Un ejemplo de dirección IP pública es la 202.12.27.33 (perteneciente a m.root.servers.net), nadie más en el mundo puede utilizarla para comunicarse en Internet.

lo que deseamos es conectarnos a una máquina con dirección 66.249.91.104 (dirección IP pública). Alguien en Internet debe de encargarse de traducir el nombre escrito en el navegador por su correspondiente dirección IP. Las máquinas encargadas de realizar este trabajo se denominan DNS. Las siglas DNS significan Domain Name Server (Servidor de Nombres de Dominio).

Se dice que un DNS resuelve un nombre cuando es capaz de transformar ese nombre en una dirección IP. Cualquier ordenador que permita la navegación por Internet utilizando *nombres* en lugar de direcciones IP debe conocer al menos la dirección IP de un DNS[3]. Así, cuando escribimos en el navegador, por ejemplo, `www.iescomercio.com` nuestro ordenador pregunta a ese DNS la dirección IP asociada al *nombre* que hemos escrito. El DNS se encargará, por medio de procedimientos que posteriormente serán explicados, de resolver la pregunta devolviendo en este caso a nuestro ordenador la dirección IP 217.76.130.87.

Los *nombres* asociados a las redes de ordenadores se denominan *nombres de dominio*. Un nombre de dominio consta de dos o más palabras separadas por puntos. La palabra situada más a la izquierda da la mayor especifidad al nombre y a medida que se llega a la de la derecha se produce una generalización[4]. Por ejemplo `pc1.aula19.cossio.net` haría referencia a la máquina 1, perteneciente al aula 19 del I.E.S. Manuel Bartolmé Cossio. La palabra de la derecha, `net`, se denomina nombre de dominio de alto nivel[5].

En general un nombre de dominio consta del dominio de alto nivel (es, edu, com, net, ...) y de un identificador de domino. Ambos conforman lo que se denomina nombre de dominio cualificado o FQDN (Fully Qualified Domain Name).



Para identificar a las máquinas de un dominio se añaden subdominios, por ejemplo: `pc1.bajoaragon.com` o `pc2.bajoaragon.com`. Lo normal es ir construyendo una estructura jerárquica como la mostrada en la figura 2.2.

2.1.1. Nombres de dominio de nivel y servidores raíz

Inicialmente los *nombres de dominio de alto nivel* eran:

.com	Empresas comerciales.
.edu	Educacional (Universidades y colegios).
.org	Organizaciones de fin no lucrativo.
.net	Redes de ordenadores independientes pero conectadas a Internet.
.int	Internacional.
.gov	Gobierno y organismos oficiales de USA.
.mil	Organismos militares de USA.
.nato	Organización del Tratado del Atlántico Norte (OTAN/NATO).

Posteriormente hubo que añadir nombres a la lista, debido a la creciente necesidad de empresas y organismos por presentar sus productos en Internet. Algunos ejemplos son: `.au` (Australia), `.at` (Austria), `.es` (España), ... Los nombres de dominio que cuelgan de estos se denominan dominios de segundo nivel[6]. Los que cuelgan de los de segundo nivel se denominan de tercer nivel y así sucesivamente.

[3] Recuerda que esta dirección o direcciones se encuentran en `/etc/resolv.conf`.
[4] Esto no es norma, aunque lo normal es que sea así.
[5] En ingles TLD, Top Level Domain name.
[6] En inglés 2LD, Second Level Domain Name.

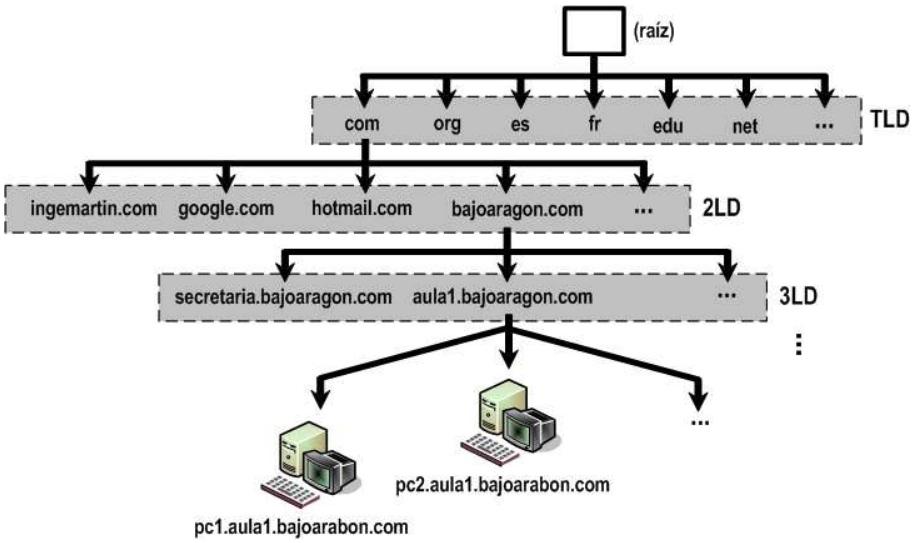


Figura 2.2: Estructura jerárquica de nombres de dominio.

El responsable de estos dominios y del denominado dominio raíz que está por encima de ellos[7] es el ICANN[8]. Una de las misiones de este organismo es gestionar la concesión de nombres de dominio y sus direcciones IP asociadas.

El ICANN posee 13 servidores de nombres *distribuidos* por el mundo[9]. Se denominan Servidores Raíz o Top Name Servers (TNS) y todos ellos tienen la misma información de forma que se reparten el trabajo de resolución a la vez que cada uno de ellos es una copia de seguridad del resto. Estos servidores contienen las zonas con los nombres de dominio de segundo nivel (2LD) de todo el mundo; advierte que esto es una cantidad de información enorme.

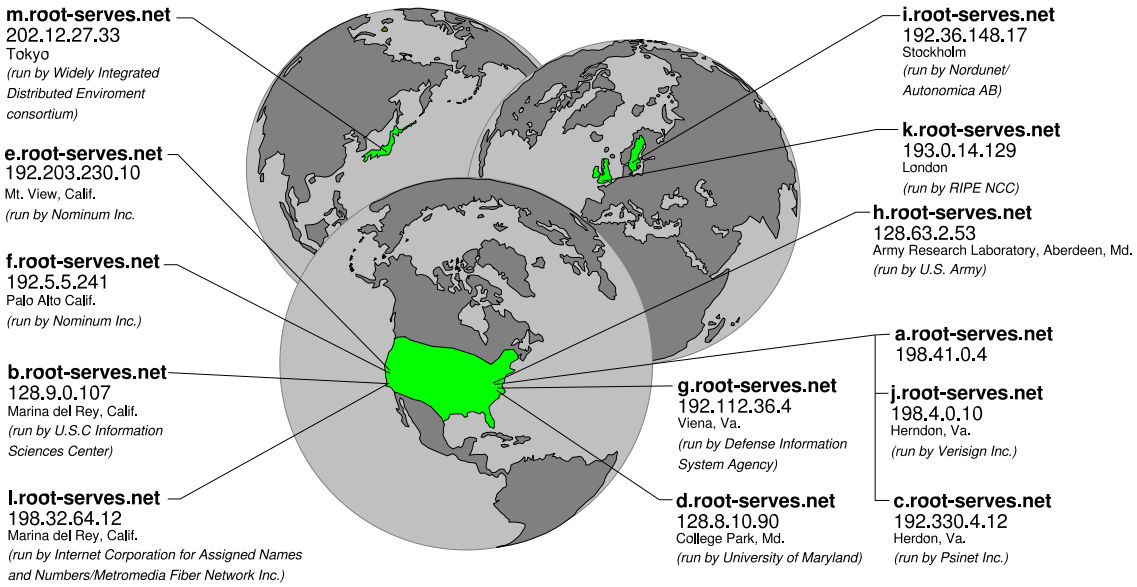


Figura 2.3: Situación mundial de los DNS raíz.

[7] El dominio raíz no tiene ningún nombre asociado, ver figura 2.2.

[8] ICANN son las iniciales de Internet Corporation for Assigned Names and Numbers.

[9] Decimos distribuidos por el mundo, pero como puede verse en la figura 2.3 la mayoría están en USA, dos en Europa y otro en Japón.

Lo dicho no significa que los TNS sean los encargados de hacer todas las resoluciones de nombres de dominio de segundo nivel. Existen gran cantidad de servidores de nombres pertenecientes a empresas y particulares que realizan también esta función, si tienen autoridad para ello. Por ejemplo el nombre de dominio `iescomercio.com` es de segundo nivel y tiene asignada la dirección IP 217.76.130.87; uno de los servidores que tiene autoridad para hacer la resolución de dicho nombre es `dns7.servidoresdns.net` que a su vez tiene la dirección IP 217.76.128.131 y que evidentemente no es ninguno de los trece servidores raíz.

Cuando alguien hace una consulta a la `url www.iescomercio.com` se envía la petición al servidor DNS que tenemos configurado en nuestro ordenador (fichero `/etc/resolv.conf`), como seguramente no tendrá autoridad para hacer la resolución vuelve a realizar una petición que conducirá finalmente a `dns7.servidoresdns.net`. Este último buscará en su denominada zona de autoridad el nombre `www.iescomercio.com`, que estará enlazado a la dirección IP 217.76.130.87. Está será devuelta a quien hizó la petición en un principio completando así la resolución.

2.1.2. Tipos de servidores de nombre de dominio

En el mundo existen miles de servidores. No todos se comportan de la misma forma ante una solicitud de resolución. Básicamente se pueden distinguir cuatro tipos:

1. *Servidores de tipo caché.* Este tipo de servidores no tienen autoridad sobre ninguna zona. Tras el arranque un servidor de este tipo no es capaz de hacer ninguna resolución por sí solo. Cuando inicialmente le hacen una consulta, él debe reenviarla a los servidores que saben responderla, cuando estos le contestan el servidor caché almacena la respuesta en memoria para no tener que volverla a preguntar. Normalmente las consultas las hacen a los TLD. Junto a la figura 2.4 se da una explicación de su funcionamiento.

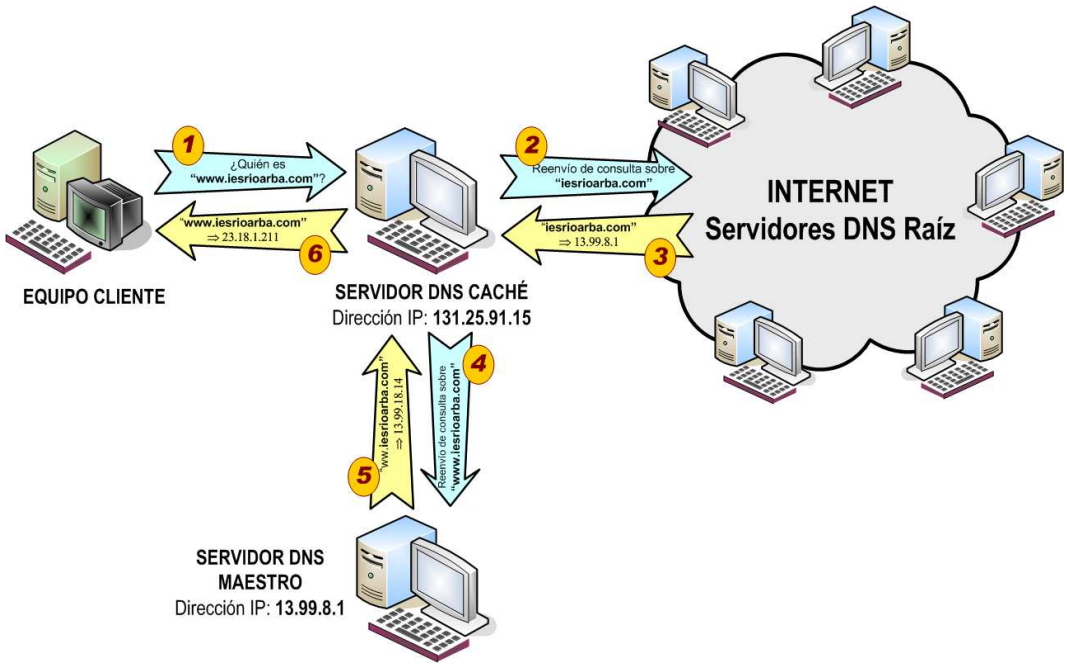


Figura 2.4: Funcionamiento de un DNS cache.

Comentarios a la figura 2.4

Los pasos seguidos en la resoción de un nombre de dominio a través de un servidor DNS de tipo cache son:
1. El equipo cliente realiza una consulta a su `DNS` (en el fichero `/etc/resolv.conf` debe figurar `nameserver 131.25.91.15`) con la finalidad de conocer la dirección IP asociada al equipo que tiene asignado el nombre de dominio `www.iesrioarba.com`.
2. El servidor `131.25.91.15` consulta la lista de todas las resoluciones de nombres de dominio que tiene cachea-

das en memoria como consecuencia de consultas anteriores. En caso de no encontrar la información solicitada es reenviada la petición a los DNS raíz para que estos le informen donde poder encontrar el equipo servidor con autoridad sobre la zona en cuestión `iesrioarba.es`.
3. Los DNS raíz consultan el nombre de segundo nivel (2LD) e informan de la dirección IP del servidor DNS que gestiona los nombres de equipo para ese dominio:

`iesrioarba.es` $\Longrightarrow$ `13.99.8.1`

4. El servidor `13.99.8.1` recibe la petición, lleva a cabo la resolución del nombre (`www.iesrioarba.com` $\Longrightarrow$ `23.18.1.211`), y le entrega dicha información al servidor `131.25.91.15`.
5. El servidor `131.25.91.15` al recibir la información la almacena (cachea) en memoria con la finalidad de poder responder en el futuro a solicitudes relacionadas con el mismo nombre de dominio.
6. El servidor `131.25.91.15` suministra al equipo cliente la información solicitada.

2. *Servidores esclavos o secundarios.* Contienen información sobre algunas zonas. Estos datos son copias de los que contienen los servidores maestros. Cuando la información cambia en un servidor maestro el esclavo simplemente la copia para actualizarse. Ver la figura 2.5 para aclarar conceptos sobre el funcionamiento de este tipo de servidores.

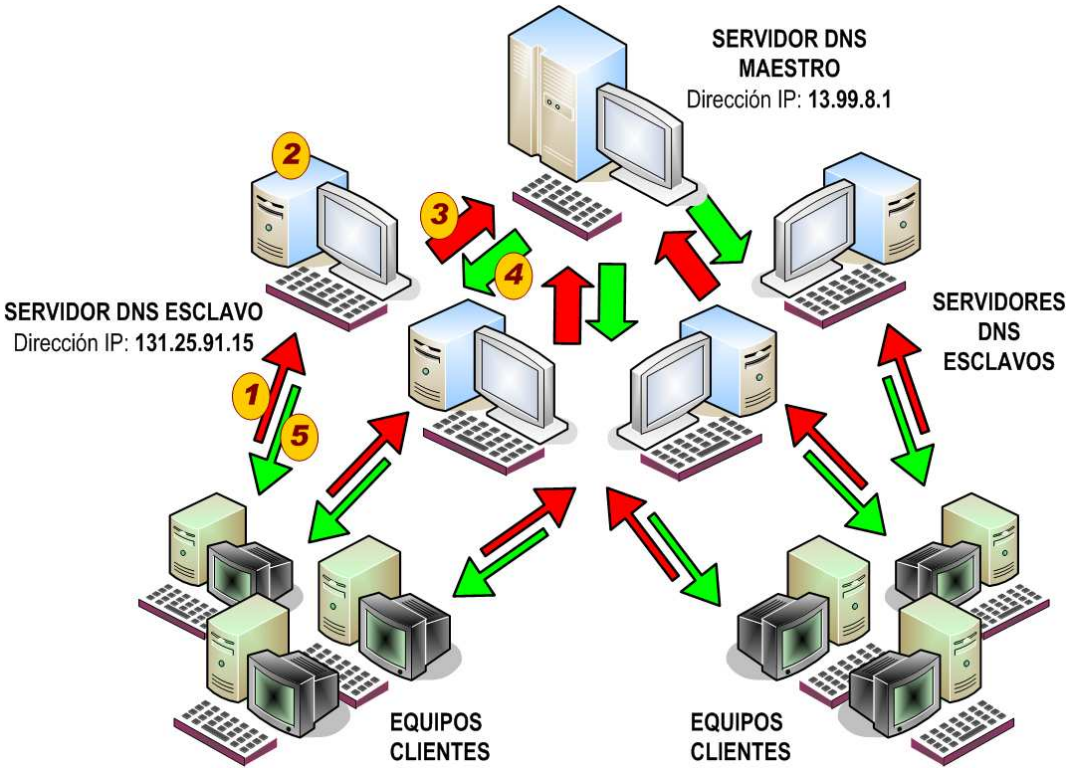


Figura 2.5: Funcionamiento de un DNS esclavo.

Comentarios a la figura 2.5

Tal como se puede observar en la figura los pasos seguidos en la resolución de un nombre de dominio a través de un servidor DNS de tipo esclavo son:
1. El equipo cliente realiza una consulta a su servidor DNS (en el fichero `/etc/resolv.conf` debe figurar `nameserver 131.25.91.15`) con la finalidad de conocer la dirección IP asociada al equipo que tiene asignado el nombre de dominio `www.iesrioarba.com`.
2. El servidor `131.25.91.15` consulta la lista de todas las zonas sobre las que tiene información con la finalidad de encontrar la correspondiente a `iesrioarba.es`.
3. En el caso de detectar que se corresponde con una zona esclava sobre la cual no ha recibido previamente información por parte de su servidor maestro (por ejemplo, `13.99.8.1`), le solicita ha este último la información

necesaria para configurar el fichero de zona asociado a `iesrioarba.es`.
4. El servidor `13.99.8.1` al recibir la petición, suministra a su servidor esclavo la información necesaria.
5. El servidor `131.25.91.15` al recibir esa información genera automáticamente el fichero de zona correspondiente, lo consulta y lleva a cabo la resolución solicitada por el equipo cliente. Una vez que ya tiene configurado el fichero de zona, el servidor esclavo ya se encuentra preparado para contestar cualquier otra solicitud asociada a ese (`www.iesrioarba.com`) u otro nombre de equipo dentro del dominio `iesrioarba.es`. De esta forma se consigue distribuir el trabajo de resolución de nuestro servidor maestro entre diferentes servidores esclavos.

3. *Servidores maestros*. Son aquellos que tienen autoridad sobre una zona. La figura 2.6 da una breve explicación de funcionamiento.

Contienen los datos que permiten hacer la resolución correspondiente sin necesidad de reenviar la petición. Esos datos son una copia maestra, es decir, son los que hay que cambiar cuando la información de la zona a resolver cambia.

Figura 2.6: Funcionamiento de un DNS maestro.

Comentarios a la figura 2.6

Tal como se puede observar en la figura los pasos seguidos en la resolución de un nombre de dominio a través de un servidor DNS de tipo maestro son:
1. El equipo cliente realiza una consulta a su DNS (en el fichero `/etc/resolv.conf` debe figurar `nameserver 131.25.91.15`) con la finalidad de conocer la dirección IP asociada al equipo que tiene asignado el nombre de dominio `www.ingemartin.es`.
2. El servidor `131.25.91.15` consulta la lista de zonas sobre las que tiene autoridad y en caso de encontrar la zona especificada (`ingemartin.es`), consulta el archivo de zona asociado a ella para informar de la dirección IP correspondiente al nombre del equipo `www.ingemartin.es` dentro del dominio `ingemartin.es`:

$$\texttt{www.ingemartin.es} \Longrightarrow \texttt{21.8.109.21}$$

3. La información obtenida es devuelta al cliente que inicio la consulta.

4. *Servidores de reenvío*. No poseen autoridad sobre las zonas que resuelven. Cuando se les hace una petición ellos la reenvían a los servidores que tienen configurados esperando que estos le devuelvan la respuesta.

Comentarios a la figura 2.7

Tal como se puede observar en la figura los pasos seguidos en la resolución de un nombre de dominio a través de un servidor DNS de tipo forward (reenvio) son:
1. El equipo cliente realiza una consulta a su servidor DNS (en el fichero `/etc/resolv.conf` debe figurar `nameserver 131.25.91.15`) con la finalidad de conocer la dirección IP asociada al equipo que tiene asignado el nombre de dominio `www.ingemartin.com`.
2. El servidor `131.25.91.15` reenvía la solicitud al servidor DNS que tenga configurado (por ejemplo, hacia el equipo DNS `10.168.9.18`).
3. El servidor `10.168.9.18` recibe la petición y trata de realizar la resolución del nombre. En el caso de que tenga la información solicitada, se la suministrará al servidor `131.25.91.15`. En caso contrario, se repetirá el proceso de reenvió hasta que algún servidor con autoridad contenga la información requerida.

Figura 2.7: Funcionamiento de un DNS de reenvío.

4. Una vez que el servidor `131.25.91.15` recibe una respuesta, esta es suministrada al equipo cliente que generó la consulta, que además la almacena en memoria cache con la finalidad de agilizar la resolución ante futuras solicitudes sobre el mismo nombre de dominio.

La estructura jerárquica de los servidores de nombre hace que con muy pocas consultas se pueda hacer la resolución.

2.2. Estructura del capítulo

El objetivo principal de este capítulo es ser capaces de convertir una máquina en uno de esos miles de servidores de nombres de dominio existentes. Para ello emplearemos BIND, una aplicación ampliamente utilizada en Internet para este fin. El capítulo está estructura para conseguir:

1. Instalar correctamente BIND, el software que permite convertir tu equipo en un DNS.
2. Configurar de forma sencilla tu DNS. A modo de tutorial se mostrará como crear un DNS caché que podrás dejar funcionando, de forma que ya no dependas de los DNS que te proporciona tu proveedor de servicios de Internet (ISP).
3. Saber como se puede controlar BIND de forma remota.
4. Comprender algunos de los estamentos y cláusulas de BIND.
5. Manipular los archivos de configuración y saber como se chequean.
6. Conocer y configurar los distintos tipos de DNS que es posible encontrar.

PRÁCTICA: CONFIGURACIÓN DE UN DNS

2.3. Instalación del software necesario

Para que un ordenador desarrolle las funciones de servidor de nombres de dominio es necesario que tenga el software específico que se encarga de hacer este trabajo. Nuestra elección es BIND. Existen otras aplicaciones de software libre capaces de hacer esta función, como por ejemplo DJBDNS; los motivos que justifican esta decisión son que BIND está bien documentado y muchos de los servidores de Internet lo utilizan actualmente. El paquete software específico que debemos instalar es *bind-9.x*, donde *x* denota la sub-versión. En el momento de escribir este documento, la versión disponible de BIND es la 9.3.1, pero los conceptos y procedimientos que en este capítulo se van a describir serán válidos para cualquier otra versión, independientemente de la utilizada aquí.

Su instalación puede llevarse a cabo de dos formas:

1. A través de una interfaz gráfica de usuario (GUI), tras ejecutar el comando `rpmdrake`. Para ello escribiremos en la línea de comandos:

```
[root@linux]# rpmdrake &
```

   Aparecería un asistente gráfico, tal como se muestra en la figura 2.8 que nos permite elegir el paquete deseado, y a continuación basta con hacer un click en *instalar*.

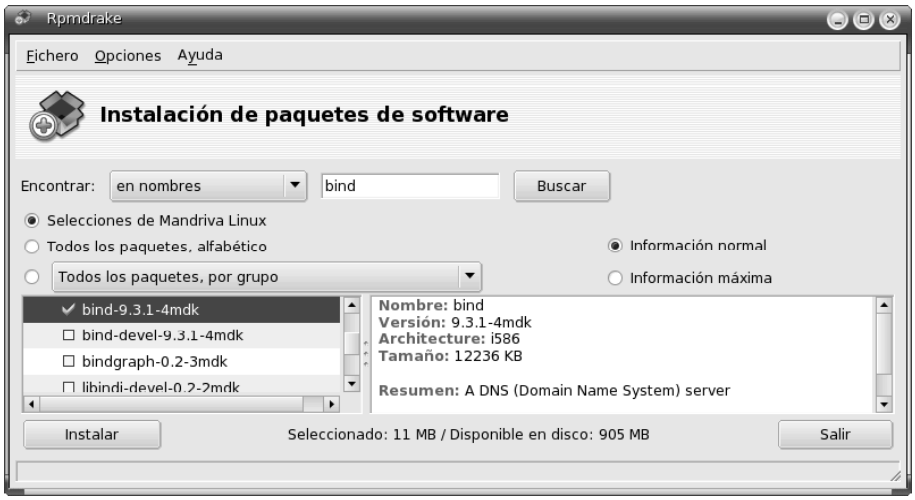


Figura 2.8: GUI para instalación de software en Mandriva.

2. Una segunda opción más cómoda consiste en hacer la instalación desde la línea de comandos (CUI, Interfaz de Línea de comandos, o mediante el uso de una consola o Terminal).

```
[root@linux]# urpmi bind
```

2.4. Conceptos básicos sobre la configuración de BIND

Una vez instalado el software, deberemos configurarlo. Puesto que el programa se encarga de transformar los nombres de dominio en direcciones IP, la configuración consiste en escribir sobre determinados archivos las equivalencias entre unos y otras.

Esta operación requiere de dos pasos:

1. En primer lugar editar el fichero `named.conf`. En él se van a definir las distintas zonas o nombres de dominio que nuestro servidor de nombres esta autorizado a resolver.

2. En el caso de estar configurando una zona maestra se deberá crear un archivo[10] asociado a cada una de estas zonas (definidas en `named.conf`). En él se indicarán las direcciones IP asociadas a los equipos que forman parte del dominio y que además podrán ser resueltas. Este fichero se llama *archivo de configuración de zona* o simplemente *archivo de zona*.

¡¡Importante!! Debes distinguir entre nombre de dominio o zona (definida en el fichero `named.conf`), y los nombres de los equipos pertenecientes a dicha zona definidos en el archivo asociado a ella. Los equipos, y los servicios que ofrecen estos, pertenecientes a la zona se identifican con un acrónimo precediendo al nombre del dominio. Por ejemplo, un nombre de dominio podría ser `ieschirinos.com`, y dentro de esta ofrecerse servicios `HTTP` (`www.ieschirinos.com`) y FTP (`ftp.ieschirinos.com`); o existir una máquina perteneciente al mismo que se llame `pc1.depinformatica.ieschirinos.com`.

2.4.1. Una configuración sencilla: El DNS caché

La configuración más sencilla que podemos hacer con BIND es la de un DNS caché. Un servidor de nombres caché sólo tiene configurada una zona o dominio que indica que es un servidor caché. Cuando el servidor recibe una solicitud de resolución, pregunta a otro con autoridad sobre la zona solicitada. Este le transmitirá la información de la misma (ver figura 2.4).

Siguiendo los pasos expuestos en la sección 2.4 en primer lugar crearemos[11] un fichero llamado `/etc/named.conf`. Para la configuración como servidor caché, bastaría con que el contenido del mismo fuera el mostrado a continuación[12].

```
zone "."{
  type hint;                          //Zona de tipo caché
  file "/var/named/named.ca"; };
```

A través del tipo `hint` le decimos a BIND que deseamos un servidor caché para cualquier nombre de dominio (indicado mediante `zone "."`). Cuando le hagamos una consulta al servidor, este buscará en alguno de los servidores contenidos en el archivo `/var/named/named.ca`.

La composición del archivo `/var/named/named.ca` se corresponde con el paso 2 indicado en la sección 2.4. En realidad, nos podemos evitar construir este archivo, ya que es igual a cualquier otro `named.ca` utilizado en el mundo. Es fácil encontrarlo en Internet[13], pero más fácil es encontrarlo en nuestro ordenador ya que normalmente se suministra con el programa BIND. Para encontrarlo puedes escribir[14]:

```
[root@linux]# find /usr -name named.ca
```

El archivo que contiene la información de la zona, en este caso el fichero `/var/named/named.ca`, se denomina *archivo de configuración de zona* o simplemente *archivo de zona*. El contenido inicial del mismo es:

[10] Es un sólo archivo en el caso más sencillo, pero pueden ser varios. Por otro lado, en la configuración de otro tipo de servidores, por ejemplo el servidor caché (ver sección 2.4.1), ni siquiera hay que crearlo, ya que es proporcionado por la propia distribución.

[11] En el caso de que ese archivo ya existiese renombralo para reemplazarlo por el que aquí se presenta.

[12] Observa que en este fichero los comentarios se hacen con // cuando sólo ocupan una línea y con /* ...*/ en caso de que ocupen varias (similar a los ficheros de código fuente en C y C++). Además, al igual que en los ficheros de configuración y script de los sistemas UNIX, se puede utilizar el símbolo # para hacer los comentarios.

[13] Puedes encontrar una versión actualizada de este en `ftp://ftp.rs.internic.net/domain` con el nombre de `named.root`, que deberás renombrar a `named.ca`

[14] Tras encontrarlo debes copiarlo en el lugar especificado por el estamento `zone`. En nuestro caso deberías copiarlo en `/var/named/`

```
@                           IN SOA  localhost. lsfl.sflj. (1 21 10 100 1200)
.                  3600000  IN NS   A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.  3600000  A      198.41.0.4
;
.                  3600000  NS      B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.  3600000  A      192.228.79.201
                           ⋮
```

En secciones posteriores se explicarán detalladamente cada uno de los elementos que componen este archivo. Por ahora saber que cada pareja de líneas como la mostrada a continuación quiere decir que la zona "." es resuelta, por ejemplo, por el servidor de nombres `B.ROOT-SERVERS.NET`.

```
                                ⋮
.                  3600000  NS   B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.  3600000  A    192.228.79.201
                                ⋮
```

Este a su vez tiene asignada la dirección IP (address) 192.228.79.201.

2.4.2. Comprobación del servidor caché

Tras realizar lo indicado anteriormente, el servidor BIND ya está configurado para dar servicio caché. Para probar su funcionamiento y suponiendo que dispones de conexión a Internet, haremos lo siguiente,

1. Editaremos el fichero `/etc/resolv.conf`[15] y tras comentar su contenido escribiremos en la primera línea `nameserver localhost`, de esta forma indicamos a nuestro equipo que él mismo realizará la labor de servidor de nombres. Una forma rápida de hacer esto es ejecutar,

   ```
   [root@linux]# echo "nameserver localhost" >/etc/resolv.conf
   ```

   pero ten cuidado porque entonces *machacas* todo el contenido.

2. Iniciar el servicio ofrecido por BIND. El programa que ofrece el servicio en background es `named`. Bastará con que ejecutemos[16]:

   ```
   [root@linux]# service named start
   Iniciando named:                           [ OK ]
   ```

   Pudiera ocurrir que el comportamiento fuese el siguiente,

   ```
   [root@linux]# service named start
   named: already running
   ```

[15] Recuerda que en este fichero se especifican el servidor o servidores de nombres para nuestra máquina. En este caso debemos indicar que la dirección IP del servidor de nombres es la de cualquiera de las interfaces de red o simplemente `localhost`.

[16] En las distribuciones en las que no existe el comando `service` el servicio debe arrancarse por medio de:

```
[root@linux]# /etc/init.d/named start
```

Esto significa que el servicio ya estaba arrancado. En este caso, para que la configuración realizada sea tenida en cuenta el servicio deberá reiniciarse,

```
[root@linux]# service named restart
Deteniendo named: rndc: connect failed: connection refused
                                                           [ OK ]
Iniciando named:                                           [ OK ]
```

Observa que aparece un mensaje indicando que la conexión `rndc` ha fallado. De momento no daremos importancia a este hecho y posteriormente, en la proxima sección (la 2.5), se explicará como solucionarlo.

A continuación puedes utilizar el comando `dig` para comprobar que tu máquina, actuando como DNS, resuelve nombres. Escribe:

```
[user@linux]$ dig www.cossio.net
```

Con este comando se solicita información sobre la dirección IP asociada a `www.cossio.net` y los servidores de nombres que resuelven el dominio `cossio.net`. El resultado que obtendrás será parecido a esto:

```
; <<>> DiG 9.3.1 <<>> www.cossio.net
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33509
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;www.cossio.net.  IN         A

;; ANSWER SECTION:
www.cossio.net.   86400 IN   A   217.76.130.84

;; AUTHORITY SECTION:
cossio.net.       172800 IN  NS  dns1.servidoresdns.net.
cossio.net.       172800 IN  NS  dns2.servidoresdns.net.

;; Query time: 1695 msec
;; SERVER: 192.168.1.128#53(192.168.1.128)
;; WHEN: Wed Jul 12 22:19:45 2006
;; MSG SIZE rcvd: 100
```

Observa que en la cuarta línea empezando por el final aparece un tiempo de 1695ms (1.695 segundos). Este indica lo que le ha costado a nuestro servidor hacer la resolución. Esto es así porque ha tenido que preguntar en Internet a otro/s servidor/es sobre la resolución de `www.cossio.net`. La próxima vez ya no tendrá que preguntar a un servidor del exterior; nuestro servidor caché ha guardado la información y si le volvemos a preguntar lo mismo comprobaremos que el tiempo de respuesta es mucho menor.

```
[user@linux]$ dig www.cossio.net

; <<>> DiG 9.3.1 <<>> www.cossio.net
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31295
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;www.cossio.net.  IN         A

;; ANSWER SECTION:
www.cossio.net.   84802 IN    A   217.76.130.84

;; AUTHORITY SECTION:
cossio.net.       171202 IN   NS  dns1.servidoresdns.net.
cossio.net.       171202 IN   NS  dns2.servidoresdns.net.

;; Query time: 1 msec
;; SERVER: 192.168.1.128#53(192.168.1.128)
;; WHEN: Wed Jul 12 22:19:45 2006
;; MSG SIZE rcvd: 100
```

Observa que el tiempo de respuesta se ha reducido a 1 milisegundo. Observa también que las cantidades numéricas que están delante de `IN` también se han reducido, exactamente en 1598 unidades. Esos números son los tiempos de vida de los registros de recurso[17] e indican la cantidad en segundos que tardará nuestro servidor de nombres caché en perder la información que ha almacenado. Por ejemplo el tiempo de vida inicial del registro `A` eran 86400 segundos, es decir 24 horas; transcurrido ese tiempo el registro `A` será borrado.

2.5. Control remoto del servidor de nombres: rndc

`rndc` significa control remoto del servidor de nombres (**r**emote **n**ame **d**aemon **c**ontrol). Esta herramienta permite recargar, parar, comprobar el estado, mostrar estadísticas, ... del servidor de nombres de forma remota.

El funcionamiento de `rndc` requiere de un archivo de configuración con una grámatica similar a la de `/etc/named.conf`[18], pero con la salvedad de que sólo acepta tres tipos de estamentos `options`, `key` y `server`, además de la cláusula `include` que puede emplearse en cualquiera de los estamentos.

Estamentos de `/etc/rndc.conf`

- **options**

La estructura es,

```
options{
        Cláusulas de configuración;
};
```

Dentro de este estamento hay tres posibles cláusulas:

[17] Registros de recurso son por ejemplo `A` y `NS`. El tiempo de vida se denota por `TTL`. Todo esto se verá posteriormente en la sección 2.7.

[18] El archivo `/etc/named.conf` y sus correpondientes estamentos y cláusulas serán estudiados en la sección 2.6.

default-server *nombre del servidor o dirección IP*;
Este será el nombre o dirección del servidor utilizado por `rndc` cuando no se ejecute con la opción `-s`.
Ejemplo:

```
default-server localhost;
```

default-key *clave*;
Aquí debe indicarse el nombre de la clave dada en el estamento `key`.
Ejemplo:

```
default-key "miclave";
```

default-port *nº de puerto*;
Puerto utilizado en la comunicación con el servidor remoto. Este puerto será empleado siempre y cuando no se especifique ninguno al ejecutar el comando (opción `-p`) o se haya definido el puerto en el estamento `server`. El puerto que se usará por defecto es el 953.
Ejemplo:

```
default-port 953;
```

- **server**

La estructura es,

```
server direccion IP o nombre{
                                Cláusulas de configuración;
};
```

Este estamento es opcional y en caso de no existir se tomará la configuración dada en el estamento `options`. Dentro de este estamento hay dos posibles cláusulas:

key {*clave1;clave2; ...*};
Esta cláusula se utiliza para vincular una clave definida en el estamento `key` con el servidor. Si no se estipula ninguna se tomará la indicada en el estamento `options`.
Ejemplo:

```
key {"miclave";};
```

port *nº de puerto*;
Indica el puerto por el que se produce la comunicación con el servidor. Si no se estipula ninguna se tomará la indicada en el estamento `options`.
Ejemplo:

```
port 953;
```

- **key**

La estructura es,

```
key clave{
            Cláusulas de configuración;
};
```

Dentro de este estamento hay dos posibles cláusulas:

algorithm *Tipo de algoritmo*;
BIND sólo soporta el algoritmo `hmac-md5` por lo que el empleo de esta cláusula es único.
Ejemplo:

```
algorithm hmac-md5;
```

secret *Clave secreta*;
Sirve para indicar la clave secreta utilizada en la comunicación segura. Debe ser compartida por los ficheros `/etc/rndc.conf` y `/etc/named.conf` para que se pueda establecer dicha comunicación. Esta clave puede ser obtenida utilizando el comando `rndc-confgen` que será comentado después.
Ejemplo:

```
secret "yXFhgShGEZ1oMsnNe03tWg==";
```

Tras exponer las partes de las que está compuesto el fichero `/etc/rndc.conf` el siguiente paso sería crearlo. Podríamos hacerlo escribiendolo línea a línea, pero existe una forma más sencilla. Debido a que su contenido no admite grandes modificaciones, la mayoría de los `rndc.conf` utilizados en el mundo son muy similares. Esto hizo que los desarrolladores de BIND desarrollaran un comando que crea automáticamente el archivo de configuración. Este comando es `rndc-confgen` y su uso es tan sencillo que basta con ejecutarlo para obtener un posible fichero de configuración de `rndc` para nuestro ordenador:

```
[root@linux]# rndc-confgen
# Start of rndc.conf
key "rndc-key"  {
                algorithm hmac-md5;
                secret "aPYDw+6OQCak0yRVW3cXPQ==";
};

options         {
                default-key "rndc-key";
                default-server 127.0.0.1;
                default-port 953;
};
# End of rndc.conf

# Use with the following in named.conf, adjusting the allow list as needed:
# key "rndc-key"{
# algorithm hmac-md5;
# secret "aPYDw+6OQCak0yRVW3cXPQ==";
# };
#
# controls {
# inet 127.0.0.1 port 953
# allow { 127.0.0.1; } keys { "rndc-key"; };
# };
# End of named.conf
```

Unicamente deberemos copiar el contenido mostrado. Puesto que debe haber relación entre el fichero `/etc/rndc.conf` y el `/etc/named.conf`, en este último se deben incluir las correspondientes líneas propoporcionadas tras la ejecución del comando `rndc-confgen`. Así los ficheros `/etc/rndc.conf` y `/etc/named.conf` quedarían:

```
# Fichero /etc/named.conf
options {
    directory "/var/named/";
};

key "rndc-key"{
    algorithm hmac-md5;
    secret "aPYDw+6OQCak0yRVW3cXPQ==";
};

controls {
    inet 127.0.0.1 port 953
    allow { 127.0.0.1; } keys { "rndc-key"; };
};

zone "."{
    type hint;
    file "/var/named/named.ca"; };
```

```
# Fichero /etc/rndc.conf
key "rndc-key"{
    algorithm hmac-md5;
    secret "aPYDw+6OQCak0yRVW3cXPQ==";
};

options {
    default-key "rndc-key";
    default-server 127.0.0.1;
    default-port 953;
};
```

Cuando reiniciemos de nuevo el servicio `named`, el demonio `rndc` estará ya activo,

```
[root@linux]# service named restart
Deteniendo named: rndc: connect failed: connection refused
                                                           [ OK ]
Iniciando named:                                           [ OK ]
```

El aviso que aparece es debido a que durante la parada del servicio no se detecta el funcionamiento de `rndc`. Comprueba que reiniciando de nuevo el mensaje desaparece:

```
[root@linux]# service named restart
Deteniendo named:                          [ OK ]
Iniciando named:                           [ OK ]
```

El siguiente paso será aprender a utilizar `rndc`. La sintaxis del comando es:

```
rndc [-c config ] [-s server ] [-p port] [-y key ] command
```

Comandos y opciones de `rndc` Índice alfabéticorndc!comandos y opciones

- **Opción -c**

El lugar donde normalmente se encuentra el fichero de configuración de `rndc` es `/etc/rndc.conf`. Esta ubicación puede ser cambiada utilizando la opción `-c`.
Ejemplo:

```
[root@linux]# rndc -c /etc/named/rndc.conf restart
```

- **Opción -s**

Sirve para indicar el servidor con el que se hace la comunicación remota `rndc`. Si no se indica esta opción se tomará el servidor por defecto dado con la cláusula `default-server` en el estamento `options`. Si existe el estamento `server` en `/etc/rndc.conf`, el servidor indicado será el escogido antes que el designado en `default-server`.
Ejemplo:

```
[root@linux]# rndc -s 10.1.23.6 status
```

- **Opción -p**

Sirve para indicar el puerto por el que se realizará la comunicación con el servidor. Si no se especifica esta opción se tomará el puerto asignado en el fichero `/etc/rndc.conf`.
Ejemplo:

```
[root@linux]# rndc -p 953 stop
```

- **Opción -y**

Se utiliza para indicar la clave utilizada en la comunicación en el caso de que en `/etc/rndc.conf` se hayan definido varias.
Ejemplo:

```
[root@linux]# rndc -s 11.2.54.7 -y "rndc-key" halt
```

- **Comando reload**

Tras su ejecución se recargarán por completo el archivo de configuración `/etc/named.conf` y todos los ficheros de configuración de zona.
Ejemplo:

```
[root@linux]# rndc reload
```

La ejecución del anterior comando puede ser muy pesada para el sistema si el servidor cuenta con miles de zonas (y sus respectivos ficheros de configuración). Normalmente no es necesario recargar todas las zonas sino aquella que es modificada, por esta razon el comando `reload` ofrece la opción de especificar una zona en concreto.
Ejemplo:

```
[root@linux]# rndc reload cossio.net
```

- **Comando reconfig**

Similar al comando `reload`, pero en este caso recarga el fichero de configuración `/etc/named.conf` y las nuevas zonas creadas. No se recargan las zonas que ya existían, aunque hayan sido cambiadas.
Ejemplo:

```
[root@linux]# rndc reconfig
```

- **Comando stop**

Este comando detiene el servidor salvando los cambios recientes que se hayan podido producir por una actualización dinámica. Así antes de parar se guardarán los ficheros maestros de las zonas actualizadas.
Ejemplo:

```
[root@linux]# rndc stop
```

- **Comando halt**

Para el servidor inmediatamente. Las actualizaciones no serán guardadas, pero serán recuperadas cuando el servidor sea de nuevo arrancado. Su utilización es similar a la de `stop`.

- **Comando status**

Muestra por pantalla el estado del servidor.

Antes de continuar es conveniente que hagas pruebas con el comando `rndc`.

2.6. El fichero `named.conf`

El archivo `named.conf`, al ser un fichero de configuración, debería encontrarse dentro del directorio `/etc/`. Es posible que tras la instalación de BIND el archivo no se encuentre en ese lugar, en cuyo caso, tenemos dos opciones:

1. Crear el archivo `/etc/named.conf` completamente vacío para posteriormente introducir las líneas de configuración necesarias. Esto es lo que se ha hecho en la configuración del servidor caché.

2. Utilizar un fichero de configuración de ejemplo que es proporcionado con la instalación de BIND y que normalmente se ubica en `/usr/share/doc/bind-9.2.3/chroot/named/etc/named.conf`.

¡¡Ayuda!! Si el fichero de ejemplo `named.conf` no se encuentra en la ruta especificada puedes buscarlo en el sistema de ficheros de GNU/Linux. Para ello ejecuta el comando:

```
[root@linux]# find /usr -name named.conf
```

Puesto que la sencillez del fichero `/etc/named.conf` lo permite, nosotros elegiremos la primera opción y crearemos línea a línea el archivo. Esto permitirá una mejor comprensión de su contenido. Comenzaremos con una configuración muy sencilla y gradualmente, a medida que avance la práctica, se escribirán ejemplos más complejos.

2.6.1. Contenido del fichero `/etc/named.conf`

Supongamos que no disponemos del fichero `/etc/named.conf` tras realizar la instalación de BIND. El primer paso será crearlo nosotros mismos. Como el directorio `/etc/` pertenece a *root*, deberemos convertirnos en dicho usuario para crearlo[19],

```
[root@linux]# touch /etc/named.conf
```

El contenido de este archivo normalmente está dividido en varias partes llamadas estamentos. Algunos estamentos son: `acl`, `controls`, `include`, `key`, `options`, `view`, `zone`, ... En este momento sólo se van a comentar cuatro de ellos, los estamentos `options`, `key`, `controls` y `zone`. La estructura sería:

```
// Parte correspondiente al estamento options
options{
            Cláusulas de configuración; };

// Parte correspondiente al estamento key
key "clave"{
            Cláusulas de configuración; };

// Parte correspondiente al estamento controls
controls{
            Cláusulas de configuración; };

//Parte para definir las diferentes zonas
zone "nombre de la zona"{
            Cláusulas de configuración; };
```

Un ejemplo podría ser:

[19] Para crearlo y editarlo a la vez escribiremos,

```
[root@linux]# vi /etc/named.conf
```

```
options{
          directory "/var/named/";                          //Directorio de trabajo
          allow-query {any};                                //Se permite el uso a todos
};

key "mikey"{
          algorithm hmac-md5;
          secret "KdVONf9FplriTVubw85SgQ==";
};

controls {
          inet 192.168.19.12 port 953
          allow { 192.168.19.12; } keys { "mi-key"; };
};

zone "cosmegarcia.org"{
          type master;                                      //Zona de tipo maestra
          file "maestra.cosmegarcia.org";
          notify no;
};
```

A continuación describiremos algunos de los parámetros que pueden contener los estamentos `options`, `key`, `controls` y `zone`.

Parámetros del estamento `options`

- **directory**

Este parámetro indica el directorio de trabajo del servidor. Todos los ficheros indicados en `/etc/named.conf` se podrán escribir con una ruta relativa a la indicada en este parámetro. En el ejemplo mostrado antes, el fichero `maestra.cosmegarcia.org` tiene como ruta absoluta: `/var/named/maestra.cosmegarcia.org`.
Si no se utiliza `directory` se deben indicar las rutas absolutas. Si suponemos que en el ejemplo anterior no se ha estipulado el directorio de trabajo, la línea correspondiente a `file` debería haberse escrito como:

```
file "/var/named/maestra.cosmegarcia.org";
```

- **pid-file**

Aquí se indica la ruta absoluta, o relativa con respecto a `directory`, del fichero en el que el servidor escribe los identificadores de los procesos (PID) que ejecuta el servicio `named`. Si no se especifica nada el fichero por defecto es `/var/run/named/named.pid`.

El fichero de `pid` es utilizado por los programas que quieren enviar señales al servidor de nombres.

- **port**

Aquí se indica el número de puerto que el servidor usa para recibir y enviar el tráfico causado por el protocolo DNS. El protocolo es de tipo `udp` y por defecto el puerto es el 53.

Se recomienda cambiar de puerto únicamente en procesos de testeo del servidor, ya que otro puerto diferente al 53 hará imposible la comunicación global del DNS.
Ejemplo:

```
options{
        port 5353;
};
```

Para comprobar su funcionamiento puedes utilizar `dig` con el parámetro `-p`:

```
dig -p 5353
```

- **allow-query**

Especifica a que hosts se les permite hacer preguntas de resolución de nombres al servidor. Por defecto se permite que cualquier host pregunte al servidor, tal y como se muestra en el ejemplo anterior.
Ejemplo:

```
options{
        allow-query {192.168.19.0/24;};  /*Se permite a los ordenadores
        de la red 192.168.19.0 con máscara 255.255.255.0 */
};
```

Para hacer referencia a una o varias redes simultáneamente es posible definir lo que se denominan listas de acceso (access lists). Para ello se utiliza el estamento `acl`. Evidentemente por su catergoría de estamento debe colocarse fuera del de `options`.
Ejemplo:

```
acl "redescossio" {192.168.19.0/24; 192.168.17.0/24; };
options{
        allow-query {"redescossio";};  /*Se permite a los ordenadores
        de las redes 192.168.19.0 y 192.168.17.0 */
};
```

- **notify**

Es una variable que puede tomar los valores `yes`, `explicit` y `no`. Por defecto vale `yes` y esto hace que cuando se produce algún cambio en las zonas maestras que el servidor debe resolver, se envían mensajes de notificación a aquellos servidores listados en los registros de recurso `NS` (los registros de recurso son descritos en una sección posterior; en particular el `NS` está definido en la página 61) contenidos en el correspondiente archivo de configuración de zona (el archivo de configuración de zona se describe en la sección 2.7).

Si a `notify` se le asigna `explicit` las notificaciones únicamente son enviadas a las servidores listados en el parámetro `also-notify`; y si se le asigna `no`, no envía notificaciones a ningún servidor.

Ejemplo:

```
options{
notify yes;  //Valor por defecto
};
```

- **also-notify**

Define una lista global de las direcciones IP de los servidores a los que también se les enviará notificación cuando una zona es recargada con nueva información, en adición a la lista de servidores contenidos en los registros de recurso `NS`.

Esto ayuda a asegurar que la nueva actualización de la zona se difunde rápidamente entre todos los servidores.

Ejemplo:

```
options{
also-notify {202.12.27.33;192.203.230.10;198.32.64.12;};
};
```

- **forward**

Cuando se le pide a `named` que resuelva un nombre de dominio hay varias opciones; una de ellas es que reenvíe la pregunta a otro servidor. Este es el comportamiento `forward`.

Esta opción sólo tiene sentido si no está vacía la lista de `forwarders`. Por defecto el valor asignado a `forward` es `first`, esto provoca que el servidor pregunte en primer lugar a los `forwarders` y si no encontrara respuesta de los mismos trataría de encontrarla por sí mismo.

Cuando se da el valor `only` a `forward` significa que el servidor sólo preguntará a los `forwarders` (si ellos no responden no podrá hacer la resolución).
Ejemplo:

```
options{
forward only;  //Funcionamiento de solo reenvío
};
```

- **forwarders**

Permite especificar una lista de direcciones IP que será usada para hacer reenvío (forwarding).
Ejemplo:

```
options{
forward first;
forwarders {192.168.19.1;192.168.17.1; };
};
```

Aunque el forwarding se puede hacer desde el estamento `options` de forma general, es posible indicarlo directamente en cada zona que así se desee, teniendo diferentes `forwarders` para cada una. He incluso se pueden definir diferentes comportamientos `first/only` (ver sección 2.10).

Parámetros del estamento `key`

- **algorithm y secret**

La sintaxis y las cláusulas contenidas en este estamento ya fueron explicadas en la página 46. Invitamos al lector que tenga alguna duda sobre su uso a que se remita a dicha página.

Parámetros del estamento `controls`

- **inet**

El estamento `controls` se utiliza para declarar los canales de control que pueden usar los administradores del sistema para modificar la operación del servidor de nombres. Los canales de control son usados por medio del comando `rndc`.

Un control `inet` permite definir un canal de comunicación vía TCP especificando una dirección IP de comunicación y un puerto.

- **allow**

Aunque se abra un canal de cominicación por medio de `inet`, el acceso al mismo está restringido por las cláusulas `allow` y `keys`.

Con `allow` se fija una lista de direcciones IP correspondientes a los hosts permitidos a entrar en el canal abierto por `inet`.

- **keys**

Establece una lista de identificadores de clave válidos para entrar en el canal. Cada uno de los identificadores de las claves contenido en la lista se podrá utilizar para autentificar, vía firma digital, los comandos y respuestas de control dadas en el canal abierto por `inet`.

Parámetros del estamento `zone`

- **type master**

Un servidor de tipo maestro es aquel que contiene una copia maestra de los datos de la zona con autoridad para responder consultas sobre la misma. En la sección 2.8 se mostrarán ejemplos de uso.

- **type slave**

Una zona de tipo `slave` (esclava) es una réplica de una zona maestra. Junto con el tipo `slave` se debe añadir el parámetro `masters` que establece una lista con una o más direcciones IP de servidores maestros con los que el esclavo puede contactar para actualizar la copia de la zona.

Además es aconsejable, aunque no obligatorio, el uso del parámetro `file` para indicar el fichero en el que se escribirán los datos de la zona cada vez que sea recargada, y del cual se leeran cuando el servicio se reinicie. Esto hará más rápido al servidor y reducirá el consumo de ancho de banda.

- **type stub**

Una zona tipo `stub` es similar a la tipo `slave` con la diferencia que en esta se replican únicamente los registros de recurso `NS` en lugar de la zona completa. La zona `stub` no es parte del estándar DNS, sino que es una característica específica de BIND y algunos de los usos para los que fue creada ya no están recomendados. Por esta razón en este libro no va a ser utilizada.

- **type hint**

Con este tipo de zona se crean los servidores caché. Dentro de este tipo de zona debe hacerse uso del parámetro `file` para indicar el fichero en el que estarán recogidos los nombres y direcciones de los servidores raíz. Junto con la instalación de BIND se suele proporcionar el fichero `named.ca` que contiene esa lista. En la página 41 se expuso su uso.

- **type forward**

Esto permite crear una zona de reenvío. Cuando un servidor tiene una zona de este tipo no es él quien hace la resolución sino que pregunta a otra máquina (descrita en el parámetro `forwarders`, página 53) que será la encargada de hacer la resolución.

Es importante no confundir la zona tipo forward `type forward` con las cláusulas `forward` y `forwarders`, ya que son cosas muy diferentes. En una zona de tipo `forward` es donde normalmente se incluyen dichos estamentos (aunque también pueden ser incluidos en el estamento `options`).

El proceso de reenvío sucede cuando el servidor no tiene la respuesta en la caché. Una descripción de su uso puede verse en la sección 2.10.

- **file**

El parámetro `file` se utiliza dentro del estamento `zone`. A `file` se le asigna un fichero, este es el fichero de configuración de zona.
En relación al contenido del fichero, en el caso del servidor maestro es una copia maestra de la misma, en el caso de la zona esclava es una réplica, en el caso de la zona `hint` contiene una lista de los servidores raíz (TLDs), ... A lo largo del capítulo se verán muchos ejemplos de uso.

- **masters**

El parámetro se utiliza dentro del estamento `zone` y únicamente cuando esta zona es de tipo esclavo (`slave`).

Permite indicar quién o quienes son los servidores maestros de esa zona. Esto significa que cuando la zona deba ser actualizada las solicitudes de actualización se harán a los servidores contenidos en la lista `masters`. *Ejemplo:*

```
masters {19.128.9.4;12.68.7.61; };
```

- **notify y also-notify**

`notify` es un parámetro que actuará de forma genérica para todas las zonas si se incluye en el estamento `options` o que actuará de forma local al ser incluido en el estamento `zone`. Lo mismo ocurre con el parámetro `also-notify`.

La descripción de ambos parámetros es similar a la dada anteriormente.

2.7. Los ficheros de configuración de zona

Como ya se ha comentado, cada zona definida en `/etc/named.conf` debe tener un fichero asociado en el que se expongan las relaciones entre las diferentes direcciones IP pertenecientes al dominio y sus nombres.

Para facilitar la comprensión de este archivo supongamos que disponemos de una estrutura de red como la mostrada en la figura 2.9. En el fichero de zona aparecerán las relaciones entre nombres de dominio y direcciones correspondientes a la supuesta estructura del dominio `iesbajoaragon.com`. Observa que por ejemplo la máquina con dirección IP 62.167.122.10 ofrece dos servicios, web y ftp. Pese a que sólo es una máquina tenemos dos servicios y por tanto deberemos asignar a dos nombres diferentes la misma dirección. Como cada servicio se ofrece por un puerto diferente, no habrá ninguna posible confusión cuando se solicite alguno de los dos.

El nombre dado al fichero de zona asociado al dominio puede ser cualquiera, pero una nomenclatura del mismo al azar podría convertir el mantenimiento de BIND en un trabajo tedioso.

Como existen diferentes tipos de zonas y cada una de ellas puede tener uno o más archivos de configuración asociados, recomendamos nombrar a estos archivos con la siguiente estructura:

tipo de zona.nombre de zona

Esto permitira hacer más legibles los listados (comando `ls`) de los archivos de configuración de zona, que nosotros almacenaremos en `/var/named/`. Por ejemplo, el archivo de configuración asociado a la zona de tipo maestro `iesbajoaragon.com` se llamaría

```
maestra.iesbajoaragon.com
```

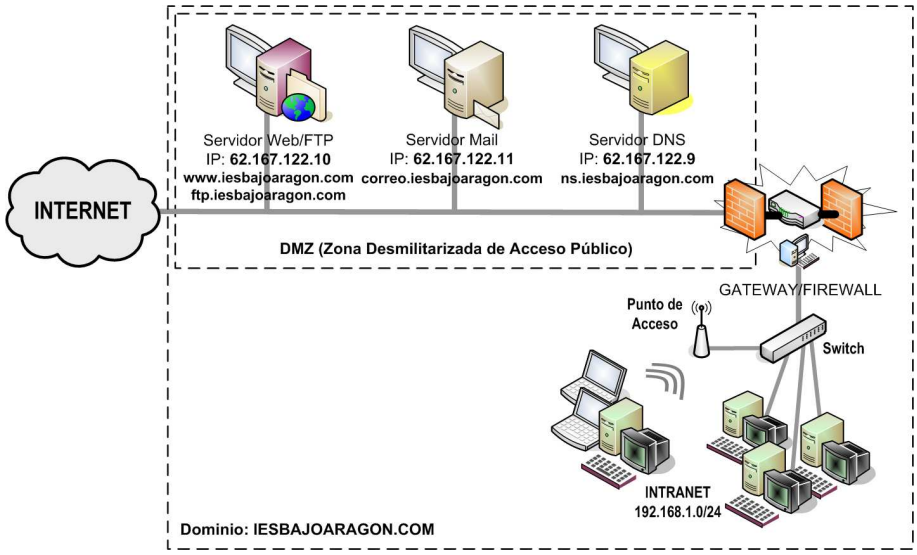


Figura 2.9: Ejemplo de una estructura de red donde se ofrecen servicios a Internet de tipo WEB, FTP Y DNS.

Un ejemplo del contenido de `/var/named/maestra.iesbajoaragon.com` podría ser el siguiente:

```
$TTL 24h
$ORIGIN iesbajoaragon.com.

iesbajoaragon.com.          IN SOA    ns.iesbajoaragon.com.  root.localhost.
                                                             (3001200602;
                                                             21600;
                                                             10800;
                                                             604800;
                                                             21600; )
iesbajoaragon.com.          IN NS     ns.iesbajoaragon.com.
iesbajoaragon.com.          IN MX 10  correo
ns.iesbajoaragon.com.       IN A      62.167.122.9
www.iesbajoaragon.com.      IN A      62.167.122.10
ftp                         IN A      62.167.122.10
correo                      IN A      62.167.122.11
```

En general este fichero contendrá dos tipos de elementos bien diferenciados. En el primero de estos tipos, se encuentran las llamadas directivas (`directives`); el segundo tipo se corresponde con los registros de recurso (`resource records`). Las directivas van precedidas por el símbolo `$`, y pueden ser cuatro: `$ORIGIN`, `$INCLUDE`, `$TTL` y `$GENERATE`. Por su parte los registros de recurso representan el conjunto de información asociada a un nombre de dominio. En principio el orden de los registros dentro del fichero no es importante, con la salvedad de que el registro `SOA` debe ser el primero. Existen muchos registros de recurso, algunos de ellos son `SOA`, `A`, `CNAME`, `PTR`, ... A cotinuación comentaremos algunos registros y directivas, lo que nos permitirá entender el contenido del fichero ejemplo mostrado.

Descripción del contenido de `maestra.iesbajoaragon.com`

- **Directiva $TTL**

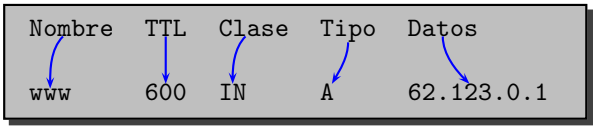
El tiempo de vida general para los registros de recurso se designa por medio de la directiva `$TTL`, que significa Time To Live. Esta directiva se coloca en la primera línea del archivo de configuración, antes del registro `SOA`.

Es utilizado por los servidores caché para saber cuando tienen que actualizar los registros de las zonas descargadas y guardadas en memoria. Esto fue explicado en la sección 2.4.1 (página 44). Si la cantidad asignada a `$TTL` es cero, significa que el registro o registros de recursos no pueden ser cacheados.

Las unidades de la cantidad asociada a `$TTL` son segundos y puede estar comprendida entre 0 y 2147483647 (esto correspondería aproximadamente a 68 años). También puede ser indicada en otras unidades utilizando los siguientes sufijos para la cantidad numérica:

Sufijo	*Unidades*
s	segundos
m	minutos (60 s)
h	horas (3600 s)
d	días (86400 s)
w	semanas (604800 s)

El tiempo asignado por la directiva `$TTL` es utilizado por los registros de recursos que no lo tienen definido esplícitamente. En cada registro es posible asignar de forma particular un tiempo de vida. En el siguiente ejemplo se muestran claramente las partes de la definición de un registro de recurso,



En cualquier caso es un buen criterio establecer una directiva `$TTL` al principio del archivo de configuración, y así lo haremos en adelante.

- **Directiva $ORIGIN**

La segunda línea del ejemplo contiene a esta directiva. La sintaxis es:

```
$ORIGIN nombre_dominio
```

Establece el nombre de dominio por omisión. En otras palabras, mediante esta directiva se indica el nombre de dominio que será añadido a los nombres de máquinas que acompañan a los registros de recurso que no acaban en punto (.). La repercusión de esta línea en el ejemplo anterior se pone de manifiesto por ejemplo en la línea,

```
ftp    IN A    62.167.122.11
```

que es equivalente a escribir,

```
ftp.iesbajoaragon.com.    IN A    62.167.122.11
```

Como `ftp` no termina en punto, el nombre se completa con lo establecido en `$ORIGIN`. Observa que el contenido de la directiva `$ORIGIN` no se añade al nombre especificado cuando este acaba en punto.

Comentarios adicionales sobre el uso de `$ORIGIN`

El uso de esta directiva no es necesario en caso de que la zona declarada en el fichero `/etc/named.conf` se corresponda con el nombre de dominio que se quiere utilizar. Así, en nuestro ejemplo podríamos haber prescindido de ella y todo hubiese funcionado correctamente. Su utilidad aparece cuando el archivo de configuración de zona está compuesto por otros archivos y en alguno de estos utilizamos otro nombre de dominio.

La inclusión de otros archivos se realiza utilizando la directiva `$INCLUDE`. El contenido del fichero se incluye en la misma posición en la que se encuentra `$INCLUDE`. Un ejemplo de verdadero uso de `$ORIGIN` e `$INCLUDE` podría ser el siguinte:

Imaginad que en `/etc/named.conf` se ha definido la zona `cossio.net` de tipo `master`, y que su fichero de configuración asociado, `maestra.cossio.net`, es

```
cossio.net.    IN SOA  ns              root.localhost.
                                       (3001200602
                                       21600
                                       10800
                                       604800
                                       21600 )
               IN NS   ns
www            IN A    62.167.122.10
$INCLUDE       /var/named/cossio/maestra.aula19
ftp            IN A    62.167.122.11
```

El significado de cada una de las líneas se podrá deducir de las explicaciones dadas en apartados posteriores. El objetivo aquí es el de mostrar que en este fichero no se utiliza `$ORIGIN` y sin embargo BIND entendería perfectamente que `www` equivale a `www.cossio.net`, ya que al no terminar el nombre `"www"` en punto, BIND lo completa con el nombre de dominio. Observa también que la penúltima línea indica que se debe incluir el contenido del archivo `maestra.aula19` perteneciente al directorio `/var/named/cossio`. Imagina que el contenido de este archivo es,

```
$ORIGIN aula19.cossio.net.
equipo1   A   192.168.19.1
equipo2   A   192.168.19.2
equipo3   A   192.168.19.3
```

en él sí se ha incluido la directiva `$ORIGIN`. El uso de la misma provoca que, por ejemplo, la línea,

```
equipo1   A   192.168.19.1
```

equivalga a,

```
equipo1.aula19.cossio.net.   A   192.168.19.1
```

Si la directiva `$ORIGIN` no hubiese sido incluida, la línea asociada al `equipo1` hubiera equivalido a,

```
equipo1.cossio.net.   A   192.168.19.1
```

ya que se hubiese completado con el nombre de dominio tomado por defecto. Por último es importante señalar que la directiva `$ORIGIN` sólo tiene influencia en el archivo `maestra.aula19`, es decir se comporta como una directiva local. Por tanto, la línea,

```
ftp   IN A   62.167.122.11
```

contenida en el archivo `maestra.cossio.net` y escrita posteriomente a la llamada del archivo `maestra.aula19` es equivalente a,

```
ftp.cossio.net.   IN A   62.167.122.11
```

- **Registro SOA**

Define el comienzo de una zona de autoridad (`SOA` siginifca Start Of Authority). Es un registro de uso obligatorio, de hecho será el primer registro que nos encontremos en los ficheros de configuración de zona. Su sintaxis es:

```
nombre de dominio IN SOA servidor    responsable    (parámetros)
```

◇ En ***nombre de dominio*** se debe indicar el nombre de dominio que se va a resolver; en el ejemplo anterior (página 57) el nombre de dominio era `iesbajoaragon.com`, que normalmente coincidirá con el nombre de zona dado en el fichero `/etc/named.conf`. Si coincide es posible prescindir de su escritura y sustituirlo por el comodín @:

```
@ IN SOA servidor    responsable    (parámetros)
```

◇ En ***servidor*** se indica el nombre de la máquina que hace de DNS. Se suele utilizar como nombre `ns` (name server) seguido del nombre de dominio, por ejemplo `ns.iesbajoaragon.com`, pero puede darse cualquier nombre permitido.

◇ En ***responsable*** debe escribirse una cuenta de correo o un alias de la misma en la que el mantenedor o mantenedores del servicio encontrarán las incidencias que se produzcan.

◇ Por último en ***parámetros*** se especificarán en este orden:

Numero de serie. Para este se suele elegir la fecha en la que se produjo la última actualización. Por ejemplo, si el día 30 de enero de 2006 se hicieron dos actualizaciones en el número de serie se podría indicar la siguiente cantidad 3001200602.

Periodo de refresco. Los sevidores esclavos hacen la resolución de nombres a partir de la información proporcionada por los servidores maestros que tienen configurados. Esta información es actualizada cada vez que transcurre el tiempo indicado en este parámetro. El funcionamiento de los servidores esclavos será descrito en la sección 2.9. En realidad cada *periodo de refresco* se hace una solicitud de actualización, pero el esclavo sólo actualizará en caso de haberse producido algún cambio en el servidor maestro. Los cambios serán detectados por el esclavo comprobando si el número de serie ha cambiado.

Frecuencia de reintentos. Indica el tiempo que debe esperar el DNS esclavo para solicitar de nuevo la información de actualización al maestro en caso de que este último no le haya respondido transcurrido el *periodo de refresco*.

Tiempo de expiración. Tiempo máximo que será mantenida la información sin actualizar en el servidor esclavo en caso de que el maestro no responda a las solicitudes. Transcurrido ese tiempo la información de la zona se perderá.

TTL Mínimo. Esta cantidad es asignada por el servidor a los registros de recurso en caso de no utilizar la directiva `$TTL`. Indica el mínimo tiempo que permanecerá la información obtenida por el DNS en memoria en caso de funcionar como servidor caché (el funcionamiento como servidor caché fue explicado en el apartado 2.4.1). Valor máximo 3 horas.

Así, un ejemplo podría ser:

```
@  IN SOA  ns  root.localhost. (
           3001200602; Numero de serie
           21600; Periodo de refresco en sg (6h)
           10800; Frecuencia de reintentos en sg (3h)
           604800; Tiempo de expiración en sg - (1semana)
           21600; Tiempo de vida de registros en caché (6h)
           );
```

Comentarios adicionales sobre el uso del registro `SOA`

Observad que en el ejemplo anterior los tiempos se han colocado en segundos y que lo escrito tras un punto y coma se entiende como un comentario[20].

Como ya ha sido comentado, la @ del principio de la línea del registro `SOA` es un *comodín* para indicar que el nombre de dominio es el mismo que el definido en el archivo `/etc/named.conf`. Es decir, si estuvieramos hablando del ejemplo de la página 57, la @ significaría: `iesbajoaragon.com`.

A continuación y después de `SOA` aparece `ns`; al no acabar en punto (.), BIND lo completará con el nombre del dominio. En nuestro ejemplo, escribir `ns` es similar a escribir `ns.iesbajoaragon.com`.

Por último indicar que una de las características de cualquier registro de recurso (como por ejemplo `SOA`) es la clase a la que pertenece; existen varios tipos de clases[21], pero para las aplicaciones usuales emplearemos la clase `IN`, que indica que es un registro que hace referencia a la Internet. De hecho, si no se especifica ninguna clase BIND tomará por defecto la clase `IN`.

- **Registro NS**Índice alfabéticoregistro!NS

`NS` significa nombre del servidor (Name Server). Sirve para indicar el nombre de los servidores de nombres de dominio que tienen autoridad sobre la zona definida. Se suele colocar a continuación del registro `SOA`. Su sintaxis es la que se muestra a continuación:

```
nombre de dominio IN NS servidor con autoridad sobre la zona
```

Un ejemplo podría ser el indicado anteriormente:

[20] En este archivo no se pueden emplear los // o /* ... */ del `named.conf`

[21] Además de la clase `IN` existen otros, como por ejemplo `HS` (hesiod) o `CHAOS` ambos desarrolladas por el MIT (Massachusetts Institute of Technology).

```
bajoaragon.com.    IN NS   ns.bajoaragon.com.
```

Tras lo explicado sobre `$ORIGIN` sabemos que esta línea podría simplificarse por,

```
bajoaragon.com.   IN NS    ns
```

Como `ns` se ha escrito sin el punto final BIND lo completará con el nombre de dominio (o lo que se haya indicado en la directiva `$ORIGIN`). Incluso puede simplificarse todavía más, ya que como en el `SOA` ya se ha definido el nombre de dominio, en este registro puede obviarse. Así se podría escribir simplemente,

```
          IN NS    ns
```

Recuerda que la clase `IN` es la tomada por defecto, luego también podía haberse omitido.

Por último señalar que se pueden añadir tantos registros `NS` como se desee. Un dominio puede estar resuelto por varias máquinas con autoridad.

- **Registro A**Índice alfabéticoregistro!A

El nombre `"A"` proviene de Address (dirección). Mediante este registro se establecen las asociaciones entre nombres de dominio y las correspondientes direcciones IP. Su sintaxis se muestra a continuación:

```
nombre de dominio IN A dirección IP
```

Para decir que la máquina con dirección IP 192.168.19.1 va a ser identificada con el nombre `equipo1.cossio.net` escribiríamos:

```
equipo1.cossio.net.   IN A   192.168.19.1
```

Para indicar, tal como se mostró en el ejemplo de la página 57, que la máquina que ofrece el servicio web `www.iesbajoaragon.com` tiene dirección 62.167.122.10 debería escribirse

```
www.iesbajoaragon.com.   IN A    62.167.122.10
```

De forma similar a como ocurría con el registro `NS` es posible simplificar la línea anterior y escribir,

```
www   A    62.167.122.10
```

Comentarios adicionales sobre el uso del registro `A`

Los ejemplos mostrados corresponden a IP's del tipo IPv4, con bind también es posible asociar IP's del tipo IPv6. En este caso se debe utilizar el registro `A6`, por ejemplo

```
Host.ejemplo.com.   IN A6   0 3ffe:8050:201:1860:42::1
```

Para terminar, señalar que en ocasiones es necesario escribir muchas líneas similares, sobre todo si se está configurando un DNS para una Intranet. Imagina que deseas escribir en el archivo las líneas que permiten resolver las direcciones IP de las 52 máquinas pertenecientes a un subdominio de `cossio.net` al que se ha llamado `aula19.cossio.net`; deberías escribir las siguientes 52 líneas

```
equipo1.aula19.cossio.net.    A   192.168.19.1
equipo2.aula19.icossio.net.   A   192.168.19.2
                              ⋮
equipo52.aula19.cossio.net.   A   192.168.19.52
```

Para evitar este trabajo repetitivo apareció la directiva `$GENERATE`. Su uso se entenderá fácilmente al mostrar la solución para el ejemplo anterior. Las 52 líneas anteriores podrían ser reducidas a una sola[22]:

```
$GENERATE 1-52 equipo$.aula19.cossio.net. A 192.168.19.$
```

Observa que el símbolo `$` es un elemento iterador que variará entre 1 y 52. La directiva `$GENERATE` puede ser usada con cualquier conjunto de registros de recurso que únicamente difieran entre sí por un iterador.

- **Registro MX**Índice alfabéticoregistro!MX

Las siglas `MX` significan Mail eXchanger. Es el registro consultado por un servidor de correo para conocer la dirección IP de otro servidor de correo, al que desea transferir un email.

Para entender este registro imagina dos servidores de correo en la situación mostrada en la figura 2.10, el primero es `mail.cossio.net` que tiene definidos varios usuarios entre ellos `carmen` con dirección `carmen@cossio.net` y el segundo `correo.aulir.com` en el que uno de los usuarios es `juan`, con dirección `juan@aulir.com`.

Cuando `carmen` manda un correo a `juan`, este llegará hasta su servidor `mail.cossio.net`. Lo primero que hará este servidor será comprobar que `carmen` es una de las usuarias permitidas. En caso de que así sea, tomará el nombre de dominio asociado a la dirección de correo del destinatario; en este caso el dominio asociado es `aulir.com`, ya que el destinatario es `juan@aulir.com`. Este nombre de dominio asociado es el que se recoge en un servidor de nombres por medio del registro `MX`. Este es el registro consultado por `mail.cossio.net` para conocer la dirección IP del servidor de correo destino.

El servidor de nombres al recibir una petición de `mail exchange` devolverá la dirección asociada al servidor de correo de `aulir.com`. Esta dirección deberá ser la de `correo.aulir.com`.

Comentarios adicionales sobre el uso del registro MX

Habrás observado que en el ejemplo del `iesbajoaragon.com` el registro `MX` contiene además una cantidad numérica, un 10.

[22] Podría hacerse uso también de la directiva `$ORIGIN`:

```
$ORIGIN aula19.cossio.net.
$GENERATE 1-52   equipo$     A   192.168.19.$
```

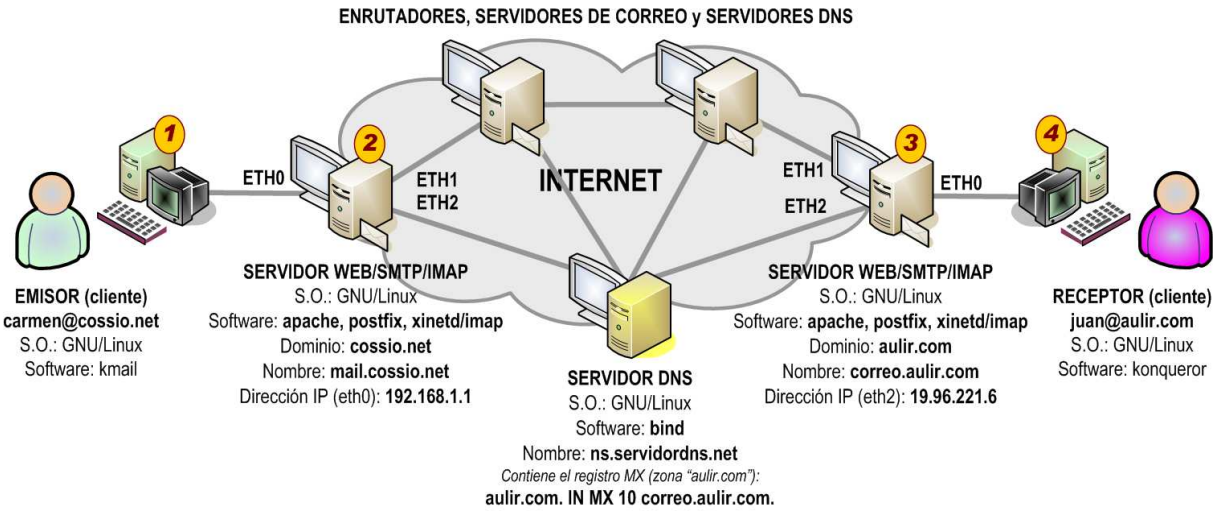


Figura 2.10: Ejemplo comunicación entre dos servidores de correo utilizando los registros `MX`.

```
iesbajoaragon.com.    IN MX 10    correo
```

Esta cantidad especifica el orden de prioridad para responder a una consulta de mail exchange, la mayor prioridad se da a la cantidad numérica menor. Si existen varios registros con igual prioridad se elige uno de ellos de forma aleatoria. Y en el caso de que los servidores de mayor prioridad no respondan se trasladará la petición a los registros con menor prioridad. Las cantidades numéricas no tienen ningún significado especial simplemente sirven para hacer una comparación relativa al resto de registros `MX`. Supongamos que parte de un archivo de configuración es el siguiente,

```
cosmegarcia.com.          IN MX 10   mail.cosmegarcia.com.
                          IN MX 10   mail2.cosmegarcia.com.
                          IN MX 20   correo.backup.es.
mail.cosmegarcia.com.     IN A       11.34.123.2
mail2.cosmegarcia.com.    IN A       11.34.123.3
```

Una petición de mail exchange a `cosmegarcia.com` será atendida por `mail.cosmegarcia.com` o `mail2.cosmegarcia.com` de forma aleatoria y en el caso de que ninguno de los dos respondiese, la petición se trasladaría a `correo.backup.es`.

Es importante señalar que un registro `MX` debe tener asociado siempre una dirección IP por medio de un registro `"A"`. No es suficiente con que tenga asignado un registro `CNAME`.

En el caso en que un dominio tenga asignado un registro `CNAME` y un registro `MX`, la dirección IP apuntada por el `MX` será ignorada y la petición de mail exchange será resuelta devolviendo la dirección apuntada por `CNAME`.

- **Registro CNAME**Índice alfabéticoregistro!CNAME

`CNAME` significa Canonical NAME. Sirve para identificar el verdadero nombre de un alias. Un ejemplo real de esto lo podemos encontrar con `google`. Parte del contenido del fichero de configuración de zona de `google.es` es el siguiente:

```
www.google.es.      47225 IN    CNAME   www.google.com.
www.google.com.     604188 IN   CNAME   www.l.google.com.
www.l.google.com.   13 IN       A       66.249.91.147
www.l.google.com.   13 IN       A       66.249.91.99
www.l.google.com.   13 IN       A       66.249.91.104
```

Aquí se observa que `www.google.es` es un alias de `www.google.com`, que a su vez es alias de `www.l.google.com`. A este último nombre de dominio se le asignan varias direcciones IP. Cuando hacemos esto la resolución va rotando entre las diferentes direcciones provocando un equilibrio (o balanceo) entre las diferentes máquinas.

Lo anterior puede ser una buena solución cuando un sitio web es muy visitado. Para evitar retardos se pueden configurar varias máquinas con el mismo sitio web y que sea el DNS el encargado de resolver cíclicamente las direcciones de dichas máquinas con el fin de conseguir el equilibrado de la carga. Observa en el ejemplo, que lo importante es refrescar los servicios `www.l.google.com` y no los alias; por eso tienen unos tiempos de vida tan bajos.

Por otro lado, el registro `CNAME` tiene muchos detractores entre los administradores de DNS y como en realidad no es un registro necesario para realizar las configuraciones en este libro se evitará usarlo.

2.8. Servidor de nombres de dominio maestro

Como ya se comento en la sección 2.6 una zona de tipo maestro es aquella que tiene una copia maestra del archivo de configuación de zona, es decir, no es ninguna copia efectuada por alguna transferencia de zona.

Para describir la configuración de un DNS maestro nos apoyaremos en la figura 2.11. Para que el DNS resuelva las direcciones de las máquinas del dominio mostrado deberá incluirse en el fichero `/etc/named.conf` la zona `iesrioarba.com`. Posteriormente se debe crear el llamado archivo de zona que contendrá las relaciones entre nombres y direcciones IP de los equipos y servicios contenidos en las redes 192.168.1.0/24 y 192.168.2.0/24.

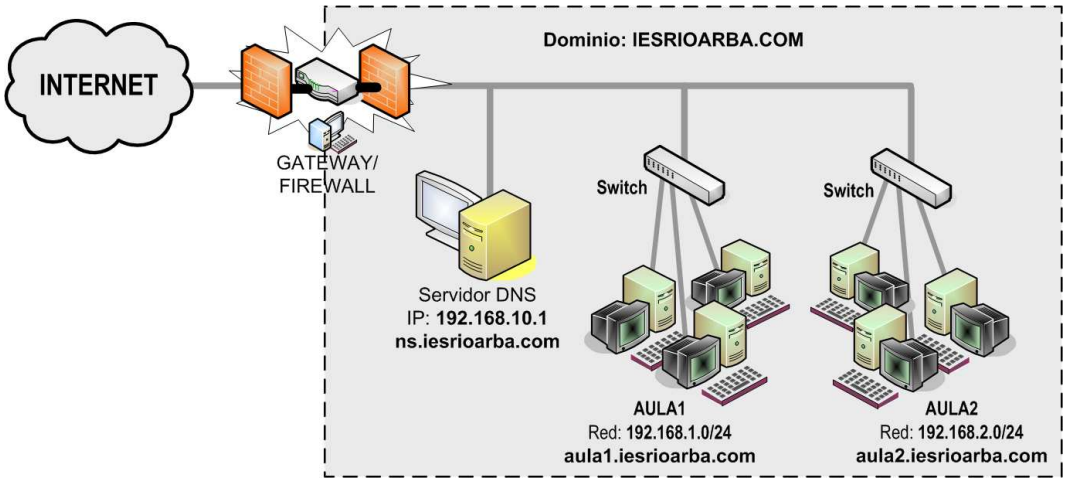


Figura 2.11: Ejemplo de funcionamiento de servidor maestro.

2.8.1. Contenido de `/etc/named.conf`

En la sección 2.6 se comentó que el fichero `/etc/named.conf` estaba compuesto por diferentes secciones o partes. Aquí nos vamos a centrar en la parte correspondiente al estamento `zone`. La estructura de una zona tipo maestro y su correspondiente sintaxis es similar a la que se muestra a continuación:

```
zone   "nombre del dominio a resolver" in {
       type master; //Servidor con autoridad para resolver el dominio
       file "fichero de zona para el dominio";/*El fichero se puede
       especificar mediante una ruta absoluta o relativa*/
};
```

Por ejemplo,

```
zone   "iesrioarba.com" in {
       type master;
       file "/var/named/maestra.iesrioaraba.com";//ruta absoluta
};
```

En la declaración anterior, definimos a `iesrioarba.com` como un nombre de dominio sobre el que tiene autoridad completa nuestro servidor DNS GNU/Linux. Autoridad completa significa que el servidor tiene la potestad de resolver este nombre devolviendo la dirección o direcciones IP que correspondan y que además es dueño del archivo de configuración `/var/named/maestra.iesrioaraba.com`, pudiendo modificarlo cuando lo desee.

Es necesario que el fichero asociado al parámetro `file` exista antes de arrancar el servicio. Observa también que en el ejemplo anterior se ha colocado la ruta absoluta para señalar su ubicación. Se podría haber dado la ruta relativa si se hace uso de la cláusula `directory` dentro del estamento `options`.

```
options {
          directory /var/named";
};
```

El contenido del fichero `/etc/named.conf` que permite a BIND resolver el dominio `iesrioarba.com` utilizando rutas relativas sería:

```
options {
          directory "/var/named";
};

zone      "iesrioarba.com" in {
          type master;
          file "maestra.iesrioarba.com"; };
```

La configuración del servidor maestro es tan sencillo como lo que se acaba de exponer. Por tanto el fichero de configuración de `named` estaría ya completo[23].

2.8.2. Chequeo del archivo `/etc/named.conf`

Cuando nos iniciamos en la configuración de BIND, la causa general de errores se suele deber a problemas de sintaxis. Es muy importante colocar todas las llaves, puntos y coma, comillas, ... tal y como se indica en el ejemplo. En otro caso el servicio no arrancará. Imaginad que el archivo `named.conf` contiene lo siguiente:

```
options {
          directory "/var/named";
};

zone      "iesrioarba.com" in {
          type master;
          file "maestra.iesrioarba.com";
```

[23] En el ejemplo no se han incluido los estamentos `key` y `controls`. Si deseas que `rndc` este operativo deberás incluirlos tal como se mostró en la sección 2.5.

El contenido es muy parecido al mostrado anteriormente, pero con la diferencia de que las llaves ({}) asociadas a la zona no están cerradas[24]. Al arrancar/rearrancar (`start/restart`) el servicio[25] se produciría un error,

```
[root@linux]# service named restart
Deteniendo named:                          [ OK ]
Iniciando named:                           [FALLÓ ]
```

Para corregir este tipo de errores, existe una herramienta administrativa diseñada para comprobar la correcta configuración; así, el chequeo del fichero `/etc/named.conf` se realiza por medio del comando `named-checkconf`. La forma de utilizar este comando es:

```
[root@linux]# named-checkconf [nombre_fichero]
```

El parámetro *nombre_fichero* está entre corchetes porque es opcional. Si no se coloca, por defecto se buscará en `/etc/named.conf`. Un ejemplo de uso sería,

```
[root@linux]# named-checkconf /etc/named.conf
```

que proporcionaría,

```
[rootlinux]# named-checkconf /etc/named.conf
/etc/named.conf:8: '}' expected near end of file
```

indicando que en la línea 8 se esperaba un cierre de llaves } que no se ha producido.

2.8.3. Contenido del archivo de zona

El archivo de zona correspondiente a la zona `iesrioarba.com` contendrá la relación entre direcciones IP y nombres de dominio utilizados en la red mostrada en la figura 2.11. Siguiendo el convenio adoptado en este documento (página 56), este archivo de zona deberá llamarse `/var/named/maestra.iesrioarba.com`. El contenido del mismo podría ser algo parecido a:

```
iesrioarba.com.                IN SOA  ns.iesrioarba.com.  root.localhost.
                                                           (3001200602
                                                           21600
                                                           10800
                                                           604800
                                                           21600 )
iesrioarba.com.                IN NS   ns.iesrioarba.com.
ns.iesrioarba.com.             IN A    192.168.10.1
equipo1.aula1.iesrioarba.com.  IN A    192.168.1.1
equipo2.aula1.iesrioarba.com.  IN A    192.168.1.2
...
equipo1.aula2.iesrioarba.com.  IN A    192.168.2.1
equipo2.aula2.iesrioarba.com.  IN A    192.168.2.2
...
```

[24] Se abren tras "`in`", pero no se cierran tras el parámetro "`file`".

[25] Volvemos a recordar que en sistemas basados en `RedHat`, como `Mandriva`, el control de servicios puede hacerse por medio del comando `service`. En otros sistemas, como por ejemplo los basados en `Debian`, el servicio ofrecido por BIND se arrancaría escribiendo,

```
[root@linux]# /etc/init.d/named restart
```

Los puntos suspensivos indican que el fichero continúa con las líneas que relacionan nombres e IP's de los, por ejemplo, 16 equipos del aula 1 y los 18 equipos del aula 2.

Como el lector habrá podido deducir, este archivo puede ser reducido en gran medida si aprovechamos las posibilidades antes descritas que posee BIND. Así podría ser reescrito como,

```
@          IN SOA   ns            root.localhost.
                                  (3005200601;
                                  21600;
                                  10800;
                                  604800;
                                  21600; )
           IN NS    ns
ns         IN A     192.168.0.1
$INCLUDE   /var/named/arba/maestra.aula1
$INCLUDE   /var/named/arba/maestra.aula2
```

El contenido de los archivos[26] `/var/named/arba/maestra.aula1` y `/var/named/arba/maestra.aula2` respectivamente podría ser:

```
$GENERATE 1-16   equipo$.aula1.iesrioarba.com.   A   192.168.1.$
```

```
$GENERATE 1-18   equipo$aula2.iesrioarba.com.   A   192.168.2.$
```

2.8.4. Chequeo del archivo de configuración y arranque del servicio

Al igual que ocurre con el fichero `/etc/named.conf`, con frecuencia se cometen errores de sintaxis al escribir los ficheros de configuración de zona. Esta es una de las razones por las que es necesario utilizar herramientas que nos permitan detectarlos. El comando que permite realizar esta operación para los archivos de configuración de zona es,

```
[root@linux]# named-checkzone zona [nombrefichero]
```

Por ejemplo,

```
[root@linux]# named-checkzone iesrioarba.com /var/named/maestra.iesrioarba.com
```

proporcionaría,

```
/var/named/maestra.iesrioarba.com:1: no TTL specified; using SOA MINTTL instead
zone iesrioarba.com/IN: loaded serial 3005200601
OK
```

Observa que aparece algo similar a un aviso (`warning`) indicando que no se ha especificado un `TTL`. El `TTL` es el tiempo de vida de los registros de recursos y se definía en el archivo de zona utilizando la directiva `$TTL`. Si no es especificado, como es el caso, BIND asigna de forma automática el `MINTTL`[27] dado en el registro `SOA`.

Antes se ha comentado que los problemas de sintaxis son una de las razones que justifican el chequeo; otra de las razones es la de comprobar la consistencia del servicio. Por ejemplo si dentro del fichero de configuración de zona se incluye una máquina no perteneciente al dominio, durante el chequeo se advierte esta situación indicando que la descripción realizada va a ser ignorada. Imaginad el archivo,

[26] Para utilizar la directiva `$GENERATE` no es necesario incluirla en otro fichero. Las líneas contenidas en `maestra.aula1` y `maestra.aula1` podrían haberse escrito directamente en el archivo de zona.

[27] Ver en la página 61 la explicación de los parámetros del registro `SOA`. Este parámetro fue llamado **TTL mínimo**.

```
@              IN SOA   ns              root.localhost.
                                        (3005200601;
                                        21600;
                                        10800;
                                        604800;
                                        21600; )
               IN NS    ns
ns             IN A     192.168.0.1
$INCLUDE       /var/named/arba/maestra.aula1
$INCLUDE       /var/named/arba/maestra.aula2
fabrica.com.   IN A     62.167.122.14
```

tras realizar el chequeo se obtiene,

```
/var/named/maestra.iesrioarba.com:11:ignoring out-of-zone data (fabrica.com)
zone iesrioarba.com/IN: loaded serial 3005200601
OK
```

Esta respuesta indica que la línea 11, correspondiente a `fabrica.com`, va a ser ignorada.

Por último, y una vez que los chequeos han sido pasados debe arrancarse el servicio `named` para que nuestra máquina comience a resolver nombres teniendo en cuenta la configuración realizada. Ejecutaremos,

```
[root@linux]# service named start
Iniciando named:                           [ OK ]
```

Ejercicio 2.1

Edita el fichero `/etc/named.conf` y crea dos zonas nuevas de tipo maestro (`master`) denominadas `ies1.com` e `ies2.net`. Los ficheros de configuración de las zonas anteriores se llamarán siguiendo la norma indicada en este texto, y se ubicarán dentro del directorio predeterminado (`directory`) `/var/named`.

A continuación, crea y edita los ficheros anteriores (`maestra.ies1.com` y `maestra.ies2.net`) de tal forma que ante una petición de resolución de nombres de dominio, responda con:

- `www.ies1.com` ⇒ `192.168.43.1`
- `ftp.ies1.com` ⇒ `192.168.43.2`
- `www.ies2.net` ⇒ `192.168.65.3`
- `www.ies2.net` ⇒ `192.168.65.4`

Observa que el servicio Web del `ies2` puede ser dado por dos máquinas diferentes. Esto se hace así para evitar sobrecargas, equilibrando las peticiones al servicio Web (se produzca un balanceo) mediante las dos máquinas.

Además de los anteriores nombres de dominio el `ies1` dispone de las aulas 17 y 19 (redes 192.168.17.0/24 y 192.168.19.0/24) con 10 y 15 ordenadores respectivamente que se definirán dentro de los subdominios `aula17.ies1.com` y `aula19.ies1.com` como `equipo1`, `equipo2`, ...

Por su parte, el `ies2` dispone de las aulas 1 y 2 (redes 192.168.1.0/24 y 192.168.2.0/24) con 15 y 16 ordenadores en cada una de ellas. Siguiendo la misma filosofía se desea que las máquinas pertenezcan a los subdominios `aula1.ies2.net` y `aula2.ies2.net` según corresponda, y con nombres `equipo1`, `equipo2`, ...

Solución del ejercicio 2.1

En primer lugar deberemos editar el fichero `/etc/named.conf` y describir las zonas de la siguiente forma:

```
options  {
         directory "/var/named";
         };
zone     "ies1.com" in {
         type master;
         file "maestra.ies1.com";
         };
zone     "ies2.net" in {
         type master;
         file "maestra.ies2.com";
         };
```

A continuación escribiremos los archivos de configuación asociados a ambas zonas. Primero editamos el archivo `/var/named/maestra.ies1.com`. Puesto que se han de resolver dos subdominios correspondientes a las dos aulas 17 y 19 se ha decidido utilizar llamadas a ficheros. Así, el fichero de configuración de zona queda más legible y estructurado.

```
$TTL 3h
@          IN SOA   ns            root.ies1.
                                  (3001200602;
                                  21600;
                                  10800;
                                  604800;
                                  21600; )
           IN NS    ns
www        IN A     192.168.43.1
ftp        IN A     192.168.43.2
$INCLUDE   /var/named/ies1/maestra.aula17
$INCLUDE   /var/named/ies1/maestra.aula19
```

A continuación escribiremos el contenido del fichero `maestra.aula19`, que como se ha dicho, contiene 15 ordenadores. Este es un caso típico en el que conviene utilizar la directiva `$GENERATE`.

```
$GENERATE 1-15   equipo$.aula19.ies1.com.   A   192.168.19.$
```

Como se habrá podido observar, se podría haber utilizado la directiva `$ORIGIN`. En el fichero `maestra.aula17` haremos uso de la misma para mostrarlo,

```
$ORIGIN aula17.ies1.com.
$GENERATE 1-10   equipo$    A   192.168.17.$
```

Por último, en el archivo `/var/named/maestra.ies2.net` escribiremos la configuración de la zona `ies2.net`. Para ejemplizar el posible uso de `$ORIGIN` en un solo archivo, las relaciones entre nombres de máquinas y sus direcciones IP de las aulas 1 y 2 serán realizadas en un único archivo llamado `maestra.aulas`.

```
$TTL 3h
@          IN SOA   ns            root.ies2.
                                  (3001200602;
                                  21600;
                                  10800;
                                  604800;
                                  21600; )
           IN NS    ns
www        IN A     192.168.65.3
www        IN A     192.168.65.4
$INCLUDE   /var/named/ies1/maestra.aulas
```

El contenido del archivo `maestra.aulas` sería,

```
$ORIGIN aula1.ies2.net.
$GENERATE 1-15   equipo$   A   192.168.1.$
$ORIGIN aula2.ies2.net.
$GENERATE 1-16   equipo$   A   192.168.2.$
```

En el caso del `ies2.net` tenemos dos máquinas para un mismo nombre de dominio (el correspondiente al servicio Web). En el ejercicio se nos pedía que la resolución fuera equilibrada, es decir que al escribir en un navegador Web `http://www.ies2.net` el servidor de nombres resolvería unas veces con la dirección 192.168.65.3 y otras con la 192.168.65.4. Para conseguir esto basta con escribir, tal y como se ha hecho, varios registros seguidos con el mismo nombre. BIND se encarga automáticamente de ir alternando la resolución, produciendo así el balanceo deseado.

2.9. Servidor de nombres de dominio esclavo

Un servidor de nombres de dominio no tiene porqué tener completa autoridad sobre una zona que resuelve, y sin embargo, puede contener el archivo de zona correspondiente. Este tipo de servidor se denomina esclavo, o secundario. Tendrá autoridad sobre las zonas de las que es esclavo, ya que posee los archivos de configuración de zona. La autoridad no es completa porque cualquier modificación sobre los mismos no conlleva una actualización[28] en el resto de los DNS. Los archivos de zona contenidos en los DNS esclavos son una réplica de los existentes en los servidores de nombres maestros que poseen completa autoridad sobre esas zonas.

Normalmente los archivos de zona son transferidos desde el servidor maestro, pero podrían ser transferidos desde otro DNS secundario. Es decir pueden existir DNS esclavos que a su vez dependen de otros DNS esclavos. Estos archivos serán recogidos por `named` que los almacenará en el lugar que le indiquemos, normalmente en `/var/named/`.

¡¡Importante!! Cuando es instalado BIND se crea el usuario `named` (NAME Daemon), que es el demonio o dueño del proceso en background encargado de dar el servicio `DNS`. Por esta razón, los ficheros asociados a las zonas esclavas deben ser accesibles por el usuario del sistema `named`, ya que no sólo los debe leer sino que también tiene que modificarlos.
Al hacer una consulta o modificación este usuario accede a los ficheros de configuración asociados a las zonas como lo hace cualquier otro usuario del sistema, con las correspondientes restricciones de acceso, permisos de lectura y escritura sobre directorios y archivos. Por tanto, debemos asegurarnos de que el usuario `named` tenga permisos sobre el fichero de configuración asociado.

Cuando se produce una modificación en el archivo de zona, los servidores secundarios llamados para actualizarse son aquellos que el servidor maestro tiene recogidos en la cláusula `also-notify`[29] y los que se definen por medio del registro `NS` en el caso de que `notify` este puesto a `yes`.

Los archivos de zona se almacenan exactamente igual que lo hace un servidor maestro. La diferencia con el servidor maestro es que el esclavo estará comprobando continuamente el `periodo`

[28] Observa que cuando en un DNS maestro se modifica algún archivo de zona, se informa a todos los servidores que tienen autoridad sobre la zona, para que hagan las actualizaciones oportunas.

[29] Las cláusulas `also-notify` y `notify` están descritas en la página 53.

`de refresco`[30] y cuando este haya transcurrido solicitará una actualización. Esta será efectiva si el `número de serie` del registro `SOA` ha cambiado.

Las zonas transferidas desde el DNS maestro se deben almacenar en un fichero. El nombre y el lugar de almacenamiento de este fichero se define a través del parámetro `file`.

Para comprender mejor lo que se acaba de describir vamos a declarar una nueva zona llamada `ieslosenlaces.com` sobre la cual no tiene autoridad nuestro equipo servidor de nombres, sino otro equipo servidor de la red con dirección IP 192.168.199.136:

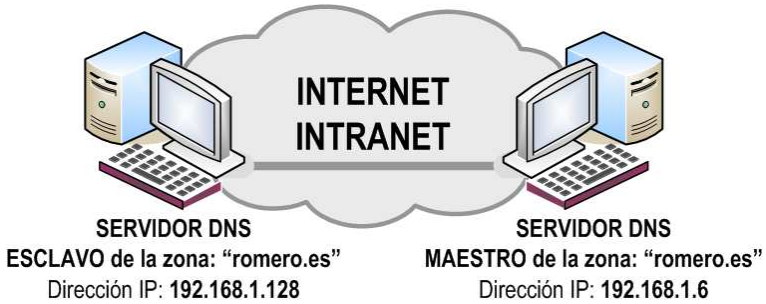
```
zone   "ieslosenlaces.com" in {
       type slave;
       masters {192.168.199.136;};
       file "esclava.ieslosenlaces.com"; /*ruta relativa del fichero
                  de configuración que almacenará la información de zona*/
};
```

Observa que para que el servidor secundario solicite la actualización es necesario que conozca la dirección IP del maestro. Esta dirección IP se indica por medio del parámetro `masters` *`dirección IP`*, ya comentado en la página 56.

Por tanto, `masters` es el parámetro que nos permite indicar a BIND cual es el servidor DNS al que tendrá que consultar información sobre el dominio `ieslosenlaces.com` en el caso en que algún cliente requiera de la resolución de algún nombre asociado a él. La información suministrada será almacenada en el fichero de configuración indicado por `file`, lo que le facilitará la resolución de posteriores peticiones.

Ejercicio 2.2

Implementa la estructura mostrada en la figura. En la que el equipo con IP 192.168.1.6 funciona como DNS maestro de la zona correspondiete al dominio `romero.es` y el equipo con IP 192.168.1.128 es un DNS esclavo de la anterior zona.



La zona `romero.es` dispone de :

www.romero.es	⇒	192.168.1.1
ftp.romero.es	⇒	192.168.1.2
equipo1.romero.es	⇒	192.168.1.3

1. Configura ambos equipos para cumplir con los requisitos impuestos. Recuerda que debes crear un archivo de zona vacío para el servidor esclavo (`touch /var/named/esclava.romero.es`).

2. Arranca el servicio en los dos equipos. Primero en el maestro y después en el esclavo. A continuación lista el contenido del archivo de zona esclava ¿Está vacío?

[30] El `periodo de refresco` es uno de los parámetros del registro `SOA`, que fue explicado en la sección 2.7, página 60

3. Comprueba que el esclavo resuelve los nombres de dominio asociados a la zona esclava.

Solución del ejercicio 2.2

El archivo `/etc/named.conf` del servidor maestro deberá presentar la siguiente estructura,

```
options  {
         directory "/var/named";
         };
zone     "romero.es" in {
         type master;
         file "maestra.romero.es";
         };
```

Y según las especificaciones marcadas su correspondiente archivo de zona será:

```
$TTL 4h

@                   IN SOA  ns           root.localhost.
                                         (3001200602;
                                         21600;
                                         10800;
                                         604800;
                                         21600; )
                    IN NS   ns
ns                  IN A    192.168.1.6
www                 IN A    192.168.1.1
ftp                 IN A    192.168.1.2
equipo1.romero.es.  IN A    192.168.1.3
```

El contenido del `/etc/named.conf` del servidor esclavo será:

```
options  {
         directory "/var/named";
         };
zone     "romero.es" in {
         type slave;
         masters {192.168.1.6;};
         file "esclava.romero.es";
         };
```

Deberemos crear el archivo de la zona esclava, aunque no tendrá ningún contenido. Además deberemos cambiar la propiedad del mismo y dársela a `named`:

```
[root@Linux]# touch /var/named/esclava.romero.es
[root@Linux]# chown named.named /var/named/esclava.romero.es
```

A continuación arrancamos los servicios en las dos máquinas. Así como se inicia el servicio en el servidor esclavo podemos hacer un `cat` del archivo `/var/named/esclava.romero.es` para comprobar que la información de zona ha sido transferida:

```
[root@localhost named]# cat esclava.romero.es
$ORIGIN .
$TTL 14400 ; 4 hours
romero.es               IN SOA  ns.romero.es.        root.localhost. (
                                1120061 ; serial
                                21600 ; refresh (6 hours)
                                100 ; retry (1 minute 40 seconds)
                                1280000 ; expire (2 weeks 19 hours)
                                14400 ; minimum (4 hours)
                                )
                        NS      servidor.romero.es.
$ORIGIN romero.es.
ns                      A       192.168.1.6
www                     A       192.168.1.1
ftp                     A       192.168.1.2
equipo1.romero.es       A       192.168.1.3
```

Por último se puede comprobar que la resolución funciona mediante `dig`:

```
[user@linux]$ dig www.romero.es

; <<>> DiG 9.3.1 <<>> romero.es
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44758
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;www.romero.es.  IN        A

;; ANSWER SECTION:
www.romero.es.   14400 IN  A   217.76.130.84

;; AUTHORITY SECTION:
romero.es.       14400 IN  NS  ns.romero.es.

;; ADDITIONAL SECTION:
ns.romero.es.    14400 IN  A   192.168.1.6

;; Query time: 1 msec
;; SERVER: 192.168.1.128#53(192.168.1.128)
;; WHEN: Wed Jul 19 17:55:26 2006
;; MSG SIZE rcvd: 86
```

2.10. Servidor de nombres de dominio forward

Este servidor hace preguntas sobre las zonas que tiene configuradas a otro DNS que comúnmente se denomina *forwarder*. Los `forwarders` son empleados normalmente cuando se desea que no todos los DNS de un sitio interactúen con el resto de servidores de Internet. Un ejemplo típico podría ser el siguiente: imagina una red con un firewall de salida a Internet y varios DNS, de los cuales todos menos uno son incapaces de pasar paquetes a través de dicho firewall. De esta forma, todos los servidores preguntan al DNS con autorización para salir a Internet, quien a su vez se encarga

de hacer la pregunta al exterior en nombre de ellos. La ventaja de esta implementación es que este servidor puede dar servicio cache a todos los demás.

La zona `forward` puede contener las cláusulas `forward` y `forwarders` (ver página 53) que indicarán los equipos a los cuales se les reenviarán las preguntas sobre la zona en cuestión. Es necesario establecer los `forwarders` en la zona ya que si esta lista estuviera vacía no se haría forwarding, aunque se hubiesen definido de forma general en el estamento `options`.

Observa que este tipo de servidor ni tiene, ni necesita un archivo de configuración de zona. No es él quien hace la resolución, otro la hace por él; será este último el que necesitará el correspondiente archivo de zona.

2.10.1. Contenido de `/etc/named.conf`

Un servidor de reenvío se diferencia de otro tipo de servidor en la defición de las zonas. La estructura de una zona para un servidor de este tipo es:

```
zone  "nombre del dominio a resolver" in {
      type forward;
      forward only|first;
      forwarders {lista de los DNS que nos servirán};
};
```

Con la cláusula `forward` es posible redefinir lo estipulado en el estamento `options` (cambiar de `first` a `only` o viceversa).

Un ejemplo podría ser,

```
zone  "ingemartin.es" in {
      type forward;
      forwarders {194.224.52.4;194.224.52.6;};
};
```

De esta forma cuando hagamos una petición de resolución del dominio `ingemartin.es` el DNS preguntará en primer lugar al DNS con IP 194.224.52.4 esperando respuesta, si este no le responde la pregunta pasará a 194.224.52.6.

Si deseamos que nuestro DNS responda a cualquier petición utilizando los `forwarders` bastaría con que el fichero `/etc/named.conf` contuviera:

```
options {
          directory /var/named";
          forwarders {194.224.52.4;194.224.52.6;};
};
```

Donde al no incluir la cláusula `forward` se tomará su valor por defecto: `first`.

Ejercicio 2.3

Agrupaos de tres en tres equipos o utiliza tu propio equipo con dos máquinas virtuales para implementar la siguiente configuración:

1. Una de las máquinas hará de cliente y tendrá la dirección 192.168.19.x (la x se corresponde con el número de tu equipo). Este equipo tendrá como DNS a la segunda de las máquinas cuya dirección será 192.168.19.x+50.

2. La segunda máquina (con IP 192.168.19.x+50) estará configurada como servidor de reenvío para cualquier zona que se le solicite.

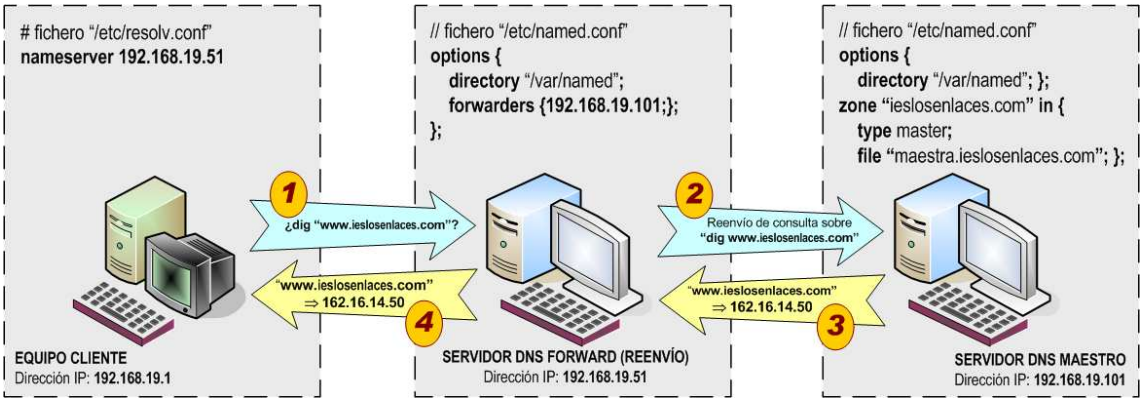
3. El tercer equipo, que tendrá dirección IP 192.168.19.x+100 será un DNS con autoridad sobre la zona `ieslosenlaces.com`.

Una vez realizada la anterior configuración comprueba mediante `dig` que el equipo 192.168.19.x+50 recibe una respuesta al preguntar por el nombre `www.ieslosenlaces.com` asociado a la dirección IP 162.16.14.50.

Solución del ejercicio 2.3

La configuración solicitada en el ejercicio queda resumida en la siguiente figura. En ella se muestran los pasos seguidos cuando desde un equipo cliente (192.168.19.1) se hace un `dig` a otro equipo a través de su nombre de dominio (`www.ieslosenlaces.com`)

```
[user@linux]$ dig www.ieslosenlaces.com
```



En la figura se muestran los contenidos de los archivos de configuración. Falta por mostrar el archivo de zona `/var/named/maestra.ieslosenlaces.com`, cuyo contenido podría ser:

```
$TTL 4h
@       IN SOA  ns                      root.localhost.
                                        (3001200602
                                        21600
                                        10800
                                        604800
                                        21600 )

;Comenzamos a escribir los registros:
        IN NS   ns.ieslosenlaces.com.
ns      IN A    192.168.19.101
www     IN A    162.168.14.50
```

Tras hacer la consulta `dig` desde el equipo con dirección 192.168.19.1 se obtiene como respuesta:

```
[user@linux]$ dig www.ieslosenlaces.com

; <<>> DiG 9.3.1 <<>> www.ieslosenlaces.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 31295
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;www.ieslosenlaces.com.   IN        A

;; ANSWER SECTION:
www.ieslosenlaces.com.    14400 IN   A    162.16.14.101

;; AUTHORITY SECTION:
ieslosenlaces.com.        14400 IN   NS   ns.ieslosenlaces.com.

;; Query time: 3 msec
;; SERVER: 192.168.19.51#53
;; WHEN: Wed Jul 14 20:15:42 2006
;; MSG SIZE rcvd: 100
```

2.11. Resolución inversa

Hasta este momento se ha estudiado como el servidor de nombres de dominio convierte los nombres a direcciones IP. Pero un DNS también es capaz de transformar una determinada dirección IP en un nombre, esto se denomina *resolución inversa*.

Pueden darse situaciones en las que es necesaria este tipo de resolución. Por ejemplo la descrita en el capítulo 5, cuando un servidor de correo hace una petición de resolución inversa para comprobar si una máquina tiene un nombre de dominio cualificado (FQDN). En este capítulo no nos preocuparemos por las situaciones en las que la resolución inversa puede ser útil y simplemente estudiaremos como configurar BIND para que sea capaz de hacerlas.

La resolución inversa se consigue por medio del dominio `in-addr.arpa` y el registro de recurso `PTR`.

En primer lugar se debe definir en `/etc/named.conf` una zona de resolución inversa. Un ejemplo del contenido de este fichero en lo referente a la zona que permitirá a BIND resolver las direcciones de la red 192.168.19.0 podría ser:

```
zone "19.168.192.in-addr.arpa"{
  type master;
  file "/var/named/inversa.19.168.192"; };
```

La lectura de la dirección IP por BIND se realiza desde el octeto menos significante al más significante, al revés de como normalmente se escribe. Basándonos en el anterior ejemplo, cuando el servidor recibe una petición de resolución inversa, BIND busca un servidor `arpa`, a continuación un servidor `in-addr.arpa`, después uno `192.in-addr.arpa`, posteriormente `168.192.in-addr.arpa` y por último el servidor con autoridad para la zona `19.168.192.in-addr.arpa`. Cuando lo ha encontrado se dirige al fichero de configuración de zona para leer el registro que le permita responder a la pregunta.

El archivo de configuración de zona se llama en este ejemplo `/var/named/inversa.19.168.192` y podría tener una forma similar a la mostrada a continuación:

```
$TTL 1d

@                           IN SOA  ns                              root.localhost.
                                                                    (1107200604;
                                                                    21600;
                                                                    10800;
                                                                    604800;
                                                                    21600; )
                            IN NS   ns

1.19.168.192.in-addr.arpa.  IN PTR  equipo1.taller19.cossio.com.
2.19.168.192.in-addr.arpa.  IN PTR  equipo2.taller19.cossio.com.
3                           IN PTR  equipo3.taller19.cossio.com.
4                           IN PTR  equipo3.taller19.cossio.com.
...
```

Como el dominio es `19.168.192.in-addr.arpa` es posible escribir el número del equipo simplemente (sin ir seguido del punto .), siguiendo una notación similar a la mostrada en ejemplos anteriores en los que se hacía una resolución *directa*.

Para comprobar el funcionamiento se puede utilizar `dig`, pero en este caso se debe colocar la opción `-x` que indica solicitud de resolución inversa.

```
[user@linux]$ dig -x 192.168.19.1

; <<>> DiG 9.3.1 <<>> -x 192.168.19.1
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38586
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
1.19.168.192.in-addr.arpa.  IN PTR

;; ANSWER SECTION:
1.19.168.192.in-addr.arpa.  86400 IN  PTR  equipo1.taller19.cossio.com.

;; AUTHORITY SECTION:
1.19.168.192.in-addr.arpa.  86400 IN  NS   ns.cossio.com.

;; Query time: 56 msec
;; SERVER: 192.168.0.5#53
;; WHEN: Wed Jul 19 16:56:57 2006
;; MSG SIZE rcvd: 112
```

Capítulo 3

CONFIGURACIÓN DE UN SERVIDOR WEB: APACHE

3.1. Introducción al servicio HTTP

A pesar de tratarse de uno de los servicios más jóvenes que existen (surge en 1990), sin lugar a dudas, HTTP es el servicio ofrecido en Internet con una mayor aceptación, generalmente conocido como servicio Web. En la actualidad, nos permite de una manera muy sencilla acceder a todo tipo de información[1] convirtiendo a Internet en un medio de comunicación sin precedentes. Originalmente, todos los contenidos que podían encontrarse erán estáticos, caracterizándose por ofrecer siempre la misma información independientemente de condicionantes externos, como podrían ser: el momento del acceso, la persona que lo realiza, o el lugar y equipo desde donde se lleva a cabo. En la actualidad, este servicio se ha visto fuertemente potenciado gracias al dinamismo que aporta a los contenidos, llegando a ofrecer al usuario servicios tan dispares como acceder a bases de datos, controlar equipos remotos[2], consultar el correo electrónico, establecer rutas óptimas de carretera en base a determinados criterios, y todo ello, vía Web.

Toda esta información se encuentra almacenada en equipos servidores distribuídos a través de la red. Está organizada en documentos denominados páginas Web enlazadas entre sí formando los llamados sitios Web. Estos sitios Web son accesibles desde aplicaciones clientes, denominadas habitualmente navegadores Web. De esta forma se establece una jerarquía dentro de Internet entre ambos tipos de equipos (servidores y clientes), cada uno caracterizándose por desempeñar funciones concretas:

1. Los servidores se encargan de almacenar y gestionar los contenidos de los sitios Web. Además atienden las solicitudes emitidas por los clientes sobre estos.

2. El cliente tan sólo se encarga de tramitar las peticiones de conexión hacia los servidores solicitando los contenidos de los sitios Webs que tienen alojados.

Sus diferencias radican únicamente en el software del que hacen uso[3], ya que no existen aplicaciones software que, al mismo tiempo, nos permitan tanto consultar como servir páginas Web. En cambio, ambas aplicaciones tienen algo en común, el protocolo del que tienen que hacer uso para ponerse de acuerdo. De otra forma, sería imposible establecer comunicación desde un cliente con un servidor. Concretamente, el protocolo que utilizan es el HTTP (Hypertext Transfer Protocol, Protocolo de Transferencia de Hipertexto), el cual da nombre al servicio. Este protocolo, recibió su nombre debido a que la información que se transmitía a la red era algo más que texto, era hipertexto. Además del texto, contenía imágenes, sonidos o videos. Además es importante tener en cuenta que este protoloco forma parte de la capa de aplicación del modelo TCP/IP.

[1] Es muy sencillo encontrar vía Web el estado de las carreteras, pronóstico del tiempo, ciencia, historia, ...

[2] Con acceso remoto a equipos queremos decir: cambios en la configuración, modificación de ficheros, control del estado de servicios, ... en otro equipo distante.

[3] Por ejemplo, un software servidor es `apache`, y un software cliente `mozilla firefox`

La primera versión del protocolo HTTP, v0.9, aparece en 1990. Inicialmente se trataba de un protocolo muy simple pensado para la transmisión de datos a través de Internet en formato digital, que pronto fue modificado apareciendo una nueva versión, v1.0, definido en el RFC1945, mejorando algunas de las características de su antecesor. Sin embargo, con el tiempo, se advirtió que seguía presentando una seríe de limitaciones que no se correspondían con la demanda que se avecinaba. Entre estas limitaciones cabe destacar la imposibilidad de poder servir más de un sitio Web por equipo servidor (hoy en día equivaldría a decir que existen tantos equipos servidores Web como sitios Web accesibles, algo prácticamente imposible), lo cual se solucionó en su siguiente versión, v1.1, mediante la implementación de `Hosts Virtuales`. Esta última versión, v1.1, fue normalizada en junio de 1999 en el RFC2616 (`http://www.ietf.org/rfc/rfc2616.txt`) ofreciendo una solución a todos los inconvenientes que presentaban sus antecesores y, aún hoy, sigue vigente.

Por último señalar que el servicio HTTP forma parte de la lista de servicios orientados a conexión[4], por lo que hace uso del protocolo TCP (Transmission Control Protocol, Protocolo de Control de Transfercia) dentro del nivel 4 (capa de transporte) del modelo TCP/IP.

3.2. El servidor Web APACHE

Apache es el software servidor HTTP más ampliamente utilizado en el mundo, al contar con cerca del 70 % del servidores Web que existen en el mundo. Si tenemos en cuenta su corta trayectoria (surge en 1995) y la aceptación tan enorme que ha ido teniendo en todo este tiempo, se augura un futuro muy prometedor con unas cuotas cercanas al 90 %, en detrimento del resto de software disponible en el mercado. Tan sólo el software servidor ofrecido por Microsoft (`IIS`, Internet Information Server) se vislumbra como un posible competidor, aunque con muchas limitaciones, debido a las vulnerabilidades que presenta.

Apache aparece públicamente en abril de 1995 (v0.6.2) como una alternativa al servidor del NCSA[5] `httpd` cuyo desarrollo quedó definitivamente suspendido en 1998, reaprovechándose prácticamente todo su código. Ofrece el código abierto (software libre) y posee unas estraorinarias características, entre las que podrían destacarse:

- *Portabilidad:* Están disponibles versiones de apache para muchos de los sistemas operativos que existen (Microsoft Windows, GNU/Linux, UNIX, MacOS, ...).
- *Facilidad de configuración:* Mediante un simple editor de textos, modificando únicamente un único fichero (httpd.conf), es posible decidir el valor de las directivas de configuración del servicio, y por tanto, determinar el comportamiento del servidor.
- *Modularidad:* Cuenta con un gran número de paquetes de funciones (directivas) llamados modulos DSO (Dynamic Shared Objects) ofreciendo al administrador del servidor Web multitud de opciones de configuración aumentando así su versatilidad.

  Añadiendo los modulos adecuados, apache nos ofrece la posibilidad de interacturar de una manera muy sencilla con sistemas gestores de bases de datos (DBM, DataBase Management).
- *Web dinámicas:* Ofrece soporte para todo tipo de sitios Web dinámicos (comandos SSI, scripts CGI, PHP, ...)
- *Comunicaciones seguras:* Mediante el uso de algoritmos criptográficos nos permite establecer comunicaciones seguras (confidencialidad y autentificación) haciendo uso del protocolo SSL (Secured Socket Layer, Seguridad de la Capa de Transporte).
- *Servicio multiweb:* Mediante el uso de Hosts Virtuales basados en dirección IP y en nombre de dominio nos ofrece la posibilidad de servir diferentes sitios Webs.

[4] Se requiere un establecimiento previo de la comunicación antes de que se produzca la transferencia de los datos.
[5] National Center for Supercomputing Applications, Centro Nacional de Aplicaciones de Supercomputación.

- *Servicio proxy:* En caso de requerirlo, integra un servidor Proxy por si necesitaramos de un intermediario entre nuestro servidor y el resto de la red.

Actualmente la ASF (Apache Software Foundation, Fundación del Software Apache) es la organización no lucrativa encargada de gestionar y desarrollar el proyecto apache. Al tratarse de un código abierto (cualquiera es libre de manipular su código) esta organización esta compuesta por una comunidad de usuarios expertos que de manera descentralizada colaboran en ello (`http://www.apache.org`).

Como se verá en el desarrollo de la práctica, existen dos versiones de este software ampliamente distribuidas: v1.3 y la v2. Aunque la versión 2, al ser posterior, presenta unas características más fiables que las correspondientes a la 1.3. Esta última sigue siendo ampliamente utilizada en multitud de servidores Web que son reacios al cambio. Esto hace que en la práctica ambas versiones sigan desarrollándose en paralelo, no centralizando todos los esfuerzos en su última versión.

3.3. Contenido del capítulo

En este capítulo tratarán de cubrirse los siguientes aspectos:

- Transformar nuestro equipo GNU/Linux en un servidor Web: instalación de apache.
- Configurar el servicio apache para dar soporte a más de un sitio Web: implementación de Hosts Virtuales basados en IP y en nombre de dominio.
- Restringir el acceso de manera total o parcial en determinados sitios Web: sitios Web no anónimos.
- Dinamizar los sitios Web servidos desde apache: comandos SSI, scripts CGI y PHP.

Se propondrán múltiples ejercicios prácticos proporcionado junto a su enunciado una posible solución. Para la implementación y comprobación de todos estos ejercicios prácticos se asumirá la disposición de una configuración en red con la que poder hacer pruebas. En aquellos casos donde se disponga de suficiente equipación para ello cabría recordar que VMware es una buena opción:

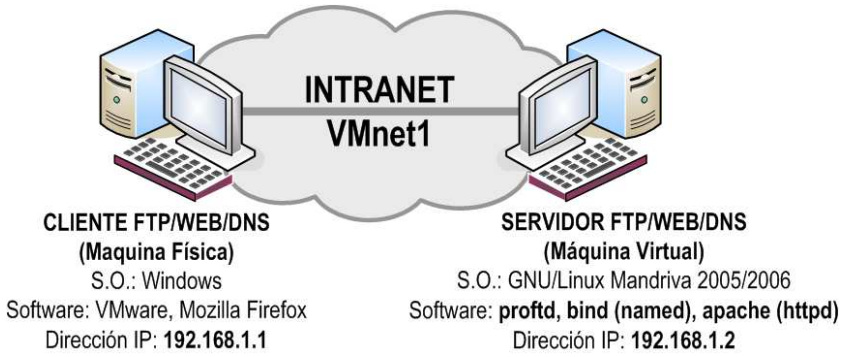


Figura 3.1: VMware nos ofrece la posibilidad de hacer pruebas en red pudiendo probar las configuraciones realizadas en la práctica.

Se recomienda leer el apéndice A para conocer algunas de las posibilidades de VMware.

Por útlimo señalar que a lo largo de los ejercicios se tratan de detallar absolutamente todos los pasos necesarios para su implementación. Aunque determinados aspectos, por cuestiones de extensión, se darán por supuestos como, por ejemplo, algunos comandos asociados al sistema operativo GNU/Linux y el lenguaje de etiquetas HTML.

PRÁCTICA: CONFIGURACIÓN DE UN SERVIDOR WEB

3.4. Instalación del software APACHE.

En esta sección configuraremos nuestro equipo GNU/Linux para que haga las veces de servidor Web. De esta forma le proporcionaremos la capacidad de atender peticiones HTTP realizadas desde algún cliente `Web` (`konqueror`, `mozilla-firefox`, `Internet Explorer`, ...) sobre algún contenido Web que hayamos alojado previamente en el servidor.

Necesitamos instalar un software específico, `apache`. En concreto instalaremos el paquete `apache2` (en lugar del paquete RPM `apache`) que se suministra con la distribución de GNU/Linux, el cual se corresponde con una segunda versión del servidor `apache` mejorada que permite su configuración de una manera más simplificada, además de aportar características interesantes: soporte para PHP (PHP4/5), autentificación externa (mod-auth_external), IPv6, ...

¡¡Aclaración!! Existen dos versiones ampliamente utilizadas del software servidor Web `apache`, la v1.3 y la v2. La versión 2 de `apache` surgió como resultado de llevar a cabo una serie de modificaciones en la versión 1.3 con la finalidad de mejorar sus prestaciones. Por este motivo, nosotros instalaremos y configuraremos su versión 2, pero se tratará al mismo tiempo de recalcar a lo largo de la práctica todas las características nuevas que presenta esta versión para saber a que atenernos en el caso de toparnos con un apache v1.3. Es importante tener esto presente, ya que como veremos, la configuración del servidor `apache`, al igual que el resto de servicios (FTP, DNS, etc.) que se presentan en el libro, depende únicamente de las directivas de configuración que se incorporen y del valor que se les asigne a estas, presentando la versión 2 directivas diferentes a las que se hacia uso en la v1.3, ya sea porque se han renombrado, han desaparecido o simplemente se han incorporado con la nueva versión del servidor. Aclarar por último, que ambas versiones siguen estando en desarrollo, ya que aunque la migración de una versión a otra es posible, muchos antiguos servidores siguen funcionando con la v1.3 y son reacios al cambio.

Tal y como se ha ido comentando en anteriores capítulos, la instalación puede hacerse usando la herramienta gráfica `rpmdrake` o a través de la interfaz de línea de comandos mediante `urpmi`. Ambas formas de instalación no son únicas, pero son las más aconsejables, ya que las dos se preocupan tanto de instalar el software especificado, como de comprobar las dependencias con otros paquetes.

1ª Forma: A través de la interfaz gráfica mediante el uso de `rpmdrake`. Ver figura 3.2.a).

```
[root@linux]# rpmdrake &
```

2ª Forma: Desde la línea de comandos o `CUI`. Esta es la opción más aconsejable por su rapidez, comodidad y la no necesidad de una interfaz gráfica. Ver figura 3.2.b).

```
[root@linux]# urpmi apache2
```

Una vez instalado el software `apache`, nuestro equipo GNU/Linux ya es capaz de servir el sitio Web que le indiquemos. En caso de no realizar modificaciones sobre la configuración predeterminada de `apache`, este dispone de un sitio Web de prueba que nos permitirá comprobar el correcto funcionamiento del servicio, para lo cual, tan sólo será necesario invocarlo. Antes de ello, deberemos arrancar el servicio[6] `httpd` para que el equipo atienda solicitudes `http`:

[6] Como ya ha sido comentado en anteriores ocasiones no es necesario utilizar el comando `service`:

```
[root@linux]# /etc/init.d/httpd start|restart
```

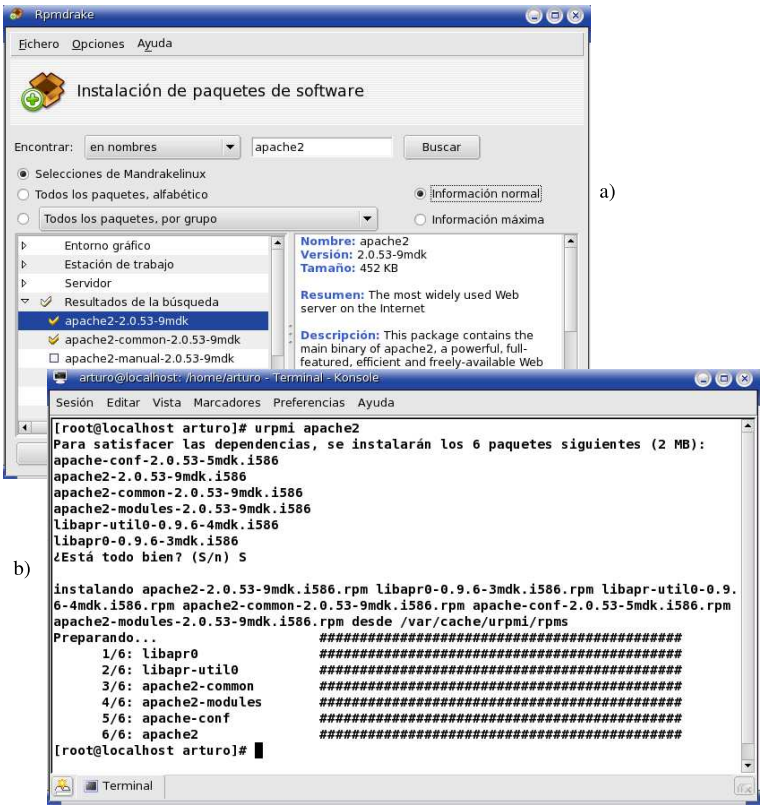


Figura 3.2: a) Instalación de `apache` desde: a) La interfaz gráfica de Mandriva, b) La línea de comando.

```
[root@linux]# service httpd start
```

Otras formas de arrancar/reiniciar el servidor `apache` son las siguientes:

1. Haciendo uso del comando `httpd` o `httpd2` (Apache Hypertext Transfer Protocol Server). Este comando como veremos nos permite controlar y configurar el servicio (observar la ayuda, `man httpd/httpd2`):

```
[root@linux]# httpd/httpd2 -k start
```

2. Utilizando su interfaz de control `apachectl` (http Server Control Interfaces) pasándole como argumento `start` o `restart`. Este comando es el que comúnmente se ha utilizado para ello en todas las versiones de `apache`, pero algunas distribuciones GNU/Linux en sus últimas adaptaciones (por ejemplo, `Mandriva 2006`) ya no lo implementa:

```
[root@linux]# apachectl start
```

A continuación, si el arranque del servicio no ha generado ningún tipo de error, para establecer conexión con el servidor `apache` tan sólo será necesario ejecutar un cliente Web y colocar como URL: `http://`*`direccion IP`* | *`alias`* | *`nombre dominio del servidor`*.

```
http://localhost/
```

En la distribución GNU/Linux Mandriva 2006 la página de inicio ha cambiado, apareciendo únicamente una señal de alerta como que `apache` funciona correctamente: `It Works!`.

A partir de este momento ya podemos *colgar* al menos un sitio Web para que sea servido por `apache` y visualizado desde nuestro cliente Web preferido.

3.5. Configuración básica de APACHE

En este apartado se mostrará de qué manera podemos configurar `apache` para que nuestro equipo GNU/Linux ofrezca servicio Web de una manera personalizada. Para ello deberemos editar mediante nuestro editor preferido (vi/kwrite/kate/gedit) el fichero de configuración de `apache` `/etc/httpd/conf/httpd2.conf` (en Mandriva 2005).

¡¡Aclaración!! En el caso de que haya sido instalado el paquete RPM `apache` (versión 1.3 de `apache`), en lugar de `/etc/httpd/conf/httpd2.conf`, el fichero de configuración que deberemos editar será `/etc/httpd/conf/httpd.conf`, el cual presenta una sintaxis similar. El digito `2` añadido por GNU/Linux Mandriva (hasta la distribución 2005) al final del nombre del fichero de configuración de apache tan sólo trata de diferenciar entre el correspondiente a la versión 1.3 (httpd.conf) y el de la v2 (httpd2.conf). Esta distinción explicita es importante, ya que ambos ficheros no admiten los mismos parámetros al existir algunos de ellos que son reconocidos por una de las versiones y no por la otra.
En otras distribuciones GNU/Linux (Debian, Red Hat, Suse, etc.) el nombre del fichero de configuración de `apache` `httpd.conf` se ha respetado y no se ha incluido ningún distintivo en función de la versión. Los propios desarrolladores de Mandriva han devuelto a su fichero de configuración su nombre original `httpd.conf` en la distribución de 2006.

3.5.1. Estructura del archivo de configuración de APACHE

Tras la instalación de `apache` se genera un fichero predeterminado de configuración `/etc/httpd/conf/httpd2.conf` totalmente operativo, que podremos modificar para personalizar el servicio `httpd` a nuestro gusto. Para ello, será necesario conocer de que manera está estructurado, y cual es el significado de cada una de las líneas que aparecen.

¡¡Importante!! La estructura interna del fichero de configuración de `apache`, `httpd/httpd2.conf`, depende de la versión (v1.3 o v2), y de la distribución de GNU/Linux (o Windows) bajo la que esté trabajando.

La estructura presentada aquí es la que utiliza `Mandrake/Mandriva` en su versión 2005 (Limited Edition). En cualquier caso, las directivas de configuración son prácticamente las mismas[7] diferenciándose únicamente en la posición que ocupan dentro del fichero. Sólo cambia la organización[8].

Como podrá observarse, una vez abierto el fichero de configuración del servidor `apache` `/etc/httpd/conf/httpd2.conf`, a grandes rasgos, la estructura de este, es la siguiente[9]:

1^{er} **Bloque.- Directivas de configuración general.** Entre las directivas disponibles, las hay que son propias del núcleo de `apache` (módulo `core`), pero la mayor parte de ellas pertenecen a diferentes módulos que son agregados expresamente a `apache` y que nos permiten aumentar las posibilidades en su configuración, y por tanto, la potencia del servidor.

[7] Si bien es cierto que existen directivas de la v1.3 que se han deshabilitado en la v2.

[8] Debes tener en cuenta que el orden en que son colocadas las directivas no influye y que aunque aquí se utiliza la estructura del fichero `httpd/httpd2.conf` de Mandriva 2005 las explicaciones son válidas para cualquier otra estructura

[9] Las líneas precedidas de `#` son comentarios de ayuda y pasan desapercibidas en la configuración del servidor `apache`

```
#1ER BLOQUE
#Directivas de configuración general del servicio httpd
ServerRoot /etc/httpd/2.0
DocumentRoot /var/www/html
Servername localhost
PidFile /var/run/httpd.pid
ErrorLog logs/error_log
Loglevel warn
...
```

Aunque existen muchas directivas de configuración general, a continuación se detallarán algunas de ellas que resultarán de utilidad para comprender el contenido predeterminado del fichero de configuración `httpd2.conf` o `httpd.conf`. En la práctica se irán presentando nuevas directivas de configuración del servidor `apache`, incrementando poco a poco la complejidad del fichero de configuración del servidor al mismo tiempo que personalizamos su funcionalidad.

Directivas generales de `apache`

- **ServerRoot *ruta***

Esta directiva informa del directorio raíz del que *cuelgan* los directorios de configuración/auditoria del servidor `apache`.
Teniendo en cuenta que al instalar `apache2` el fichero predeterminado `httpd2.conf` (o `httpd.conf`) que se genera le ha asignado un valor por defecto (`ServerRoot /etc/httpd/2.0` en Mandriva 2005) totalmente válido y operativo, su valor raramente es necesario modificarlo. Toda ruta relativa que aparezca en la configuración del servicio, será en relación a el directorio asignado a la directiva `ServerRoot`.

- **ServerName *nombre***

Indica el nombre asociado al sitio Web a servir. Cuando `apache` sólo tiene asociada una página Web por dirección IP, la directiva `ServerName` únicamente tiene un carácter descriptivo.
Ejemplo:

```
ServerName www.ingemartin.es
```

En realidad esta directiva es únicamente útil cuando `apache` está configurado para servir diferentes sitios Web asociados a una misma dirección IP. Cuando `apache` hace esto decimos que tiene configurados *Hosts Virtuales Basados en Nombre*.
Con el nombre asignado a `ServerName` se identifica cada uno de los hosts virtuales. El protocolo HTTP además de portar la dirección IP del sitio Web solicitado, también contiene el nombre asociado a la página (el dado en `ServerName`), de esta forma cuando el servidor `apache` recibe la petición es capaz de diferenciar cual de los sitios Web que puede servir con esa dirección IP es el solicitado.
En el caso de configurar hosts virtuales basados en nombre, además de esta directiva, `apache` también necesita de `NameVirtualHost`; todo esto se verá con detalle en el apartado 3.7.2.

- **PidFile *ruta***

Establece la ruta absoluta del fichero donde se almacenará el identificador del proceso, `PID`, asociado al servicio `httpd`.
Ejemplo:

```
PidFile /var/run/httpd.pid
```

Su valor puede resultar útil cuando otros procesos quieren mandarle señales. Por ejemplo en caso de querer `matarlo`:

```
[root@linux]# kill -9 `more /var/run/httpd.pid`
```

¡¡Ojo!!, este comando sólo mataría al proceso padre, pero no a los procesos hijos de éste, por lo que apache seguiría funcionando sobre estos.

- **ErrorLog** ***ruta***

Ruta del fichero donde se auditan los errores detectados por el servicio `httpd`. En el caso de que dicha ruta sea relativa, por ejemplo `ErrorLog logs/error_log`, se establece en relación al directorio indicado en `ServerRoot /etc/httpd/2.0`. Es decir, que sería equivalente a escribir:

```
ErrorLog /etc/httpd/2.0/logs/error_log
```

- **LogLevel** ***nivel***

Los ficheros `log` o de auditoria permiten registrar tanto los accesos al servidor `apache` como los errores detectados en su funcionamiento. Los errores se almacenan en `/etc/httpd/2.0/logs/error_log`.
Nota: También pueden ser almacenados en `/var/log/httpd/error_log`.

Por su parte, los accesos son almacenados en `/etc/httpd/2.0/logs/access_log`.
Nota: El fichero de *log* para los accesos también es posible que sea almacenado en `/var/log/httpd/access_log`.

La información almacenada en ellos puede ser escueta, o por el contrario, muy extensa y detallada. El parámetro `LogLevel` nos va a permitir ajustar la cantidad de información que será almacenada en estos ficheros en función del nivel escogido: `debug`, `info`, `notice`, `warn`, `error`, `crit`, `alert` y `emerg`. Un nivel razonable es el que se establece por defecto,

```
LogLevel warn
```

- **DocumentRoot** ***ruta***

Informa a `apache` donde se encuentra el directorio raíz que contiene el sitio Web que debe servir en caso de recibir una petición `http` desde un cliente Web (por ejemplo, `mozilla-firefox`). Por defecto, ante tal solicitud, el servidor Web le devolverá al cliente la página de inicio del sitio Web (`index.html`, `index.htm`, `index.shtml`) (ver parámetro `DirectoryIndex`).
Tras la instalación de `apache`, al conectarnos al equipo servidor desde un cliente Web se muestra el contenido del directorio `/var/www/html` que es el valor por defecto de esta directiva:

```
DocumentRoot /var/www/html
```

- **DirectoryIndex *Lista***

Informa a `apache` de la lista de las posibles páginas de inicio a servir al recibir una petición sobre un sitio Web que tenga alojado.
Su valor por defecto podemos encontrarlo en el fichero de configuración `/etc/httpd/conf/commonhttpd.conf` en Mandriva 2005 y directamente en el `httpd.conf` en otras distribuciones.

```
DirectoryIndex index.html index.htm index.xml index.cgi
```

Este parámetro puede utilizarse tantas veces como sea necesario, siendo la lista resultante la suma de las listas correspondientes a cada una de sus apariciones.

- **Listen *[IP:]port***

Esta directiva nos permite especificar a `apache` a través de que direcciones IP del equipo servidor y puerto TCP de comunicaciones (p.e. `Listen 192.168.1.1:80`) se escucharán/atenderán las peticiones `http` recibidas. Para comprender la importancia de esta directiva debemos tener en cuenta que un equipo puede tener asignadas múltiples direcciones IP, al menos una dirección IP por cada una de las interfaces/tarjetas de red que tenga configuradas. Según esto, mediante esta directiva, se nos permite controlar a través de que interfaz/es de red se atenderán las solicitudes de conexión al servicio `httpd`.
En el caso de no especificar ninguna dirección IP, daremos a entender que admitimos peticiones `http` recibidas por cualquiera de las tarjetas de red, sin discriminación. Por defecto, este es el comportamiento de apache, reservando para dicha labor el puerto `80`, en el caso de no disponer de `Proxy`, `Listen 80`, o el `8080` en caso de sí disponer, `Listen 8080`.

- *<VirtualHost nombre|IP> </VirtualHost>*

En la práctica, el término *virtual host* hace referencia a mantener más de un servicio Web con el mismo equipo. Con <VirtualHost>, indicamos al servidor `apache` el conjunto de directivas que afectarán exclusivamente al sitio Web referenciado mediante un nombre (host virtual basado en nombre) o mediante una dirección IP (host virtual basado en IP).
Ejemplo:

```
<VirtualHost 10.12.33.4>
DocumentRoot /var/www/html1
ServerName www.cossio.com
ErrorLog logs/cossio_error_log
</VirtualHost>
```

- **<IfDefine *parámetro*> </IfDefine>**

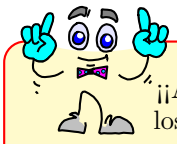
En ocasiones puede interesarnos condicionar la configuración del servidor en función del valor de alguno de los parámetros de `apache`. La directiva `IfDefine` nos permite que la asignación de determinados valores a ciertas directivas de configuración esté condicionada en función de si ha sido o no definido por el administrador el parámetro especificado antes el arranque del servicio `httpd` (opción `-D`, mirar `man httpd`). Su sintaxis es,

```
<IfDefine parámetro>
Parámetros de configuración
</IfDefine>
```

- **<IfModule *módulo*> </IfModule>**

De manera similar a la directiva `IfDefine`, `IfModule` condiciona la asignación de valores a determinadas directivas de configuración en función de si el modulo especificado ha sido incluido/cargado o no. La sintaxis es,

```
<IfModule módulo>
Parámetros de configuración
</IfModule>
```



¡¡Aclaración!! Una directiva de configuración muy habitual que aparecía en los ficheros de configuración `httpd.conf` de `apache` hasta su versión 2 (v1.3) era `ServerType`. A través de esta directiva podíamos decidir la forma en que se ejecutaba `apache` y todos sus procesos asociados, ya que permitía elegir entre dos modos de ejecución `inetd` o `standalone`. A partir de la versión 2 esta directiva ha desaparicido ejecutándose siempre en modo `standalone` (`apache` se ejecuta como un servicio independiente).

Es conveniente tener a mano el manual/tutorial de configuración de `apache2`, que viene con la propia distribución del sistema GNU/Linux Mandrake/Mandriva. Nos podrá servir de consulta ante determinadas dudas que nos surjan a lo largo de la configuración que se va a realizar en la práctica. Para ello, será necesario instalar el paquete `apache2-manual` (Mandriva 2005):

```
[root@linux]# urpmi apache2-manual
```

¡Importante! Señalar que en la última distribución de GNU/Linux Mandriva, `Mandriva 2006`, el paquete encargado de instalar el manual de ayuda de apache ya no es `apache2-manual` sino `apache-doc`.

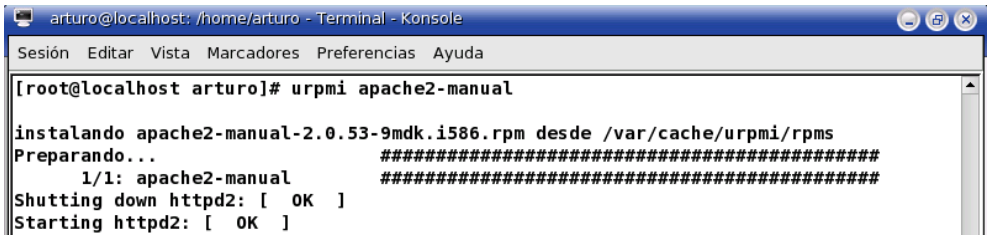


Figura 3.3: Instalación del paquete `apache2-manual`.

Una vez instalado el manual de ayuda del servidor `apache`, podremos pasar a visualizarlo `online` vía Web a través de un cliente Web colocando como URL[10]: `http://`*direccion IP*`/manual-2.0/index.html`. Como es lógico, el servicio `httpd` deberá estar activo para poder visualizarlo, ya que estamos accediendo al manual de ayuda a través del propio `apache`, de hay que tal como se puede observar en la captura de pantalla anterior (figura 3.3), tras instalarse el paquete `apache2-manual` el servicio `http` se reestablece automáticamente.

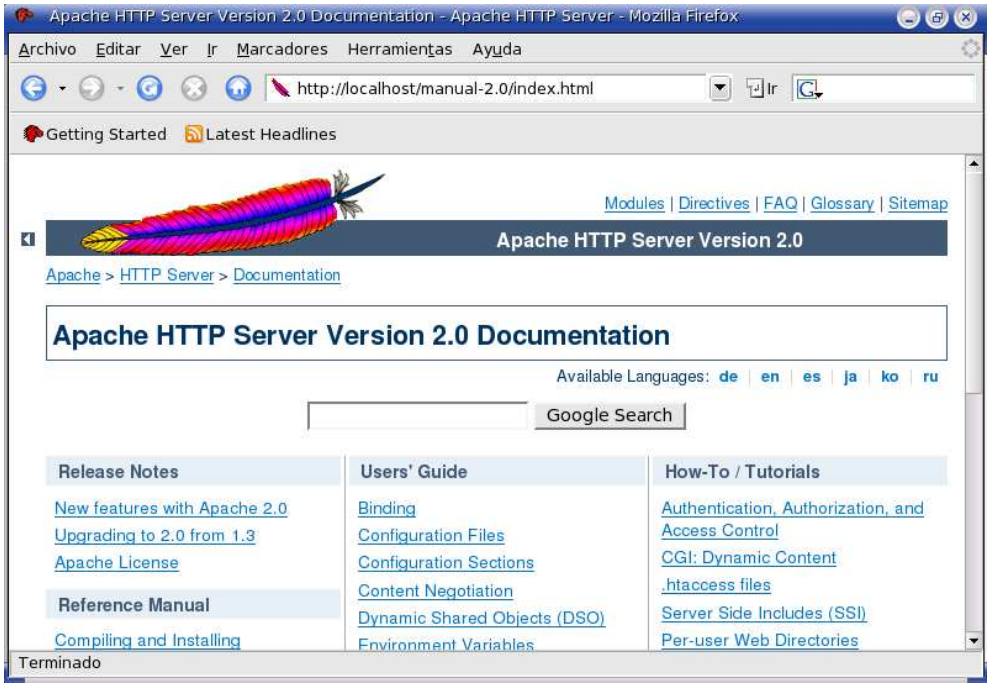


Figura 3.4: Página de inicio del manual de `apache`.

2° **Bloque.- Módulos dinámicos del servidor.** Este segundo bloque del fichero `httpd2.conf` está compuesto por una lista de los distintos módulos que utiliza el administrador del servicio `apache` para llevar a cabo su configuración. Sin estos módulos la configuración de `apache` se vería reducida a un conjunto muy simple de directivas (directivas del núcleo de `apache`), que tan sólo nos permitirían establecer una configuración muy limitada.

[10]En la `Mandriva 2006` la URL sería: `http://`*direccion IP*`/manual/`.

```
#2° BLOQUE
#Módulos/Librerías que proporcionan directivas
                    #de configuración de apache.
LoadModule   access_module       modules/mod_access.so
LoadModule   auth_module         modules/mod_auth.so
LoadModule   auth_anon_module    modules/mod_auth_anon.so
LoadModule   auth_digest_module  modules/mod_auth_digest.so
LoadModule   include_module      modules/mod_include.so
...
```

¡¡Aclaración!! A partir de la versión 2 de `apache` desaparecen las directivas `AddModule` y `ClearModuleList` utilizadas en la versión 1.3 para asegurarse de que los módulos eran cargados adecuadamente.
Además muchos de los módulos utilizados en la v1.3 han sido eliminados, renombrados o fusionados con otros módulos a partir de la v2.

Cada uno de estos módulos no es más que una librería formada por un conjunto de directivas asociadas a una funcionalidad concreta. De esta forma, cuantos más módulos se encuentren cargados (`Loaded`), se dispondrá de un número mayor de directivas de configuración permitiendo una configuración de `apache` mucho más versátil. Entre los distintos módulos podrían destacarse:

Módulos de apache

- **mod_status**

Proporciona un conjunto de directivas que permiten conocer el estado del servicio suministrando una estadística con todas las variables relevantes: porcentaje de uso de CPU, número de solicitudes atendidas, ...

- **mod_dir**

Permite acceder a la página de inicio de un sitio Web (normalmente `index.html`) sin hacer referencia a ella en la URL. Esto es gracias a su directiva `DirectoryIndex`. Colocar

`http://www.misitioweb.es`

sería equivalente a poner

`http://www.misitioweb.es/index.html`

- **mod_access**

Proporciona un conjunto de directivas especializadas en el control de acceso al servidor.

- **mod_auth**

Permite realizar la autentificación de usuarios haciendo uso de ficheros de texto.

- **mod_auth_dbm**

Permite realizar la autentificación de usuarios haciendo uso de bases de datos, `DBM` (DataBase Management).

- **mod_auth_digest**

Permite realizar la autentificación de usuarios en modo seguro haciendo uso de metódos criptográficos (MD5).

- **mod_auth_anon**

Permite la autentificación anónima de usuarios.

- **mod_cgi**

Permite la interpretación y ejecución de `scripts CGI` (el lenguaje más extendido es `Perl`) vía Web desde un cliente en el servidor `apache` con la finalidad aportar dinamismo a las páginas Web servidas por este.

- **mod_suexec**

Permite suplantar a la cuenta de usuario del sistema que especifiquemos en el momento de ejecutar `scripts CGI`.

- **mod_include**

Permite la inclusión de sentencias `SSI` (Server Side Includes) posibilitando la ejecución de comandos del sistema siendo esta una alternativa para dinamizar las Webs.

- **mod_negotiation**

Se encarga de gestionar las negociaciones cliente/servidor.

- **mod_proxy**

Implementa un proxy/gateway para `apache`.

Aunque muchos de estos módulos se encuentran compilados de manera predeterminada, otros deberán ser agregados y compilados expresamente para que sus directivas asociadas puedan ser utilizadas en la configuración de `apache`.

¡¡Atención!! Es posible que los módulos disponibles tras la instalación predeterminada de `apache` no satisfagan las necesidades del administrador. Lo advertirás cuando no dispongas de directivas asociadas a determinadas funcionalidades del servidor Web apache. Es ese caso será necesario agregar el módulo que contenga esas directivas deseadas.

Si deseas agregar un módulo de `apache` lo recomendable sería seguir los siguientes pasos:

1. Consultar si el módulo necesario está disponible en alguno de los repositorios de nuestro sistema:

```
[root@linux]# urpmq nombre modulo
```

2. En caso de que la búsqueda haya resultado satisfactoria, será necesario instalar el correspondiente módulo, y en caso de que no se realice automáticamente, reiniciar el servicio `httpd` (`service httpd restart`) para que los cambios en la configuración surtan efecto.

```
[root@linux]# urpmi nombre modulo
```

3. Si el resultado de la consulta anterior, `urpmq` *`nombre modulo`*, indicara la no existencia del módulo en los repositiorios será necesario agregar[11] alguno nuevo que disponga de los paquetes `RPM` asociados a los módulos requeridos. Otra opción sería acceder, vía Web, a la página `http://nux.se/modules_for_apache2.html`, desde la cual podemos descargarnos los módulos necesarios para agregarlos posteriormente.

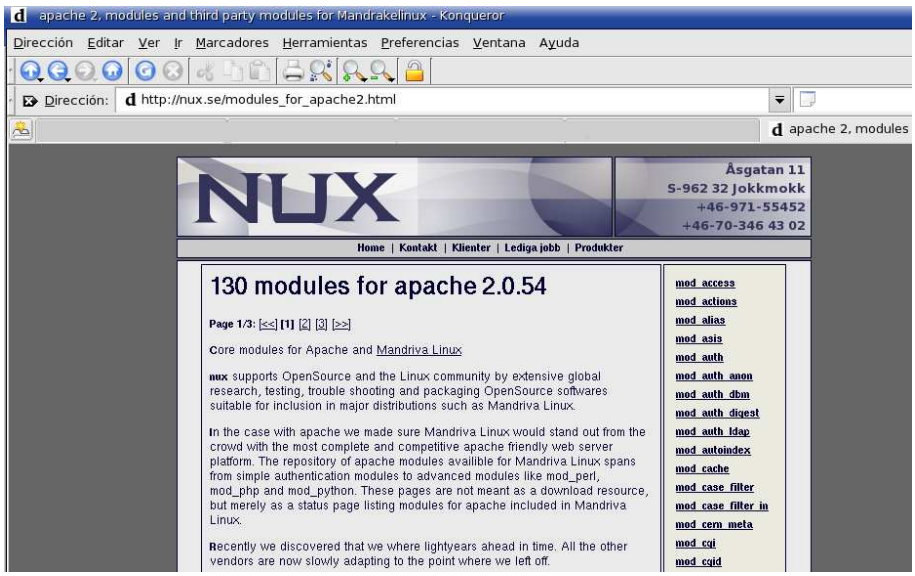


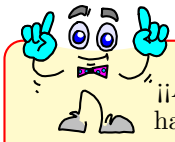
Figura 3.5: Página para la descarga de módulos `apache`.

[11] Agregar repositorios en Mandriva es un procedimiento muy sencillo gracias a la página web `easyurpmi` (busca en `google` y te dará un enlace).

3er **Bloque.- Ficheros externos incluidos en `httpd2.conf`.** Especifica los ficheros externos que serán incluidos (Include) y tenidos en cuenta para la configuración de `apache`, además del propio `httpd/httpd2.conf`.

Entre todos los ficheros que indiquemos cabría destacar `/etc/httpd/2.0/conf/commonhttpd.conf` (Mandriva 2005) ya que en él se localizan las directivas de configuración que serán comunes a todo servicio ofrecido desde `apache`.

```
#3° BLOQUE
#Ficheros de configuración insertados en apache
#Un ejemplo de indicación de ruta absoluta:
Include /etc/httpd/conf.d/*.conf
#Cuando la ruta es relativa, lo es con respecto
# a la indicada en ServerRoot:
Include conf/commonhttpd.conf
Include conf/fileprotector.conf
⋮
```



¡¡Aclaración!! En la última distribución de GNU/Linux Mandriva 2006 se han fusionado los ficheros de configuración de `apache`, `httpd2.conf` y `commonhttpd.conf` en `httpd.conf`, conformando de esta forma, un fichero de configuración mucho más extenso y denso de tratar. Por esta y otras razones, al considerarse una organización más clara la presentada en la versión de `apache` suministrada en la distribución de Mandriva 2005, se hará referencia a ella constantemente.

4° **Bloque.- Interfaces de red y puertos `TCP`.** Informa al servicio `httpd` a través de que direcciones IP y puertos TCP serán atendidas las peticiones Web que se reciban. Para ello se hará uso de la directiva `listen`,

```
Listen [direccionIP:]port
```

vista ya en la lista de directivas generales del `1ER Bloque` (página 84). Este bloque puede ser movido al principio del archivo como si se tratase de una directiva general más.

```
#4° BLOQUE
#Direcciones IP/puertos por los que se escucharán solicitudes
# sobre sitios Web colgados en el servidor apache
<IfDefine APACHEPROXIED>
  Listen 8080
</IfDefine>

<IfDefine !APACHEPROXIED>
  Listen 80
</IfDefine>
```

Como puede observarse, la configuración de este cuarto bloque viene condicionada por el valor de un parámetro de `apache`, `APACHEPROXIED`, el cual informa al servicio `httpd` de si el acceso al servidor se hace a través de un `proxy` o no. En el caso de hacer uso de él, sería necesario indicárselo a `apache` en el arranque del servicio, poniendo a `ON` dicho parámetro.
De igual forma puede advertirse como se ha especificado únicamente el puerto de escucha en la directiva `Listen` *n° puerto* pero no dirección IP. Esto significa que por defecto, si no indicamos lo contrario, en el caso de que el acceso al servidor se realice sin el uso de `proxy`, se atenderá cualquier petición dirigida a cualquier dirección IP que tenga asignada el equipo servidor por el puerto `80`.

Hasta la versión 2 de `apache` (v1.3) existían las directivas de configuración `Port` y `BindAddress`. La primera permitía especificar los puertos por los que escuchaba peticiones el servidor `apache` y la segunda especificaba las direcciones IP asociadas a las interfaces de red a través de las cuales se iban a atender las solicitudes de conexión. La función de ambas ha sido sustituida por `Listen`.

5° **Bloque.- Otras directivas de configuración.** El resto del fichero de configuración `/etc/httpd/conf/httpd2.conf` (GNU/Linux Mandriva 2005) está compuesto, en su mayor parte, por directivas de configuración condicionadas por los módulos disponibles en apache (además de los módulos dinámicos cargados en el bloque nº2, existen módulos compilados explícitamente para apache que pueden ser listados mediante `httpd -l`), las cuales permiten afinar el rendimiento del servidor. Algunas de ellas serán explicadas a lo largo de la presente práctica, omitiendo el resto a su valor por defecto.
Para ello se hace uso de las siguientes directivas correspondientes a los módulos `core` y `mod_proxy`: <`Directory`>, <`DirectoryMatch`>, <`Files`>, <`FilesMatch`>, <`IfDefine`>, <`IfModule`>, <`Location`>, <`LocationMatch`>, <`Proxy`>, <`ProxyMatch`> y <`VirtualHost`>.
El significado de algunas de estas directivas se ha explicado brevemente en la página 84 y siguientes (bloque nº 1) y sobre las mismas se incidirá a medida que vayamos avanzando en materia.

3.6. Personalización de APACHE. Sitio web anónimo

Una vez vista la estructura del fichero de configuración del servidor `apache`, a continuación, mediante la implementación de diferentes supuestos prácticos, aprenderemos a realizar los cambios oportunos en `httpd2.conf` o `httpd.conf`.

Una vez que se hagan cambios dentro del fichero de configuración de `apache` (`httpd.conf` o `httpd2.conf`), conviene testear su contenido para comprobar si existe algún error sintáctico. En caso de que todo sea correcto, reiniciaremos el servicio para que los cambios surtan efecto. Para ello existen tres opciones:

1. Podemos reiniciar directamente el servicio, ya que por defecto, antes de rearrancar a `apache` se comprueba la sintaxis del fichero por si existiesen errores de configuración:

```
[root@linux]# service httpd restart
```

2. Hacer uso de la interfaz que suministra `apache` para su gestión: `apachectl`.

```
[root@linux]# apachectl configtest #equivalente a: apachectl -t
```

   El problema que podemos encontrarnos es que `apachectl` ya no se implemente en la versión de `apache` de la que hagamos uso. Por ejemplo, GNU/Linux Mandriva 2006 no la incluye.

3. Hacer uso directamente de la herramienta suministrada por `apache` para la gestión de su servicio: `httpd`.

```
[root@linux]# httpd -t
```

   Para que el servicio funcione tan sólo es necesario que apareza el mensaje `Syntax OK`, aunque nos muestre algún aviso (Warnings). Recordad que después es necesario reiniciar el servicio.

Ejercicio 3.1

Modifica el fichero `/etc/httpd/conf/httpd2.conf` con la finalidad de que `apache` sirva un sitio Web localizado en `/var/www/html/web1` ante una solicitud emitida desde un cliente Web a través de la siguiente URL: `http://www.miweb1.es`.

Nota: No es objetivo de este libro enseñar a crear páginas Web. Para hacer las pruebas de funcionamiento puedes, por ejemplo, escribir un texto en el `OpenOffice Writer` y guardarlo con formato `html`.
En principio puedes utilizar cualquier cliente Web, pero para hacer las comprobaciones del servicio es muy útil utilizar `lynx`. Este se invoca desde la línea de comandos y sirve páginas en formato texto. Ejemplo:

```
[user@linux]$ lynx http://www.google.es
```

Solución del ejercicio 3.1

Para la realización del ejercicio práctico propuesto será necesario tener en cuenta los siguientes aspectos:

1. Disponer de un sitio Web propio realizado mediante la ayuda de un editor Web (o un procesador de textos, como por ejemplo el OpenOffice Writer). Esta página Web deberá estar almacenada en el equipo servidor y en la ruta indicada `/var/www/html/web1`. La página de inicio del sitio Web se llamará `index.html`, ya que en otro caso deberíamos añadir el nombre a la directiva `DirectoryIndex`.

2. Configurar la resolución de nombres en el equipo cliente para que el nombre de dominio `www.miweb1.es` indicado en la `URL` de tu explorador Web haga referencia al equipo servidor.

   Para ello, suponiendo que el equipo cliente también dispone de un sistema operativo GNU/-Linux, existirán dos posibles opciones (supondremos que la dirección IP del servidor apache es `192.168.1.1`):

   a) Editar el fichero de alias de máquinas `/etc/hosts` y añadir una nueva referencia. Esta solución no es factible, ya que esta modificación sería necesario hacerla en todo equipo cliente que deseara acceder al sitio web.

   b) Configurar nuestro servidor de nombres de dominio (DNS) para que resuelva el nombre anterior. Este se configurará tal como se indicó en el capítulo 2:

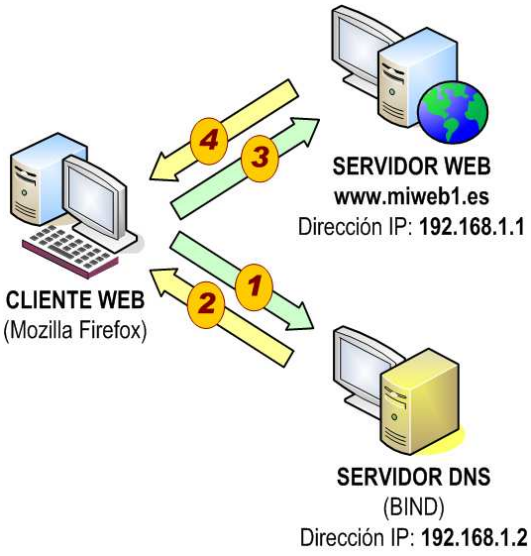
```
#Contenido del fichero de configuración de BIND: /etc/named.conf
options {;
                          directory "/var/named";
};
zone "miweb1.es"in {
type master;
file "maestra.miweb1.es";
};
```

```
miweb1.es.   IN SOA   ns              root.localhost.
                                       (2703200604;
                                       21600;
                                       10800;
                                       604800;
                                       21600; )
             IN NS    ns.miweb1.es.
www          IN A     192.168.1.1
```

```
[root@linux]# service named restart
```

En el caso de escoger esta segunda opción, será necesario indicarle al equipo cliente quien es el equipo DNS con capacidad para resolver el nombre de dominio `www.miweb1.es` (supondremos que la dirección IP del servidor de nombres es `192.168.1.2`) para lo cual será necesario editar el fichero `/etc/resolv.conf`:

```
#Contenido del fichero que informa de
#           los equipos servidores DNS: /etc/resolv.conf
nameserver 192.168.1.2
...
```



Una vez escrito en la barra de direcciones de nuestro cliente Web preferido la `URL` correspondiente al sitio Web a visitar, `http://www.miweb1.es`, tras pulsar `ENTER` se suceden las siguientes acciones:

1. El equipo cliente, para poder dar respuesta a la solicitud http realizada, se pregunta ¿Quién es el equipo que sirve el sitio Web `www.miweb1.es`? Entonces se dirige al servidor `DNS` que tiene configurado para que resuelva el nombre del equipo indicado en la `URL`.
2. El servidor `DNS` nos devuelve la dirección IP del equipo que sirve el sitio Web que deseamos visualizar:
   `www.miweb1.es` $\Longrightarrow$ `192.168.1.1`
3. El equipo cliente solicita vía protocolo `http` al equipo `192.168.1.1` el sitio Web que tiene alojado.
4. El servidor Web nos suministra la página de inicio de `miweb1`.

Antes de pasar al siguiente paso sería conveniente comprobar desde el equipo cliente que la resolución se realiza de la forma esperada mediante la herramienta `ping`:

```
[user@linux]$ping www.miweb1.es
PING www.miweb1.es (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.45 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.43 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.43 ms
```

3. Por último, configuraremos `apache` para que al recibir la petición Web anterior muestre el contenido del sitio Web localizado en `/var/www/html/web1`. Para ello, tan sólo será necesario modificar dos parámetros de configuración dentro del fichero `/etc/httpd/conf/httpd2.conf` (Mandriva 2005), `DocumentRoot` y `ServerName`, pudiendo omitir el resto a su valor por defecto.

```
#Contenido del fichero de configuración del servidor apache:
                                              # /etc/httpd/conf/httpd2.conf
ServerName     www.miweb1.es        #Nombre asociado al sitio Web a servir
DocumentRoot   /var/www/html/web1   #Raíz del sitio Web a servir
...#El resto de parámetros del fichero se pueden dejar a su valor por defecto
```

Para que los cambios realizados en la configuración del servicio `httpd2` surtan efecto deberemos reiniciar el servicio:

```
[root@linux]# service httpd restart (/etc/init.d/httpd restart)
```

La comprobación del correcto funcionamiento del servicio puedes hacerlo llamando a la página web servida por `apache`, tal y como se muestra en la figura anterior.

Ejercicio 3.2

Modifica la configuración anterior para que `apache` tan sólo atienda peticiones Web por el puerto `2006`, y sirva como página de inicio del sitio Web `pagina_inicial.html`.

Solución del ejercicio 3.2

Manteniendo la configuración realizada en el ejercicio práctico anterior, tan sólo será necesario tener en cuenta dos parámetros de `apache`:

1. `Listen 2006`: informaremos al servicio `httpd` de que tan sólo escuche peticiones al sitio Web que sirve a través del puerto `2006`. Será necesario comentar o borrar las líneas del `4º bloque` del fichero de configuración `httpd2.conf` con la finalidad de que no atienda peticiones a través del puerto por defecto `80` (sin `Proxy`) o `8080` (con `Proxy`).

2. `DirectoryIndex pagina_inicial.html`: Advertirá al servidor de que al recibir una petición Web, tenga en cuenta como página de inicio a servir `pagina_inicial.html`. Deberá renombrarse el fichero `index.html` a `pagina_inicial.html`.

```
#Contenido del fichero de configuración del servidor apache:
                                                  #/etc/httpd/conf/httpd2.conf
ServerName       www.miweb1.es              #Nombre asociado al sitio Web a servir
DocumentRoot     /var/www/html/sitioweb1    #Raíz del sitio Web a servir
DirectoryIndex   pagina_inicial.html        #Página de inicio a servir
Listen           2006                       #Puerto de escucha del servicio httpd2
...
#El resto de parámetros se pueden dejar a su valor por defecto,
#excepto el 4º Bloque que debe comentarse, precediendo las líneas
```

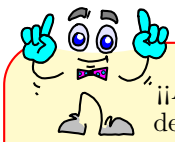
Continúa en página siguiente

```
#que lo componen con un #, o borrarse. En caso contrario escuchará peticiones
# igualmente por el puerto 80 o 8080

#<IfDefine APACHEPROXIED>
  #Listen 8080
#</IfDefine>
#<IfDefine !APACHEPROXIED>
  #Listen 80
#</IfDefine>
```

Para que los cambios realizados en la configuración del servicio `httpd` surtan efecto deberemos reiniciar el servicio:

```
[root@linux]# service httpd restart
```

Para comprobar el correcto funcionamiento de la configuración realizada, desde un cliente Web, por ejemplo Mozilla Firefox, iniciaremos una nueva sesión Web, y colocaremos en la barra de direcciones: `http://www.miweb1.es:2006`.



¡¡Aclaración!! Como habrá podido advertirse, el puerto `80` es el puerto por defecto a través del cual los servidores Web atienden las peticiones dirigidas hacia alguno de los sitios Web que sirven. Es decir, si no se indica lo contrario, cuando se inicia sesión desde un explorador/cliente Web, y se hace uso del protocolo `http` (`http://www....`) para acceder a un servidor, el acceso siempre tiende a realizarse a través del puerto `80`. Esto significa que cuando nos conectamos por ejemplo a `GOOGLE`, al cliente Web le es indiferente que pongamos en la barra de direcciones `http://www.google.es` o `http://www.google.es:80`.
Esto es así gracias a que el puerto `80` pertenece a la lista de puertos `Well-Know` (puertos bien conocidos) comprendidos entre `0:1023`, donde el número de puerto y el servicio ofrecido a través de éste están relacionados de manera única.
Una forma de conocer la relación entre los números de puerto, y el servicio asignado a estos, es a través del fichero de configuración `/etc/services`. Según esto, sólo será necesario especificar el número de puerto en aquellos casos donde éste no tenga asignado ningún servicio, o que en el caso de tenerlo, se haga uso de él para un servicio que no es el que tiene asignado por defecto.

3.7. Servir varios sitios Web: Hosts Virtuales

En el apartado anterior hemos visto como configurar `apache` para que sirva un sitio Web de manera anónima. Esta no suele ser la situación habitual, sino que normalmente un mismo servidor

Web da servicio a multitud de sitios Web (hay menos servidores Web accesibles, que sitios Webs disponibles en la red).

En el presente apartado abordaremos las diferentes formas disponibles a la hora de configurar `apache` con la finalidad de servir diferentes sitios Web: `Host Virtuales basados en direcciones IP` y `Host Virtuales basados en nombre`.

3.7.1. Hosts Virtuales basados en IP

Originalmente, la única manera de visualizar en un cliente Web un determinado sitio era accediendo al servicio ofrecido a través de su dirección IP (generalmente por el puerto 80). Es decir, una vez realizada una petición `http` al servidor, la dirección IP era el parámetro que identificaba de manera inequívoca el sitio Web alojado en el servidor. De esta forma, la única manera de que un equipo pudiese dar servicio a más de un sitio Web, era disponiendo de más de una dirección IP, al menos tantas como sitios Web se deseaban servir.

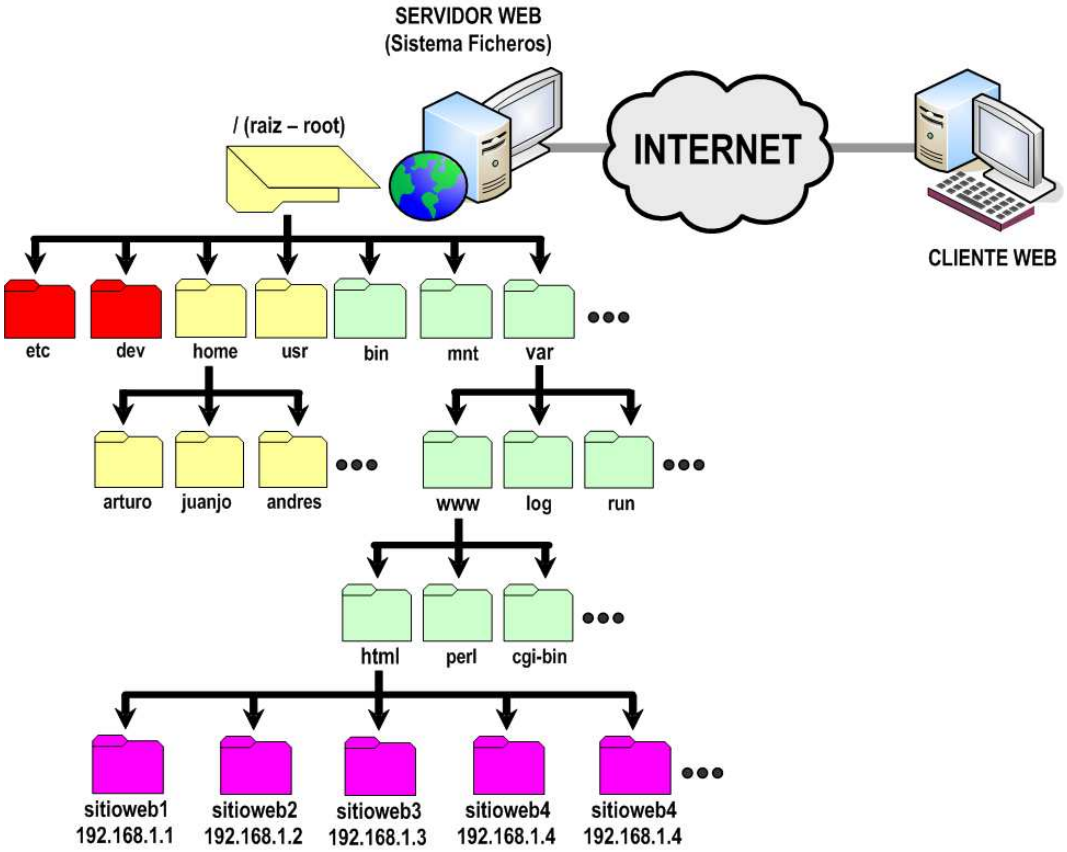


Figura 3.6: Sistema de ficheros de nuestro servidor Web apache en GNU/Linux

Para conseguir la relación anterior (dirección IP $\Longleftrightarrow$ Sitio Web), era necesario asignar a la interfaz de red del equipo servidor varias direcciones IP, y modificar la configuración del servidor `apache` para informar al servicio `httpd` de qué sitio Web debía servir en cada instante en función de la dirección IP recibida con la petición Web. Esto es posible gracias a la directiva `<VirtualHost></VirtualHost>`, al hacer creer al cliente que está accediendo a distintos equipos servidores al colocar diferentes direcciones IP con la finalidad de acceder a diferentes sitios Web, cuando al fin y al cabo, es el mismo servidor quien sirve todos ellos, de ahí que se le denomine `Host Virtual`. Para su utilización haremos uso de la siguiente sintaxis basada en lenguajes de etiquetas (`HTML`, `XML`, etc.), pudiendo repetir la directiva `<VirtualHost>...</VirtualHost>` tantas veces como `Hosts Virtuales` se desee crear:

```
<VirtualHost direccion IP>
  #Conjunto de directivas que afectarán al funcionamiento del VirtualHost
</VirtualHost>
```

Por motivos de organización, toda configuración de `apache` relacionada con la creación y configuración de los `Hosts Virtuales` se recomienda realizarla en el fichero de configuración del servidor `apache /etc/httpd/2.0/conf/vhosts/vhosts.conf`[12], el cual será tenido en cuenta al reiniciar el servicio, al ser incluido este fichero indirectamente en el fichero `httpd2.conf` (Mandriva 2005) por encontrarse en este último la siguiente línea:

```
# 5° Bloque:
# Sección referente a los Hosts Virtuales
Include conf/vhosts/vhosts.conf
# Línea equivalente a:
# Include ServerRoot/conf/vhosts/vhosts.conf
```

Aunque esta es la recomendación, en los ejercicios prácticos que se van a resolver a continuación, por simplicidad, todos los cambios en la configuración de `apache` los llevaremos a cabo directamente sobre `httpd2.conf` (o `httpd.conf`) incluyendo la directiva `<VirtualHost>` como una más dentro del primer bloque, y así evitar manejar más de un fichero de configuración.

Ejercicio 3.3

Configura el equipo servidor GNU/Linux para que pueda servir 3 sitios Web diferentes, cuyas raíces (`DocumentRoot`) se encuentran en `/var/www/html/sitioweb1`, `/var/www/html/sitioweb2` y `/var/www/html/sitioweb3`, a través de las siguientes direcciones IP: `192.168.1.101`, `192.168.1.102` y `192.168.1.103` (Host Virtuales basados en direcciones IP), garantizando que todos ellos tengan como página Web de inicio `inicio.html`.

Solución del ejercicio 3.3

En primer lugar será necesario configurar la interfaz de red del equipo servidor y asignarle las tres direcciones IP anteriores:

```
#Script de configuración de la interfaz de red del servidor:
ifconfig eth0 192.168.1.101
ifconfig eth0:1 192.168.1.102
ifconfig eth0:2 192.168.1.103
```

A continuación editaremos el fichero `httpd2.conf`, y crearemos tres `Host Virtuales`, cada uno de ellos asociado a una dirección IP y a su correspondiente sitio Web a servir:

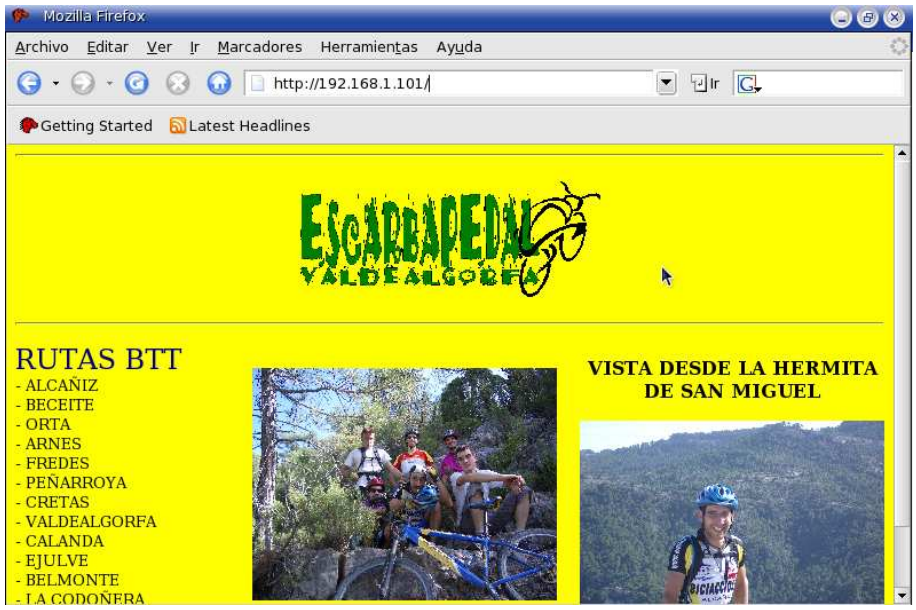
```
#Contenido del fichero de configuración del servidor apache:
#                                        /etc/httpd/conf/httpd2.conf
#1er BLOQUE
ServerRoot   /etc/httpd/2.0
PidFile      /var/run/httpd.pid
ErrorLog     logs/error_log
LogLevel     warn
                                    Continúa en página siguiente
```

[12] Esto es en Mandriva 2005, ya que en Mandriva 2006 el fichero es `/etc/httpd/conf/vhosts.d/vhosts.conf`.

```
DirectoryIndex  inicio.html
<VirtualHost 192.168.1.101>
DocumentRoot    /var/www/html/sitioweb1  #Raíz del sitio Web a servir
</VirtualHost>
<VirtualHost 192.168.1.102>
  DocumentRoot   /var/www/html/sitioweb2  #Raíz del sitio Web a servir
</VirtualHost>
<VirtualHost 192.168.1.103>
  DocumentRoot   /var/www/html/sitioweb3  #Raíz del sitio Web a servir
</VirtualHost>
#2° BLOQUE
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

Advierte que el parámetros `DocumentRoot` desaparece del contexto general para ser trasladado a los `Virtual Host` (lo mismo ocurre con `ServerName`).

Para comprobar la configuración anterior, será necesario especificar en la barra de direcciones del explorador Web el protocolo de comunicación `http` seguido de la dirección IP correspondiente al sitio Web que se desea visualizar (URL: `http://`*`direcciónIP`*`:`*`puerto`*):



Ejercicio 3.4

Modifica la configuración del ejercicio anterior de tal forma que pueda accederse a los tres sitios Web anteriores haciendo uso de los siguientes alias o nombres de dominio:

Dirección IP (Host Virtual)	**Nombre del Sito Web** (Nombre de Dominio)	**Raíz del Sitio Web** (DocumentRoot)
192.168.1.101	www.escarbapedal.es	/var/www/html/sitioweb1
192.168.1.102	www.miweb.es	/var/www/html/sitioweb2
192.168.1.103	www.tuweb.com	/var/www/html/sitioweb3

Además, deberemos garantizar que al primero de los sitios Web pueda accederse a través de los puertos 80 o 2001, al segundo a través de los puertos 80 ó 2002, y al tercero a través del 80 y 2003.

Solución del ejercicio 3.4

Para ofrecer el servicio Web solicitado en el presente ejercicio práctico, será necesario llevar a cabo previamente las siguientes acciones:

1. Registrar los nombres de dominio anteriores en el servidor de nombres de dominio (DNS) (o en `/etc/resolv.conf`). Para ello será necesario configurar en el servidor DNS el servicio `named` (`bind`) y crear tres zonas nuevas (`escarbapedal.es`, `miweb.es` y `tuweb.com`) y sus respectivos archivos de zona, tal como ya se hizo en el ejercicio práctico nº1 (No olvidar reiniciar el servicio `named` para que los cambios surtan efecto). Otra opción, tal como ya se ha indicado anteriormente, sería editar el fichero de alias de máquinas `/etc/hosts` del equipo cliente, para que pueda resolver los nombres anteriores (`www.escarbapedal.es`, `www.miweb.es` y `www.tuweb.com`) sin necesidad de tener que consultar a un DNS.

2. Modificar el fichero `httpd2.conf` en el equipo servidor Web para agregar a cada uno de los `Hosts Virtuales` creados el parámetro `ServerName`, y modificar la directiva `Listen` del cuarto bloque:

```
#Contenido del fichero de configuración del servidor apache:
#                                        /etc/httpd/conf/httpd2.conf
#1er BLOQUE
ServerRoot      /etc/httpd/2.0
PidFile         /var/run/httpd.pid
ErrorLog        logs/error_log
LogLevel        warn
DirectoryIndex  inicio.html

<VirtualHost 192.168.1.101>
  DocumentRoot   /var/www/html/sitioweb1  #Raíz del sitio Web a servir
  ServerName     www.escarbapedal.es      #Nombre asociado al sitio Web a servir
</VirtualHost>

<VirtualHost 192.168.1.102>
  DocumentRoot   /var/www/html/sitioweb2  #Raíz del sitio Web a servir
  ServerName     www.miweb.es             #Nombre asociado al sitio Web a servir
</VirtualHost>

<VirtualHost 192.168.1.103>
  DocumentRoot   /var/www/html/sitioweb3  #Raíz del sitio Web a servir
  ServerName     www.tuweb.com            #Nombre asociado al sitio Web a servir
</VirtualHost>
#2° BLOQUE
...
#4° BLOQUE
Listen          192.168.1.101:80
Listen          192.168.1.101:2001
Listen          192.168.1.102:80
Listen          192.168.1.102:2002
Listen          192.168.1.103:80
Listen          192.168.1.103:2003
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

Por último, reiniciaremos el servicio `httpd` (`apache`) para que la nueva configuración sea

adoptada por el servidor Web, y así podamos probarla desde un cliente Web:

```
[root@linux]# service httpd restart
```

Conviene aclarar que la configuración presentada en este ejercicio tiene los siguientes inconvenientes:

1. No tiene sentido añadir una nueva directiva de configuración, `ServerName`, para identificar a los diferentes sitios Web a servir, cuando quien ya establece la distinción es la dirección IP. Es decir, cuando nosotros colocamos en la barra de direcciones de nuestro cliente Web (p.e. `Mozilla Firefox`) `http://www.escarbapedal.es` en primer lugar nos dirigimos a nuestro servidor DNS para que nos resuelva el nombre anterior a su correspondiente dirección IP (`www.escarbapedal.es` ⇒ `192.168.1.101`). Una vez que ya tenemos la dirección del equipo que sirve dicho sitio Web, nos dirigimos a él haciendo una petición Web. Este identifica cuál de los sitios Web debe descargar a través de la dirección IP de destino, y como sólo existe un único `Host Virtual` identificado inequívocamente a través de esta, nos muestra su página de inicio. No se requiere por tanto de `ServerName`.

2. Según lo anterior, en el caso de que nuestro servidor Web formase parte de la `Internet`, sería necesario adquirir tantas direcciones IP públicas como sitios Web quisiéramos servir. Si tenemos en cuenta el incesante incremento de sitios Web que son creados y colgados de servidores de `Internet`, la cantidad de direcciones IP requeridas para ello sobrepasaría el número disponible.

3. Si por cada dirección IP pública que deseamos adquirir hay que pagar, el coste del servicio se ve incrementado notablemente.

Para dar solución a los inconvenientes antes citados, surgieron nuevas versiones del protocolo HTTP permitiendo la implementación en servidores Web de `Hosts Virtuales basados en nombre`.

3.7.2. Hosts Virtuales basados en nombre

Una alternativa a lo visto en el apartado anterior (`Hosts Virtuales basados en dirección IP`) es la configuración de `Hosts Virtuales basados en nombre`. Esta opción permite servir a `apache` varios sitios Web mediante una única dirección IP distinguiendo estos entre sí a través de un nombre de dominio diferente. Para advertir a `apache` de que se va hacer uso de este tipo de `Hosts Virtuales` se hace uso de una nueva directiva llamada `NameVirtualHost` *`dirección IP`* . Esta directiva va a informar al servidor `apache` de cual de las direcciones IP que puede tener asignadas el equipo servidor va a ser utilizada para dar servicio a varios sitios Web mediante diferentes nombres de dominio.

Es decir, si el fichero de configuración `httpd2.conf` (o el `httpd.conf`) contuviese el conjunto de directivas que se muestra en el siguiente ejemplo con la finalidad de servir dos sitios Web a través de dos `Hosts Virtuales` con la misma dirección IP (`192.168.1.101`), independientemente del parámetro `ServerName`, al recibir una petición con destino `192.168.1.101` serviría el primero de ellos, y omitiría el segundo:

```
#Contenido del fichero de configuración del servidor apache:
#                                    #/etc/httpd/conf/httpd2.conf
#1er BLOQUE
ServerRoot      /etc/httpd/2.0
PidFile         /var/run/httpd.pid
ErrorLog        logs/error_log
LogLevel        warn

<VirtualHost 192.168.1.101>
  DocumentRoot  /var/www/html/sitioweb1  #Raíz del sitio Web a servir
  ServerName    www.escarbapedal.es      #Nombre asociado al sitio Web a servir
</VirtualHost>

<VirtualHost 192.168.1.101>
  DocumentRoot  /var/www/html/sitioweb2  #Raíz del sitio Web a servir
  ServerName    www.miweb.es             #Nombre asociado al sitio Web a servir
</VirtualHost>
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

El ejemplo anterior presentaria una confusión para `apache` ya que se han definido dos sitios Web identificados con una misma dirección IP. Añadiendo la directiva `NameVirtualHost` *`dirección IP`* informamos a `apache` de que se han definido más de un `Host Virtual` sobre la misma dirección IP, y que para saber cual es el sitio a servir debe fijarse en el parámetro `ServerName`. Así pasan a identificarse de manera inequívoca mediante el nombre de dominio asociado.

Por tanto, para que nuestro servidor pudiese servir dos sitios Web a través de la misma dirección IP, el fichero de configuración `httpd2.conf` debería quedar de la siguiente forma:

```
#Contenido del fichero de configuración del servidor apache:
#                                    #/etc/httpd/conf/httpd2.conf
#1er BLOQUE
ServerRoot      /etc/httpd/2.0
PidFile         /var/run/httpd.pid
ErrorLog        logs/error_log
LogLevel        warn
#Advertimos al servidor APACHE de que existe más de un Host Virtual
#con la misma IP, los cuales se identifican a través su ServerName:
NameVirtualHost  192.168.1.101

<VirtualHost 192.168.1.101>
  DocumentRoot  /var/www/html/sitioweb1  #Raíz del sitio Web a servir
  ServerName    www.escarbapedal.es      #Nombre asociado al sitio Web a servir
</VirtualHost>

<VirtualHost 192.168.1.101>
  DocumentRoot  /var/www/html/sitioweb2  #Raíz del sitio Web a servir
  ServerName    www.miweb.es             #Nombre asociado al sitio Web a servir
</VirtualHost>
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

Ejercicio 3.5

Modifica la configuración del ejercicio anterior de tal forma que pueda accederse a los tres sitios Web anteriores mediante una única dirección IP (`192.168.1.101`), a través del puerto `80`, haciendo uso de sus correspondientes nombres de dominio asociados:

Dirección IP (Host Virtual)	**Nombre del Sito Web** (Nombre de Dominio)	**Raíz del Sitio Web** (DocumentRoot)
192.168.1.101	www.escarbapedal.es	/var/www/html/sitioweb1
192.168.1.101	www.miweb.es	/var/www/html/sitioweb2
192.168.1.101	www.tuweb.com	/var/www/html/sitioweb3

Solución del ejercicio 3.5

Al igual que en el ejercicio práctico anterior, se asumirá que está correctamente configurado el servidor de nombres de dominio (DNS), o en su defecto, modificado el fichero de alias de máquinas `/etc/hosts` del equipo cliente para que sean reconocidos los nombres asociados a los sitios Web a servir.

Tal como ya se ha expuesto al principio de este apartado (el 3.7.2), tan sólo sería necesario incluir en la configuración del servidor la directiva `NameVirtualHost`:

```
#Contenido del fichero de configuración del servidor apache:
#                                        #/etc/httpd/conf/httpd2.conf
#1er BLOQUE
ServerRoot        /etc/httpd/2.0
PidFile           /var/run/httpd.pid
ErrorLog          logs/error_log
LogLevel          warn
DirectoryIndex    inicio.html
NameVirtualHost   192.168.1.101
<VirtualHost 192.168.1.101>
  DocumentRoot    /var/www/html/sitioweb1  #Raíz del sitio Web a servir
  ServerName      www.escarbapedal.es      #Nombre asociado al sitio Web a servir
</VirtualHost>

<VirtualHost 192.168.1.101>
  DocumentRoot    /var/www/html/sitioweb2  #Raíz del sitio Web a servir
  ServerName      www.miweb.es             #Nombre asociado al sitio Web a servir
</VirtualHost>

<VirtualHost 192.168.1.101>
  DocumentRoot    /var/www/html/sitioweb3  #Raíz del sitio Web a servir
  ServerName      www.tuweb.com            #Nombre asociado al sitio Web a servir
</VirtualHost>
#2° BLOQUE
...
#4° BLOQUE
Listen            192.168.1.101:80
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

Ejercicio 3.6

Configura el servicio `httpd` para que cumpla los requisitos que se resumen en la siguiente tabla:

DocumentRoot	Dirección IP	ServerName	Puertos TCP	Página de inicio
/var/www/html/web1	192.168.1.201	web1.virtual.es	80, 8088	inicio.html
/var/www/html/web2	192.168.1.201	web2.virtual.es	80, 8088	inicial.html
/var/www/html/web3	192.168.1.202	web3.virtual.es	80, 8888	inicio.html
/var/www/html/web4	192.168.1.203	web4.virtual.es	80, 10010	paginicio.html
/var/www/html/web5	192.168.1.203	web5.virtual.es	80, 10010	indice.html

Solución del ejercicio 3.6

Tal como puede observarse en la tabla de especificaciones anterior, para dar servicio a los cinco sitios Web propuestos, mediante el uso de tres direcciones IP, será necesario implementar `Hosts Virtuales basados en nombre`. En concreto, será necesario crear cinco `Hosts Virtuales` de los cuales, los dos primeros, y los dos últimos, al servirse bajo la misma dirección IP se implementarán mediante `Hosts Virtuales basados en nombre`, y el tercer sitio Web se implementará como `Host Virtual basado en dirección IP` al poderse identificar de manera inequívoca mediante su dirección IP asignada.

Según esto, deberemos hacer uso del parámetro `NameVirtualHost` en dos ocasiones para informar a `apache` de que se implementan `Hosts Virtuales basados en nombre` bajo dos de las direcciones IP que tiene asignadas el equipo servidor:

```
#Contenido del fichero de configuración del servidor apache:
#                                        #/etc/httpd/conf/httpd2.conf
#1er BLOQUE
ServerRoot        /etc/httpd/2.0
PidFile           /var/run/httpd.pid
ErrorLog          logs/error_log
LogLevel          warn
#Implementamos los dos primeros Hosts Virtuales Basados
#                                        #en Nombre bajo la IP 192.168.1.201
NameVirtualHost   192.168.1.201
<VirtualHost 192.168.1.201>
  DocumentRoot    /var/www/html/web1  #Raíz del sitio Web a servir
  ServerName      web1.virtual.es     #Nombre asociado al sitio Web a servir
  DirectoryIndex  inicio.html         #Página Web de inicio del sitio Web
</VirtualHost>

<VirtualHost 192.168.1.201>
  DocumentRoot    /var/www/html/web2  #Raíz del sitio Web a servir
  ServerName      web2.virtual.es     #Nombre asociado al sitio Web a servir
  DirectoryIndex  inicial.html        #Página Web de inicio del sitio Web
</VirtualHost>

#Implementamos un tercer Host Virtual basado en la dirección IP 192.168.1.202
<VirtualHost 192.168.1.202>
  DocumentRoot    /var/www/html/web3  #Raíz del sitio Web a servir
  ServerName      web3.virtual.es     #Nombre asociado al sitio Web a servir
  DirectoryIndex  inicio.html         #Página Web de inicio del sitio Web
</VirtualHost>
#Implementamos los dos últimos Host Virtual basados
#                                        #en Nombre bajo la IP 192.168.1.203
```

Continúa en página siguiente

```
NameVirtualHost   192.168.1.203

<VirtualHost 192.168.1.203>
  DocumentRoot    /var/www/html/web4   #Raíz del sitio Web a servir
  ServerName      web4.virtual.es      #Nombre asociado al sitio Web a servir
  DirectoryIndex  paginicio.html       #Página Web de inicio del sitio Web
</VirtualHost>

<VirtualHost 192.168.1.203>
  DocumentRoot    /var/www/html/web5   #Raíz del sitio Web a servir
  ServerName      web5.virtual.es      #Nombre asociado al sitio Web a servir
  DirectoryIndex  indice.html          #Página Web de inicio del sitio Web
</VirtualHost>
#4° BLOQUE
Listen            192.168.1.201:80
Listen            192.168.1.201:8088
Listen            192.168.1.202:80
Listen            192.168.1.202:8888
Listen            192.168.1.203:80
Listen            192.168.1.203:10010
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

Además, como es lógico, para que la configuración anterior funcione adecuadamente será necesario modificar la configuración del fichero de alias (`/etc/hosts`) del equipo cliente, o modificar el archivo de zona en el servidor DNS asociado al nombre de dominio `virtual.es`:

- Una primera opción es editar el fichero `/etc/hosts` como `root` y escribir,

```
#Contenido del fichero de alias de máquinas del equipo cliente: /etc/hosts
127.0.0.1       localhost
192.168.1.201   web1.virtual.es   web2.virtual.es
192.168.1.202   web3.virtual.es
192.168.1.203   web4.virtual.es   web5.virtual.es
```

- La segunda opción es crear en `/etc/named.conf` la zona `virtual.es`,

```
#Contenido del fichero de configuración de bind del servidor DNS /etc/named.conf
options {
    directory "/var/named";
}                                                                    ;
zone "virtual.es"in {
    type master;
    file "maestra.virtual.es";
};
```

y después creamos el archivo de configuración de zona `/var/named/maestra.virtual.es`,

```
#Fichero de configuración de la zona virtual.es
$TTL 3h
virtual.es.   IN SOA   servidor   root.localhost. (
                                  1504200602
                                  21600
                                  10800
```

Continúa en página siguiente

```
                                        604800
                                        21600
                                        )

virtual.es.   IN NS   servidor
web1          IN A    192.168.1.201
web2          IN A    192.168.1.201
web3          IN A    192.168.1.202
web4          IN A    192.168.1.203
web5          IN A    192.168.1.203
```

3.8. Implementación de sitios Web no anónimos

En ocasiones puede interesarnos restringir el acceso a un sitio Web que ha sido *colgado* en nuestro servidor Web. El acceso a su contenido puede restringirse de manera total o parcial. Esto suele ser habitual cuando el contenido de algunas páginas o información a descargar es confidencial. A continuación, mediante la resolución de varios ejercicios prácticos se mostrarán algunas de las alternativas que nos proporciona el servidor `apache`.

Recordando que la funcionalidad del servidor `apache` depende de la cantidad de módulos que tiene cargados, en primer lugar deberemos conocer qué directivas de configuración están relacionadas con el control de acceso al servicio y a qué módulos pertenecen. De esta forma podemos asegurarnos de que disponemos de ellos o agregarlos en caso de que sea necesario.

Los módulos básicos relacionados con el control de acceso restringido a los sitios Web servidos desde apache son `mod_auth` y `mod_access`, los cuales proporcionan las siguientes directivas que serán utilizadas a continuación:

Directivas de los módulos `mod_auth` y `mod_access`

- **AuthUserFile**

Indica la ruta donde se localiza el fichero que contiene la lista de usuarios y contraseñas, `usuario:contraseña`, de la cual hará uso `apache` para autentificar el acceso. Estos usuarios de `apache` y sus contraseñas deberán haber sido creados previamente mediante el uso del comando `htpasswd`.
Ejemplo:

```
AuthUserFile /var/www/seguridad/usuarios
```

- **AuthGroupFile**

Indica la ruta del fichero que contiene el nombre de los grupos de cuentas de usuario que utilizará `apache` en la autentificación. Esto nos permitirá organizar los usuarios en grupos y controlar su acceso de una manera más comoda.
Ejemplo:

```
AuthUserFile /var/www/seguridad/grupos
```

- **Require *user***

Informa a `apache` de las cuentas de usuario que se encuentran dentro del fichero indicado en `AuthUserFile` que tienen acceso al contenido del directorio especificado con la directiva `Directory`.

En el caso de que deseemos permitir el acceso a todos los usuarios definidos dentro del fichero de usuarios, `AuthUserFile`, no hará falta especificar uno a uno, todos los usuarios que lo componen mediante `Require user`, sino que en su lugar podremos especificar `Require valid-user`.
Ejemplo:

```
Require user arturo
```

.

- **Require *group***

Indica que grupo de usuarios de `apache` tienen acceso al contenido del directorio especificado con la directiva `Directory`. El grupo debe estar contenido en el fichero indicado a través de la directiva `AuthGroupFile`.
Ejemplo:

```
Require group permitidos
```

- **AuthType**

Indica el tipo de autentificación que utilizará `apache`. Aunque el método más común es `Basic`, existen otros más robustos (`Digest`) y seguros pero que presentan el inconveniente de que el cliente Web desde el que se hace la petición al servidor (`Internet Explorer`, `Mozilla Firefox`, `konqueror`, etc.) debe poder soportarlo.
Ejemplo:

```
AuthType Basic
```

- **AuthName**

Indica el mensaje que aparecerá en el cliente Web, al usuario que trata de acceder al contenido del sitio Web restringido, en el momento de la autentificación. El `login:password` introducidos, serán utilizados posteriormente por `apache` para autentificar cualquier otra zona restringida a la que se acceda desde la misma sesión abierta por el cliente Web, permitiendo su acceso en el caso en que dicha zona presente el mismo `AuthName`.
Ejemplo:

```
AuthName Zona Restringida
```

- **Allow|Deny**

Estas directivas permiten especificar desde que equipos se permitirá (`Allow`) o denegará (`Deny`) el acceso al sitio Web servido por `apache`. Como argumentos, se pueden especificar tanto nombres de equipos (alias o nombres de dominio) como direcciones IP (un único equipo o redes de equipos).
Ejemplo 1:

```
Allow from iesrioarba.es
```

Ejemplo 2:

```
Deny from 192.168.1.12
```

- **Order**

Informa a `apache` en qué orden deberán consultarse las directivas de control de acceso a equipos `Allow` y `Deny`.
Ejemplo:

```
Order allow,deny
```

3.8.1. Restricción de acceso a usuarios

Cuando se configura en `apache` un sitio Web de manera no anónima, en primer lugar deberíamos tener presente que las restricciones de acceso se realizan normalmente sobre el contenido de directorios. Por ello, lo primero será conocer que directiva nos permite especificar en `apache` el directorio sobre el que deseamos que no pueda acceder todo el mundo (no anónimo). Esta directiva es `Directory`, la cual encerrará a su vez un conjunto de directivas que afectarán en el acceso a su contenido. Su sintaxis es la que se muestra a continuación:

```
#Contenido del fichero de configuración del servidor apache:
#                                        #/etc/httpd/conf/httpd2.conf
...
<Directory ruta del directorio afectado>
  Directivas de configuración que afectarán al directorio especificado
</Directory>
...
```

Combinando las directivas vistas, y lo explicado en la sección 3.7, es posible configurar `apache` para la implementación tanto de sitios anónimos, como no anónimos. En este apartado empezaremos configurando `apache` de tal forma que tan sólo tengan acceso al contenido de un sitio Web, de manera parcial o total, la lista de usuarios que se especifique. Para ello haremos uso de la directiva `Require` (presentada en el apartado 3.8) , la cual establece que usuarios están permitidos.

¡¡Importante!! Es obligatorio definir previamente las directivas `AuthType`, `AuthName`, `AuthUserFile` y `AuthGroupFile` (esta última si utilizas grupos) para poder hacer uso de la directiva `Require`.

Ejercicio 3.7

Configura `apache` para que el servicio `httpd` sirva los siguientes sitios Web a través de la interfaz de red del equipo servidor con dirección IP `192.168.1.1` puerto `80`, de tal forma que sus páginas de inicio sean en ambos casos `inicial.html`:

DocumentRoot	ServerName	Tipo	Usuarios Permitidos
/var/www/html/web1	web1.ingemartin.com	ANÓNIMO	Todos

/var/www/html/web2	web2.ingemartin.com	NO ANÓNIMO	arturo, juanjo, carol
			AuthType: `Basic`
			AuthUserFile:
			`seguridad/usuarios`
			AuthName:
			`Solo Autorizados`

Solución del ejercicio 3.7

Para cumplir las especificaciones anteriores deberemos tener en cuenta los siguientes puntos:

1. Al servir más de un sitio Web a través de `apache` es necesario hacer uso de `Hosts Virtuales`. Al utilizar la misma dirección IP para ambos sitios, la única forma de identificarlos de manera inequívoca es mediante el nombre de dominio asignado, `ServerName`, por lo que los `Hosts Virtuales` a implementar deberán estar basados en nombre.

```
#Contenido del fichero de configuración del servidor apache:
#                                        #/etc/httpd/conf/httpd2.conf
...
NameVirtualHost  192.168.1.1
<VirtualHost 192.168.1.1>
ServerName       web1.ingemartin.com
...
</VirtualHost>
<VirtualHost 192.168.1.1>
ServerName       web2.ingemartin.com
...
</VirtualHost>
...
```

2. El segundo sitio Web, al ser `no anónimo`, necesitaremos informar a `apache` de que no permita el acceso al directorio raíz (`DocumentRoot`), `/var/www/html/web2`, mediante el uso de las directivas `Directory`, `Require`, ...

```
#Contenido del fichero de configuración del servidor apache:
#                                        #/etc/httpd/conf/httpd2.conf
...
NameVirtualHost  192.168.1.1
...
<VirtualHost 192.168.1.1>
ServerName       web2.ingemartin.com
DocumentRoot     /var/www/html/web2
<Directory /var/www/html/web2>
...
</Directory>
</VirtualHost>
...
```

3. Para poder hacer referencia a los usuarios `arturo`, `juanjo` y `carol`, como usuarios de `apache` permitidos, primero deberemos crearlos. Para ello haremos uso del comando `htpasswd` al cual le tendremos que pasar como argumentos el fichero (`AuthUserFile`), donde será registrada la cuenta de usuario a crear, y el nombre de dicha cuenta de usuario.

```
[root@linux]# htpasswd ruta_del_fichero_AuthUserFile   cuenta_de_usuario_apache
```

Una vez ejecutado el comando `htpasswd`, se nos solicitará una contraseña para el usuario que deseamos crear, y ambos, nombre de usuario y contraseña (encriptada), serán registrados en el fichero indicado como parámetro.

Una vez creadas las cuentas de usuario de `apache` mediante `htpasswd` puede visualizarse el contenido del fichero donde estas son registradas (`more` *nombre_fichero_AuthUserFile*), y advertirse que está compuesto por tantas filas como usuarios creados. Cada una de estas filas está formada por dos columnas, la primera con el nombre del usuario registrado, y la segunda con la contraseña asignada, cifrada por cuestiones de seguridad.

El fichero `AuthUserFile` no es necesario crearlo como paso previo a la creación de las cuentas de usuario de `apache`, ya que si añadimos al comando `htpasswd` la opción `-c`, creará al mismo tiempo el fichero de usuarios y el usuario indicado. Es importante observar que dicha opción, `-c`, sólo será utilizada en la creación de la primera cuenta de usuario, omitiéndolo en la creación del resto de usuarios a registrar por existir ya el fichero.

De esta forma, para cumplir las exigencias del enunciado del ejercicio práctico, deberemos crear las cuentas de usuario `arturo`, `juanjo` y `carol`, y registrarlas en el fichero `seguridad/usuarios` (es una ruta relativa a `ServerRoot`, `/etc/httpd/2.0/seguridad/usuarios`):

```
[root@Linux] htpasswd -c /etc/httpd/2.0/seguridad/usuarios arturo
[root@Linux] htpasswd /etc/httpd/2.0/seguridad/usuarios juanjo
[root@Linux] htpasswd /etc/httpd/2.0/seguridad/usuarios carol
```

4. Al igual que en el resto de ejercicios prácticos, necesitaremos modificar el fichero de alias de máquinas (`/etc/hosts`) del equipo cliente, o la configuración del equipo servidor de nombres de dominio (`/etc/named.conf` y su correspondiente fichero de zona) que tenga definido el cliente en su `/etc/resolv.conf`.

```
//Contenido del fichero de configuración del DNS /etc/named.conf
options {
    directory /var/named";
}                                                              ;
zone "virtual.es" in {
    type master;
    file "maestra.virtual.es";
};
zone "ingemartin.es" in {
    type master;
    file "maestra.ingemartin.es";
};
```

Después creamos el archivo de configuración de zona `/var/named/maestra.ingemartin.es`,

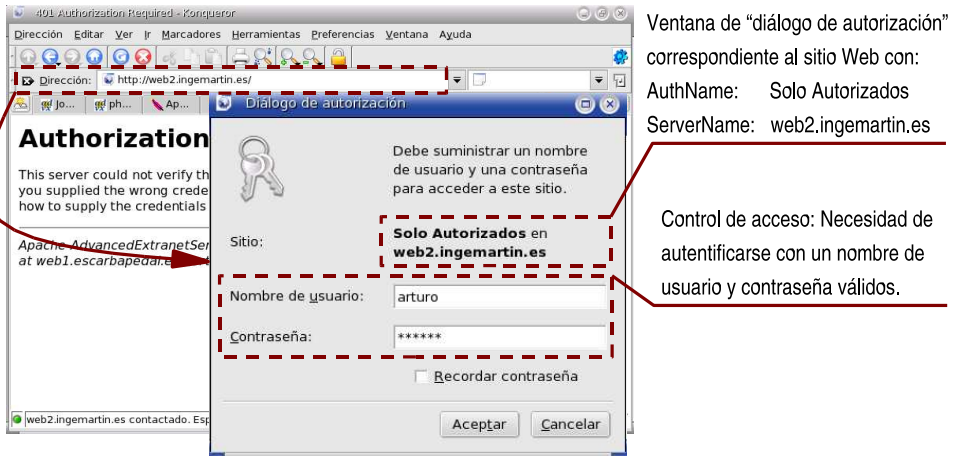
```
;Fichero de configuración de la zona ingemartin.es
$TTL 3h
ingemartin.es.   IN SOA   servidor root.localhost. (
                          1504200602
                          21600
                          10800
                          604800
                          21600)
ingemartin.es.   IN NS    servidor
web1             IN A     192.168.1.1
web2             IN A     192.168.1.1
```

5. Una vez hecho todo lo anterior, acabaríamos de editar el fichero de configuración de `apache`, `httpd2.conf` quedando de la siguiente forma:

```
#Contenido del fichero de configuración del servidor apache:
#                                    /etc/httpd/conf/httpd2.conf
#1er BLOQUE
ServerRoot         /etc/httpd/2.0
PidFile            /var/run/httpd.pid
ErrorLog           logs/error_log
LogLevel           warn
#Indicamos como posible página de inicio a:
DirectoryIndex     inicial.html
#Implementamos dos Hosts Virtuales Basados
#                                    en Nombre bajo la IP 192.168.1.1
NameVirtualHost    192.168.1.1

<VirtualHost 192.168.1.1>
  DocumentRoot     /var/www/html/web1       #Raíz del sitio Web a servir
  ServerName       web1.ingemartin.es       #Nombre asociado al sitio Web a servir
</VirtualHost>

<VirtualHost 192.168.1.1>
  DocumentRoot     /var/www/html/web2       #Raíz del sitio Web a servir
  ServerName       web2.ingemartin.es       #Nombre asociado al sitio Web a servir

  <Directory /var/www/html/web2>
    AuthType          Basic                #Tipo de autentificación
    AuthName          "Sólo Autorizados"   #Mensaje informativo
    AuthUserFile      seguridad/usuarios   # Ruta relativa al fichero
    Require user      arturo juanjo carol  #Lista usuarios permitidos
   </Directory>

</VirtualHost>
...
#4° BLOQUE
Listen             192.168.1.1:80
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

Como puede verse en la siguiente figura, para comprobar el correcto funcionamiento de la configuración anterior, realizaremos peticiones Web al servidor desde un cliente (en este caso `konqueror`, pero puedes emplear el que desees). Al solicitar al servidor el segundo de los sitios Web, `http://web2.ingemartin.es`, nos deberá aparecer una ventana de diálogo informándonos de que estamos intentando acceder en una zona restringida, y de que por tanto, será necesario autentificarnos adecuadamente para poder visualizar su contenido. Es importante comprobar que,

en el caso en que los datos nombre de usuario y contraseña no coincidan con alguno de los existentes en el fichero `AuthUserFile`, el acceso será denegado. En ese caso aparecerá una página de error que el servidor `apache` tiene por defecto.



Otra forma más directa de acceder con una determinada cuenta de usuario a un sitio Web no anónimo es colocando en la barra de direcciones, precediendo al nombre de la página, el nombre del usuario que trata de acceder seguido del carácter @. Como por ejemplo:

```
http://arturo@web2.ingemartin.es
```

El formato de la ventana de diálogo de autorización es dependiente del cliente Web que estamos utilizando. Es el cliente el encargado de advertir al usuario de que realiza una petición sobre un sitio Web no anónimo solicitándole un `login` y una `password` (ver la figura 3.7 en la que se muestran otros cuadros de diálogo posibles).

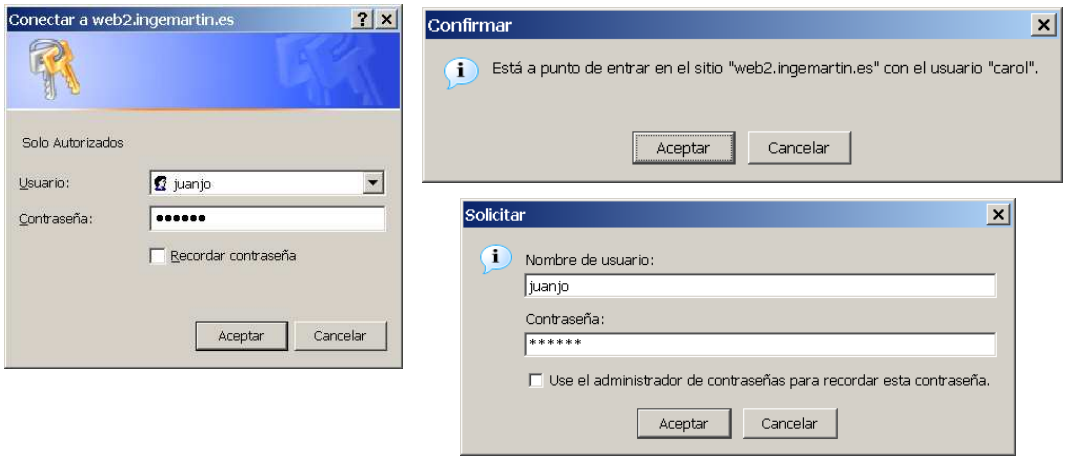


Figura 3.7: Dos posibles formatos de las ventanas de diálogo en función de los clientes `Internet Explorer` y `Mozilla Firefox`.

3.8.2. Restricción de acceso básico a grupos de usuarios

En el apartado 3.8.1 se mostró la forma de controlar el acceso a determinados usuarios. La gestión y mantenimiento de los accesos puede convertirse en algo realmente tedioso si la lista de usuarios es excesivamente amplia o dispar. Una manera más cómoda de gestionar los accesos restringindos consiste en el establecimiento de grupos de usuarios para, posteriormente, hacer referencia a ellos mediante las directivas `AuthGroupFile` y `Require group` vistas en el apartado 3.8.

La primera de ellas nos permitirá definir los grupos de usuarios y los usuarios que los forman, y la segunda directiva establece cual de los grupos de usuarios definidos tendrá permisos de acceso. La forma de utilizarlas se mostrará en el siguiente ejercicio práctico.

Ejercicio 3.8

Configura `apache` para que el servicio `httpd` sirva los siguientes sitios Web a través de la interfaz de red del equipo servidor con dirección IP `12.123.51.89` puerto `80`, con página de inicio `inicial.html`:

DocumentRoot	ServerName	Tipo	Grupos Permitidos
/var/www/html/web1	www.ingemartin.es	ANÓNIMO	Todos
/var/www/html/web2	www.miempresa.es	NO ANÓNIMO	directores: `director1`, `director2` secretarios: `secre1`, ..., `secre3` empleados: `emp1`, ..., `emp8` externos: `usu1`, ..., `usu4` AuthType: `Basic` AuthName: `Solo Autorizados`
/var/www/html/web3	www.bttfriends.es	NO ANÓNIMO	aragon: `julia`, `arturo`, `alonso` larioja: `eva`, `juanjo`, `carol`, `luis` navarra: `andres`, `celia` club1: `ana`, `sara` AuthType: `Basic` AuthName: `Solo Amigos del BTT`

Las cuentas de usuario y grupos de usuarios de `apache` serán creados y registrados en *ServerRoot*`/seguridad/usuapache` y *ServerRoot*`/seguridad/gruposapache` respectivamente.

Solución del ejercicio 3.8

Al igual que en ejercicios anteriores, a continuación se detallarán los pasos que se deberían

seguir de manera razonada para cumplir las especificaciones espuestas:

1. Al tratarse de tres sitios Web, a servir desde `apache`, haciendo uso de la misma dirección IP, nos vemos obligados a configurar tres `Hosts Virtuales` basados en nombre, cada uno de los cuales se encargará de identificar inequívocamente a cada uno de los sitios a servir, identificándolos mediante la directiva `ServerName`.

   Deberán crearse todas la cuentas de usuarios de `apache` especificadas en la tabla anterior mediante el uso del comando `htpasswd` dentro del fichero `/etc/httpd/2.0/seguridad/usuapache` si usas la Mandriva 2005 y del fichero `/etc/httpd/seguridad/usuapache` si usas otra distribución:

```
[root@linux]# htpasswd -c /etc/httpd/2.0/seguridad/usuapache director1
[root@linux]# htpasswd /etc/httpd/2.0/seguridad/usuapache director2
[root@linux]# htpasswd /etc/httpd/2.0/seguridad/usuapache secre1
[root@linux]# htpasswd /etc/httpd/2.0/seguridad/usuapache secre2
...
[root@linux]# htpasswd /etc/httpd/2.0/seguridad/usuapache ana
[root@linux]# htpasswd /etc/httpd/2.0/seguridad/usuapache sara
```

Nota: En adelante las referencias sobre las rutas con respecto al `ServerName` se indicarán suponiendo que se está usando Mandriva 2005. Haz las modificaciones oportunas si utilizas otra distribución.

2. A continuación deberemos crear los grupos de usuarios de `apache` indicados en la tabla de especificaciones del ejercicio práctico, registrando estos en `/etc/httpd/2.0/seguridad/gruposapache` (`AuthGroupFile`). La forma de establecer este registro es tan sencillo como editar con nuestro editor de textos preferido el fichero `AuthGroupFile gruposapache` siguiendo la siguiente sintaxis: *Nombre Grupo Usuarios*: *lista de usuarios separados por un espacio*, tal como muestra el siguiente ejemplo:

```
[root@linux]# vi /etc/httpd/2.0/seguridad/gruposapache
```

```
#Nombre Grupo Usuarios: lista de usuarios separados por un espacio
directores: director1 director2
...#Cada línea se corresponde con un nuevo grupo de usuarios
```

De esta forma, el contenido del fichero `/etc/httpd/2.0/seguridad/gruposapache` quedaría de la siguiente forma:

```
#Contenido del fichero de grupos de usuarios de apache:
#                                    /etc/httpd/2.0/seguridad/gruposapache
directores: director1 director2
secretarios: secre1 secre2 secre3
empleados: emp1 emp2 emp3 emp4 emp5 emp6 emp7 emp8
externos: usu1 usu2 usu3 usu4
aragon: julia arturo alonso
rioja: eva juanjo carol luis
navarra: andres celia
club1: ana sara
```

3. A continuación editaremos `httpd2.conf` (o en su caso `httpd.conf`) para configurar `apache`:

```
#Contenido del fichero de configuración del servidor apache:
#                                        #/etc/httpd/conf/httpd2.conf
#1er BLOQUE
ServerRoot          /etc/httpd/2.0
PidFile             /var/run/httpd.pid
ErrorLog            logs/error_log
LogLevel            warn
#Indicamos como posible página de inicio a:
DirectoryIndex      inicial.html
#Dos Hosts Virtuales Basados en Nombre bajo la IP 12.123.51.89:
NameVirtualHost     12.123.51.89
<VirtualHost 12.123.51.89>                  #Sitio anónimo
  DocumentRoot      /var/www/html/web1      #Raíz del sitio Web a servir
  ServerName        www.ingemartin.es       #Nombre asociado al sitio Web
</VirtualHost>

<VirtualHost 12.123.51.89>
  DocumentRoot      /var/www/html/web2      #Raíz del sitio Web a servir
  ServerName        www.miempresa.es        #Nombre asociado al sitio Web
  <Directory /var/www/html/web2>
     AuthType       Basic                   #Tipo de autentificación
     AuthName       "Sólo Autorizados"      #Mensaje informativo
     AuthUserFile   seguridad/usuapache     #Ruta relativa al fichero
     AuthGroupFile  seguridad/gruposapache  #fichero de grupos
    #Lista de grupos de usuarios permitidos
    Require group directores
    Require group secretarios
    Require group empleados
    Require group externos
   </Directory>
</VirtualHost>

<VirtualHost 12.123.51.89>
  DocumentRoot      /var/www/html/web3      #Raíz del sitio Web a servir
  ServerName        www.bttfriends.es       #Nombre asociado al sitio Web
  <Directory /var/www/html/web3>
    AuthType        Basic                   #Tipo de autentificación
    AuthName        "Sólo Amigos del BTT"   #Mensaje informativo
    AuthUserFile    seguridad/usuapache     #Ruta relativa al fichero
    AuthGroupFile   seguridad/gruposapache  #fichero de grupos
    #Lista de grupos de usuarios permitidos
    Require group aragon larioja navarra club1
   </Directory>
</VirtualHost>
...
#4° BLOQUE
Listen              12.123.51.89:80
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

Observa, en el fichero `httpd2.conf` que se proporciona, dos ejemplos de uso de `Require` para indicar los usuarios y/o grupos de usuarios con permiso de acceso a los contenidos del directorio especificado con `Directory`. Uno de ellos, define cada uno de los grupos en varias líneas y el otro lo hace en una sola.

De igual forma, si se necesitase informar a `apache` de la existencia de más de un fichero de usuarios o grupos, `AuthUserFile` o `AuthGroupFile`, tan sólo sería necesario replicar tales directivas las veces que fuese necesario:

```
<Directory ruta del directorio con acceso restringido>
...
AuthUserFile    seguridad/usuapache1
AuthUserFile    seguridad/usuapache2
...
AuthGroupFile   seguridad/gruposapache1
AuthGroupFile   seguridad/gruposapache1
...
</Directory>
```

4. Al igual que en el resto de ejercicios prácticos, deberemos modificar el fichero `/etc/hosts` del equipo cliente o añadir las correspondientes zonas en el fichero `/etc/named.conf` para crear los correspondientes archivos de zona en el equipo encargado de traducir los nombres de dominio a direcciones IP para el equipo cliente (servidor de nombres de dominio). En el caso de optar por modificar el fichero `/etc/hosts`:

```
#Contenido del fichero de alias de máquinas del equipo cliente: /etc/hosts
127.0.0.1     localhost          mimaquina
...
12.123.51.89  www.ingemartin.es  www.miempresa.es  www.bttfriends.es
```

5. Por último, para que todos los cambios realizados surtan efecto deberemos reiniciar los servicios alterados, `httpd` en el equipo servidor Web, y `named` en el equipo DNS (en el caso en que se haya optado por esta opción):

```
[root@linux]# service httpd restart
[root@linux]# service named restart
```

Ahora ya podría comprobarse el correcto funcionamiento de la configuración realizada mediante el uso de nuestro cliente Web preferido.

3.8.3. Restricción de acceso básico a máquinas

Si en los apartados anteriores hemos aprendido a restringir el acceso a usuarios, a continuación veremos de que forma hacer lo mismo pero con máquinas (`Hosts`). Esto puede resultar interesante en aquellas situaciones donde se desee un acceso controlado al servidor Web para un conjunto concreto de equipos, y no se disponga de ningún equipo específico de control como puede ser un `firewall`. Estos equipos pueden identificarse a través de su dirección IP, su alias (`/etc/hosts`), o su nombre de dominio.

Para hacer esto, se debe hacer uso de las directivas presentadas en el apartado 3.8 (página 110) `Allow`, `Deny` y `Order`. Algunos ejemplos son:

Allow

Especifica la máquina o máquinas a las que se les permite el acceso al contenido del sitio Web. Para especificar una máquina a través de su dirección IP:

```
allow from 192.168.1.1
```

Para especificar varias máquinas:

```
Allow from 192.168.1.1 192.168.0.2 192.168.5.12
```

Para especificar una red:

```
Allow from 192.168.1
```

En caso de utilizar nombre de dominio:

```
Allow from maquina1.iesbajoaragon.es
```

Para permitir el acceso a todas las máquinas de un dominio:

```
Allow from iesrioarba.es cossio.net
```

Deny

Especifica la máquina o máquinas a las que se les deniega el acceso al contenido del sitio Web, utilizando la misma sintaxis que con `Allow`:

```
Deny from 192.168.100 cibernautas.com
```

Order

Permite combinar las directivas `Deny` y `Allow` de manera adecuada para que no se interfieran. Por ejemplo, si únicamente quisiéramos permitir el acceso a las máquinas dentro del dominio `iesrioarba.es` y denegar el acceso al resto, pondríamos lo siguiente:

```
Order allow,deny #especificamos el orden
Allow from iesrioarba.es
Deny from all
```

Ejercicio 3.9

Modifica la configuración del ejercicio práctico anterior para que únicamente pueda accederse al contenido del segundo sitio Web (`www.miempresa.es`) desde los equipos de la propia empresa (dominio `miempresa.es`) y desde la casa del director1 (34.1.52.227) y director2 (68.71.3.64).

Solución del ejercicio 3.9

Tal como se puede advertir, la nueva restricción de acceso a incluir, tan sólo debe afectar al segundo de los `Hosts Virtuales` creados, dejando el resto del contenido del fichero de configuración `httpd2.conf` sin tocar:

```
#Contenido del fichero de configuración del servidor apache:
#                                        #/etc/httpd/conf/httpd2.conf
...
#2° Sitio Web no anónimo
<VirtualHost 12.123.51.89>
  DocumentRoot   /var/www/html/web2  #Raíz del sitio Web a servir
  ServerName     www.miempresa.es    #Nombre asociado al sitio Web
  <Directory /var/www/html/web2>
    AuthType     Basic               #Tipo de autentificación
    AuthName     "Sólo Autorizados"  #Mensaje informativo
```

Continúa en página siguiente

```
    AuthUserFile   seguridad/usuapache     #Ruta relativa al fichero
    AuthGroupFile  seguridad/gruposapache  #fichero de grupos
    #Lista de grupos de usuarios permitidos
    Require group directores
    Require group secretarios
    Require group empleados
    Require group externos
    Order allow deny
    Allow from miempresa.es 34.1.52.227 68.71.3.64
    Deny from all
   </Directory>
</VirtualHost>
#Siguiente Sitio Web no anónimo:
...
```

3.9. Sitios Webs no anónimos: Gestión de usuarios mediante un DBM

Cuando se desea implementar un sitio Web no anónimo se requiere gestionar una lista de usuarios. En caso de que dicha lista sea excesivamente grande, pueden aparecer elevados retardos en la respuesta y un excesivo consumo de recursos debido a que el servicio `httpd` debe rastrearla cada vez que necesita autentificar un usuario que trata de acceder a una zona restringida. Para solucionar este tipo de problemas surgieron los sistemas gestores de bases de datos (`DBM`, `Management DataBase`). Sustituyendo el fichero plano `AuthUserFile` del que hacíamos uso en los ejemplos mostrados en los apartados anteriores, por una base de datos gestionada por un `DBM`, conseguiremos que los retardos sean menores y que el sistema global sea más eficiente.

En primer lugar deberemos proporcionar a `apache` la potencia para manejar bases de datos, para lo cual tendremos que agregar el módulo, `mod_auth_dbm.so`. Normalmente este módulo `DSO`[13] esta disponible tras la instalación de la versión 2 de `apache`[14], pero no se encuentra cargado, por lo que será necesario descomentar la línea correspondiente a los módulos `DSO`, contenida en el 2º bloque de directivas del fichero `httpd2.conf`.

```
#Contenido del fichero de configuración del servidor apache:
#                                    #/etc/httpd/conf/httpd2.conf
...
#2° BLOQUE (Módulos DSO)
LoadModule   auth_dbm_module   modules/mod_auth_dbm.so
...
```

En su defecto será necesario descargarse el correspondiente módulo, y añadirlo manualmente.

Una vez cargado el módulo de autentificación mediante el uso de `DBM` (`mod_auth_dbm.so`), ya podremos hacer uso de las directivas: `AuthDBMType`, `AuthDBMUserFile` y `AuthDBMGroupFile`, cuya utilidad se mostrará a continuación mediante ejercicios prácticos.

[13] `DSO` significa `Dynamic Shared Object` o en español `Objeto Dinámico Compartido`

[14] Puedes comprobarlo listando el directorio `modules`:

```
[root@linux]# ls /etc/httpd/2.0/modules
```

Directivas asociadas al módulo `mod_auth_dbm`

- **AuthDBMUserFile**

Indica la ruta donde se localiza el fichero que contiene la lista de usuarios y contraseñas que utilizará `apache` para validar la autentificación del acceso. Estos usuarios de `apache` y sus contraseñas deberán haber sido creados previamente mediante el uso del comando `htdbm` (explicación en página 122 y siguientes).
Ejemplo:

```
AuthDBMUserFile seguridad/usuariosdbm
```

- **AuthDBMGroupFile**

Indica la ruta del fichero que contiene el nombre de los grupos de cuentas de usuario que utilizará `apache` en la autentificación. Esto nos permitirá organizar los usuarios en grupos y controlar su acceso de una manera más cómoda.
Ejemplo:

```
AuthDBMUserFile seguridad/gruposdbm
```

- **AuthDBMType**

Informa del tipo de DBM del que se va a hacer uso para el registro de los usuarios y contraseñas. Los tipos disponibles son `DB`, `SDBM`, `GDBM` y `NDBM`.
Ejemplo:

```
AuthDBMType DB
```

Ejercicio 3.10

Configura `apache` para que el servicio `httpd` sirva los siguientes sitios Web a través de la interfaz de red del equipo servidor con dirección IP `192.168.101.1`, puertos `80` y `1088`, de tal forma que la página de inicio en ambos casos sea `pag_inicio.html`:

DocumentRoot	ServerName	Tipo	Usuarios Permitidos
/var/www/html/web1	web1.ingemartin.es	ANÓNIMO	Todos
/var/www/html/web2	web2.ingemartin.es	NO ANÓNIMO	inge1, inge2, inge3, inge4, inge5 AuthType: Basic AuthDBMType: DB AuthName: `Solo Ingenieros Martin` AuthDBMUserFile: seguridad/usudbm

Solución del ejercicio 3.10

Los pasos a seguir para resolver el ejercicio son:

1. Se requiere implementar en `apache` dos `Host Virtuales basados en nombre` para dar servicio a los dos sitios Web anteriores a través de una misma dirección IP (`192.168.101.1`).

   El segundo de ellos tiene acceso restringido, por lo que en primer lugar, crearemos la base de datos (`DB`) que contendrá la lista de usuarios con permiso de acceso al sitio Web no anónimo.

   Si para crear los ficheros de usuarios y contraseñas antes utilizábamos `htpasswd`, ahora para la creación tanto de la base de datos como de los usuarios, haremos uso del comando `htdbm`.

¡¡Advertencia!! En la versión 1.3 de `apache` el comando utilizado para la creación tanto de las bases de datos de usuarios, como de los propios usuarios, era `dbmmanage`. A partir de la versión 2, el comando utilizado para llevar a cabo ambas funciones es `htdbm`.

Utilización del comando `htdbm`

- **Opción -c**

Parámetro utilizado para crear la base de datos en el momento de registrar el primero de los usuarios.

- **Opción -T**

Parámetro encargado de informar del tipo de gestor de bases de datos a utilizar. Los tipos de gestores fueron comentados con `AuthDBMType` (página 121) La sintaxis de utilización es:

```
htdbm -cTTipo_Base_Datos   nombre_base_datos   nombre_usuario
```

Ejemplo:

```
htdbm -cTDB bdusuarios amartin
```

También podría haber sido escrito de la siguiente forma:

```
htdbm -c -TDB bdusuarios amartin
```

- **Opción -l**

Parámetro que nos permite listar el total de los usuarios que han sido registrados en la base de datos. Ejemplos de su uso se muestran con al opción `-x` (a continuación).

- **Opción -x**

Parámetro utilizado para eliminar un usuario de la base de datos. Las posibles sintaxis son:

```
htdbm -xTTipo_Base_Datos   nombre_base_datos   nombre_usuario
htdbm -x -TTipo_Base_Datos   nombre_base_datos   nombre_usuario
htdbm -l -lTTipo_Base_Datos   nombre_base_datos
```

Ejemplos:

```
htdbm -l -TSDBM bdusuarios
htdbm -xTSDBM bdusuarios amartin
```

- **Opción -v**

Parámetro utilizado para verificar la contraseña de un usuario.

- **Opción -p**

Evita que se cifre la contraseña, quedando registrada en texto plano.

- **Opción -s**

Garantiza que se cifre la contraseña haciendo uso del algoritmo SHA.

- **Opción -m**

Es la opción por defecto, y garantiza que la contraseña se registre cifrada haciendo uso del algoritmo MD5.

Según la información mostrada en la tabla anterior, la lista de comandos (podrían agruparse formando un `ShellScript`) que sería necesaria ejecutar desde la consola para cumplir las especificaciones que se detallan en el enunciado del ejercicio práctico sería la siguiente:

```
[root@linux] htdbm -c -TDB /etc/httpd/2.0/seguridad/usudbm inge1
[root@linux] htdbm -TDB /etc/httpd/2.0/seguridad/usudbm inge2
[root@linux] htdbm -TDB /etc/httpd/2.0/seguridad/usudbm inge3
[root@linux] htdbm -TDB /etc/httpd/2.0/seguridad/usudbm inge4
[root@linux] htdbm -TDB /etc/httpd/2.0/seguridad/usudbm inge5
[root@linux] htdbm -l -TDB /etc/httpd/2.0/seguridad/usudbm
```

2. Una vez creada la base de datos con todos los usuarios pasaremos a editar el fichero de configuración del servicio `httpd` `httpd2.conf`:

```
#Contenido del fichero de configuración del servidor apache:
#                                    #/etc/httpd/conf/httpd2.conf
#1ER BLOQUE
ServerRoot /etc/httpd/2.0
PidFile /var/run/httpd.pid
ErrorLog logs/error_log
LogLevel warn
#Especificamos como posible página de inicio para los sitios Web: inicial.html
DirectoryIndex pag_inicio.html
#Implementamos dos Hosts Virtuales Basados en Nombre bajo la IP 192.168.1.1
NameVirtualHost 192.168.101.1
                #              SITIO WEB ANÓNIMO              #
<VirtualHost 192.168.101.1>
ServerName web1.ingemartin.es
DocumentRoot /var/www/html/web1
</VirtualHost>
                #              SITIO WEB NO ANÓNIMO              #
<VirtualHost 192.168.101.1>
ServerName web2.ingemartin.es
DocumentRoot /var/www/html/web2
  <Directory /var/www/html/web2>
    AuthType Basic #Tipo de Autentificación
    AuthDBMType DB #Tipo de gestor de bases de datos
    AuthName Solo Ingenieros Martin #Mensaje informativo al autentificarse
    AuthDBMUserFile seguridad/usudbm #Ruta relativa del fichero de usuarios
    Require valid-user #Permitimos el acceso a todo usuario registrado en el DBM
  </Directory>
</VirtualHost>
...
#4° BLOQUE
Listen 192.168.101.1:80
Listen 192.168.101.1:1088
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

3. Por último, reiniciaremos el servicio `httpd` para poder comprobar a continuación, mediante el uso de nuestro cliente Web preferido, que el servicio responde de la manera esperada:

```
[root@linux]# service httpd restart # o /etc/init.d/httpd restart o apachectl restart
```

3.10. Soporte para dar servicio a Webs dinámicas

En el mundo Web se distinguen dos tipos de sitios Web en relación al contenido que albergan, `estáticos` y `dinámicos`. Los primeros se caracterizan porque el contenido de las páginas Web que lo forman es invariante independientemente del usuario que accede, de las acciones que lleva a cabo, del momento en que se produce el acceso (hora, día, ...), del lugar desde el que se accede, ... Además no ofrecen la oportunidad de que el usuario pueda interactuar con el sistema; los segundos se caracterizan por lo contrario.

Los `sitios Web estáticos` son los más sencillos de servir para `apache`, ya que este se limita únicamente a atender la petición `http` que le llega. En función de la dirección de destino (`Hosts Virtuales basados en IP`) o de la `URL` colocada en la barra de direcciones del cliente Web (`Hosts Virtuales basados en nombre`) identifica cuál de los sitios Web que tiene alojados debe servir. Sin hacer ningún tipo de análisis de contenido, descarga al cliente la página Web solicitada. En

estos casos, la única manera de actualizar el contenido del sitio Web es modificando las páginas Web deseadas, y volviéndolas a *subir* al servidor Web renovando de esta forma los ficheros afectados.

En ocasiones puede ser interesante que parte de la información que contiene el sitio Web a servir dependa del usuario que accede, del tipo de acciones que lleva a cabo, del momento en que se accede, del lugar desde el que accede, o simplemente queremos que el usuario pueda interactuar con el sistema (por ejemplo, realizar modificaciones en el servidor), para lo cual se dice que es necesario dinamizar su contenido.

Los `sitios Web dinámicos` son cambiantes dependiendo de una serie de factores, lo cual se acaba traduciendo en una mayor viveza. Algo que a priori puede parecer tan sencillo como introducir el día y la hora en una de las páginas que compone nuestro sitio Web, requiere de un pequeño programa dependiente de `apache` que interactué con el sistema operativo del servidor, para que este le informe de ello y pueda ser mostrado como un contenido más. Por ejemplo, si quisiéramos introducir una información actualizada del tiempo que va a hacer ese día en una determina región geográfica, necesitaríamos de otro programa dependiente de `apache` que se pusiera en comunicación con la estación meteorológica correspondiente para que esta le proporcionase la información necesaria; posteriormente esta información sería procesada y mostrada, por medio de `apache`, a través de la página Web correspondiente.

Acciones similares serían necesarias para la implementación de `tiendas virtuales`, `foros de información`, `consultas a bases de datos`, `sistemas de gestión remota vía Web`, ... A continuación se mostrarán algunas de las técnicas que existen para dinamizar el contenido de nuestros sitios Web, además de mostrar algunos ejemplos que pueden resultar muy interesantes.

3.10.1. Soporte para Webs dinámicas: comandos SSI

Una de las primeras alternativas que surgieron para dinamizar las páginas Web fueron los comandos `SSI`[15]. Estos permiten la ejecución de determinados comandos dentro del equipo servidor, cuyo resultado de ejecución puede ser mostrado a través de la página Web que los invoque. Aunque pueda parecer que esta solución es limitada, debido a la pequeña cantidad de comandos `SSI` existentes, su utilización da la potencia necesaria ya que uno de ellos permite la ejecución de comandos del sistema.

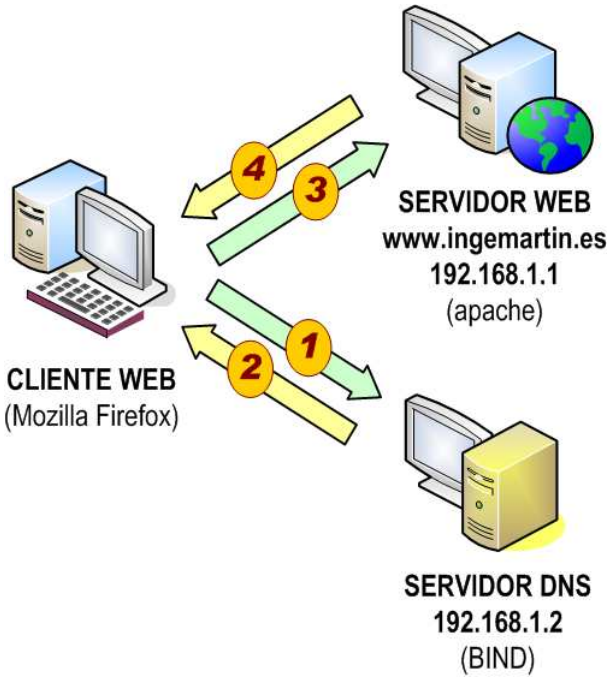
La manera de informar a `apache` de que una página Web contiene comandos `SSI` es a través de la extensión del fichero. En lugar de ser `.html` o `.htm`, debe ser `.shtml`. De esta forma, cuando `apache` recibe una solicitud `http` desde un cliente Web referente a una página `.shtml`, en lugar de descargársela directamente, en primer lugar la analiza para encontrar los comandos `SSI` que contenga. A continuación los ejecuta, y el resultado de dicha ejecución lo incluye, mediante el uso de las correspondientes etiquetas HTML, en la página Web a entregar al cliente que llevó a cabo la solicitud.

Hay que tener en cuenta que un cliente Web sólo comprende e interpreta el lenguaje HTML, lo cual implica que todo aquel contenido dentro de una página Web que no sea código HTML será ignorado por él. A través de la figura 3.8, mostrada a continuación, se trata de dar una explicación gráfica a todo lo dicho.

Para que `apache` pueda interpretar los comandos `SSI` será necesario cargar el módulo `mod_include` en el 2º bloque del fichero de configuración `httpd2.conf`:

```
#Contenido del fichero de configuración del servidor apache:
#                                        #/etc/httpd/conf/httpd2.conf
#1ER BLOQUE
...
#2° BLOQUE (Módulos DSO)
LoadModule include_module modules/mod_include.so
...
```

[15] `SSI` significa `Server Side Includes`, que en español se podría traducir por `ejecución en el lado del servidor`.

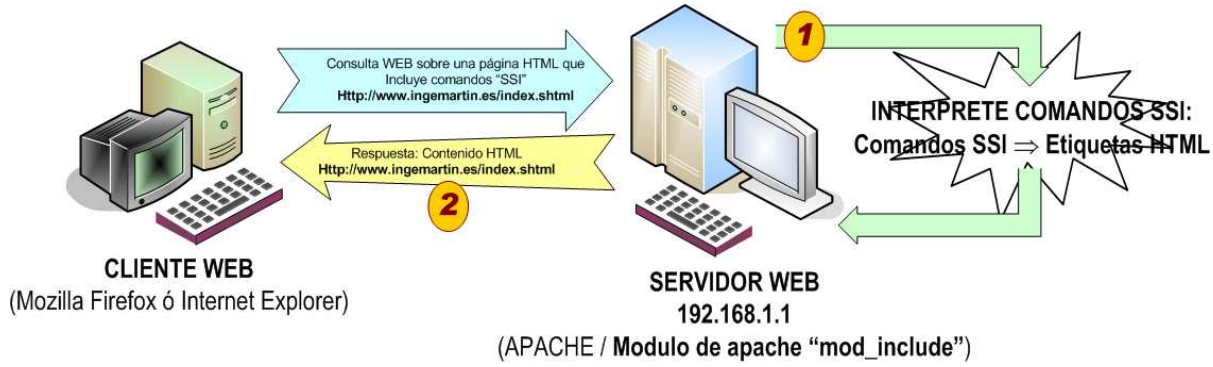


PASOS SEGUIDOS EN EL ACCESO A UN SITIO WEB ESTÁTICO:
Una vez escrito en la barra de direcciones de nuestro cliente Web preferido la `URL` correspondiente al sitio Web a visitar, `http://www.ingemartin.es`, tras pulsar `ENTER` se suceden las siguientes acciones:

1. El equipo cliente, para poder dar respuesta a la solicitud http realizada, se pregunta ¿Quién es el equipo que sirve el sitio Web `www.ingemartin.es`? Entonces se dirige al servidor `DNS` que tiene configurado para que resuelva el nombre del equipo indicado en la `URL`.
2. El servidor `DNS` nos devuelve la dirección IP del equipo que sirve el sitio Web que deseamos visualizar:

   `www.ingemartin.es` $\Longrightarrow$ `192.168.1.1`
3. El equipo cliente solicita vía protocolo `http` al equipo `192.168.1.1` el sitio Web que tiene alojado.
4. El servidor Web nos suministra la página de inicio de `ingemartin`.

PASOS SEGUIDOS EN EL ACCESO A UN SITIO WEB DINÁMICO:
Los tres primeros pasos anteriores (acceso a un sitio Web estático), correspondientes a la resolución del nombre de dominio del equipo servidor, y realización de la petición `http` al servidor Web, son independientes del tipo de sitio Web a visualizar, estático o dinámico. Una vez que el servidor ya ha recibido la solicitud:



1. El servidor Web `apache` busca la página Web de inicio del sitio Web solicitado. Al advertir de que se trata de una página con extensión `*.shtml`, por ejemplo `index.shtml`, analiza el contenido de dicha página y con el apoyo del módulo `mod_include` ejecuta todos aquellos comandos que sean encontrados. El resultado de su ejecución es incluido en la página Web correspondiente mediante la utilización de etiquetas `HTML` interpretables por el cliente Web.
2. Una vez analizado completamente el contenido de la página `*.shtml` y transformado en su totalidad en etiquetas `HTML` se le entrega al cliente Web que llevo a cabo la solicitud para que interprete su contenido y muestre al usuario correspondiente su resultado.

Figura 3.8: Acceso a un sitio Web: a) Estático, b) Dinámico

Entre las directivas asociadas al módulo `mod_include` (y otros módulos relacionados con la ejecución de comandos `SSI`) que conviene conocer podrían destacarse las siguientes:

Módulos y directivas asociadas a la ejecución de comandos `SSI`

- **Módulo: core ⟶ Directiva:Options**

Esta directiva pertene al módulo `core`. Es la encargada, entre otras cosas, de dar permiso al servidor `apache` para ejecutar los comandos `SSI` (si el permiso es afirmativo, los comandos `SSI` serán interpretados por `mod_include`). Si deseamos permitir la ejecución de comandos `SSI` deberíamos añadir a la configuración lo siguiente:

```
Options +Includes
```

Por el contrario, si no quisieramos permitirlo:

```
Options -Includes
```

Además esta directiva puede configurar el servidor `apache` para permitir la ejecución de comandos `SSI`, pero a la vez evitar la ejecución de comandos del sistema (`exec cmd`) o de `scripts CGI` (`exec cgi`).
Ejemplo:

```
Options +IncludesNOEXEC
```

- **Módulo: mod_include ⟶ Directiva:XBitHack**

La directiva `XBitHack`, perteneciente al módulo `mod_include` informa a `apache` de que todo aquel archivo (página Web, `*.html`, `*.htm`) que tenga el permiso de ejecución activado (`chmod +x` *`fichero`*) sea analizado para comprobar si contiene comandos `SSI`, y en caso de encontrarlos ejecutarlos.
Ejemplo:

```
XBitHack on
```

Mediante el uso de esta directiva no es necesario introducir la nueva extensión de archivos `.shtml` para distinguir las páginas que contienen comandos `SSI` de las que no los contienen.

- **Módulo: mod_mime ⟶ Directivas:AddType y AddOutputFilter**

En el caso de que se deseen distinguir las páginas que contienen comandos `SSI` de las que no los tienen (esta es una alternativa a la directiva `XBitHack`), deberemos crear una nueva extensión de archivos Web que por convenio es `.shtml`. Para informar a `apache` de que los archivos que encuentre con esta extensión (`.shtml`), también son páginas Web con formato HTML, será necesario hacer uso de la directiva `AddType` del módulo `mod_mime`:

```
AddType text/html .shtml
```

Además, para informar a `apache` de que dichas páginas Web (`.shtml`) contienen comandos `SSI`, y que por lo tanto, deben ser tratadas por el modulo `mod_include`, la directiva anterior, `AddType`, debe acompañarse de la directiva `AddOutputFilter`:

```
AddOutputFilter INCLUDES .shtml
```

A continuación, se mostrarán los tipos de comandos `SSI` que pueden ser incluidos en nuestras páginas Web, tras haber cargado el módulo `mod_include`. La forma de introducirlos en nuestras páginas en formato HTML es mediante el uso de las etiquetas asociadas a los comentarios (`<!--#comando SSI -->`) las cuales presentan la siguiente sintaxis (importante dejar un espacio en blanco antes de `-->`):

```
⋮                          Otras etiquetas HTML
<!--#comando argumento1="valor" argumento2="valor" -->
⋮
```

Comandos SSI

- **config**

Permite configurar el formato en el que se mostrará la fecha. Si el argumento es un archivo proporcionará el tamaño del mismo. También se utiliza para configurar los mensajes de error relacionados con problemas que puedan detectarse durante el análisis de la página Web.

Por ejemplo, si quisiéramos que se visualizase la fecha y hora de acuerdo con la norma local (`%c`) cuando se invoca el comando `date` deberíamos utilizar `config` de la siguiente forma:

```
<!--#config timefmt="%c" -->
```

Otros ejemplos:

```
<!--#config sizefmt=["bytes" "abbrev"] -->
```

```
<!--#config errmsg="Mensaje de Error" -->
```

Otros formatos disponibles para la fecha y hora son:
`%A` para mostrar el nombre completo del día de la semana en el idioma local.
`%a` igual que `%A` pero en modo abreviado.
`%B` para mostrar el nombre completo del mes en el idioma local.
`%b` igual que `%B` pero en modo abreviado.
`%d` se corresponde con el día del mes en formato decimal (01 al 31).
`%H` se corresponde con la hora en formato decimal (00 al 23).
`%I` igual que `%H` pero con formato de 12 horas (01 al 12).
`%j` se corresponde con el día del año en formato decimal (1 al 366).
`%m` se corresponde con el mes en formato decimal (01 al 12).
`%p` formato `a.m.` o `p.m.` de acuerdo con el valor local.
`%S` segundo en formato decimal.

`%w` día de la semana en formato decimal (0 ⇒ Domingo, 1 ⇒ Lunes, . . .).
`%x` fecha de acuerdo con la norma local.
`%X` hora de acuerdo con la norma local.
`%y` año en formato decimal (00 al 99).
`%Y` año completo en formato decimal (0 al 2006).
`%%` forma de representar el carácter `%`.

- **echo**

Comando encargado de mostrar el contenido de una variable. Entre las variables disponibles podrían destacarse las siguientes:
`DATE_GMT`: Fecha actual según el meridiano (`GMT` ⇒ `Greenwich Mean Time`).
`DATE_LOCAL`: Fecha actual según la zona horaria.
`DOCUMENT_NAME`: Nombre del fichero solicitado.
`DOCUMENT_URI`: La `URL` correspondiente al documento solicitado.
`LAST_MODIFIED`: Fecha de la última modificación del documento solicitado.
Ejemplo:

```
<!--#config timefmt="Hoy es %A, a %d de %B del año %Y" -->
<!--#echo var="DATE_LOCAL" -->
```

- **set**

Se utiliza para definir variables propias.
Ejemplo:

```
<!--#set var="exponsor" value="publicidad.html" -->
```

- **printenv**

Este comando se utiliza para visualizar la lista de todas las variables disponibles.
Ejemplo: `<!--#printenv -->`

- **fsize**

Nos informa del tamaño del fichero especificado (`path` ⇒ `file` o `URL` ⇒ `virtual`).
Ejemplo:

```
<!--#config sizefmt="El tamaño del fichero es [bytes abrev]" -->
<!--#fsize file="normativaISO9001.zip" -->
```

- **flastmod**

Nos informa de la fecha de la última modificación del fichero especificado (`path` ⇒ `file` o `URL` ⇒ `virtual`):
Ejemplo:

```
<!--#config timefmt="%d-%B-%Y" --> <!--#flastmod virtual="/scripts/p1.pl" -->
```

- **include**

Incluye el documento especificado como parámetro (`path` ⇒ `file` o `URL` ⇒ `virtual`). Puede resultar útil cuando un bloque de código es muy repetitivo.
Ejemplo:

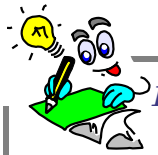
```
<!--#include file="publicidad.html" -->
```

- **exec**

El comando `exec` nos permite ejecutar tanto comandos del sistema (`cmd`) como programas externos (`cgi`). Un requisito importante que debe cumplirse para hacer uso del comando `exec` es asegurarse de que el módulo `mod_cgi` se encuentra cargado (en el 2º bloque del fichero de configuración `httpd2.conf` debe constar `LoadModule cgi_module modules/mod_cgi.so`).
Ejemplos:

```
<!--#exec cmd="/bin/more basedatos" | grep "Arturo Martin" -->
```

```
<!--#exec cgi="programa.cgi" -->
```



Ejercicio 3.11

Configura `apache` para que la página de inicio (`escarbapedal.shtml`) de nuestro sitio Web informe al usuario que acceda de la fecha, junto con la previsión del tiempo, mediante el uso de comandos `SSI`. Para distinguir las páginas Web que contienen comandos `SSI` de las que no, deberás hacer uso de la extensión convenida `.shtml`.

La fecha y hora, deberá mostrarse en el siguiente formato: `Bienvenido a nuestro sitio Web, hoy es` *`día de la semana, día del mes`* `de` *`mes`* `del año` *`año`*`.`

Supondremos que la información del tiempo, estará disponible en el fichero `/tmp/prevision_tiempo`, donde cada una de sus líneas se corresponde con la previsión del tiempo para cada uno de los días que transcurren, correspondiéndose la última de ellas, a la previsión del día actual.

Solución del ejercicio 3.11

Con la finalidad de cumplir las especificaciones impuestas será necesario modificar tanto el fichero de configuración de `apache` `httpd2.conf`, como la página de inicio de nuestro sitio Web,

inicio.shtml. Para ello, seguiremos los siguientes pasos:

1. Comenzaremos configurando apache para que las páginas de nuestro sitio Web puedan contener comandos SSI, y estos sean ejecutados al encontrarse. Para ello, asumiremos que nuestro servidor da servicio a múltiples sitios Web mediante Hosts Virtuales basados en nombre bajo la dirección IP 192.168.1.1 a través del puerto 80.

   Al imponernos, en este ejemplo, que se distinga entre páginas que contienen comandos SSI de las que no los contienen a través de la extensión del archivo (*.shtml) se deberá hacer uso de las directivas AddType y AddOutputFilter.

¡¡Advertencia!! En la versión 1.3 de apache se hacía uso de la directiva AddHandler siendo esta sustituida en la versión 2.0 por AddOutputFilter.
AddHandler server-parsed .shml ⇒ AddOutputFilter INCLUDES .shtml

Recuerda que si la anterior distinción no se quisiera llevar a cabo, y quisiéramos que cualquier página Web con extensión .html o .htm pudiese contener comandos, haríamos uso de la directiva XBitHack On, concediendo previamente el permiso de ejecución a dichas páginas (chmod +x *.html).

```
#Contenido del fichero de configuración del servidor apache:
#                                        #/etc/httpd/conf/httpd2.conf
#1ER BLOQUE
ServerRoot /etc/httpd/2.0
PidFile /var/run/httpd.pid
ErrorLog logs/error_log
LogLevel warn
#Implementamos Hosts Virtuales Basados en Nombre bajo la IP 192.168.1.1
NameVirtualHost 192.168.1.1

<VirtualHost 192.168.1.1>
  ServerName www.escarbapedal.es
  DocumentRoot /var/www/html/escarbapedal
  DirectoryIndex inicio.shtml

  <Directory /var/www/html/escarbapedal>
    #Habilitamos la opción de incluir comandos SSI:
    Options +Includes
    #Informamos a apache que los archivos con extensión .shtml
    #sean tratados como una página HTML más:
    AddType text/html .shtml
    #Informamos a apache de que los archivos con extensión .shtml
    # sean analizados por mod_include con la finalidad de ejecutar
    # todo aquel comando SSI que sea encontrado:
    AddOutputFilter INCLUDES .shtml
  </Directory>

</VirtualHost>
...
#4º BLOQUE
Listen 192.168.1.1:80
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

2. Modificaremos el contenido de nuestra página de inicio inicio.shtml para que muestre la fecha y la previsión del tiempo, con el formato especificado:

```
#Código de la Página de inicio del sitio www.escarbapedal.es: inicio.shtml
<html> <head> <meta http-equiv="Content-Language" content="es"> </head>
<body bgcolor="yellow"> <p align="center"> <hr> <p align="center">
<img src="images/escarbapedal.gif" width="200" height="80"> </p> <hr> </p>
<!--#config timefmt="Bienvenido a nuestra Web, hoy es %A, %d de %B del año %Y" -->
<p align="center"> <font size="1" color="red">
<!--#echo var="DATE_LOCAL" -->
</font> </p> <hr> <p align="center">
El tiempo que hoy vas a encontrar en la zona es:<br>
<font size="1" color="red">
<!--#exec cmd="more /tmp/prevision_tiempo | tail -1" -->
</font> </p> <hr>
...#Resto de etiquetas HTML de la página Web
```

3. Si por último reiniciamos el servicio `http` podremos comprobar desde nuestro cliente Web que los comandos `SSI` introducidos se traducen en la información esperada:

```
[root@linux]# apachectl restart #service httpd restart o /etc/init.d/httpd restart
```

Con lo que el ejercicio estaría acabado. En la figura 3.9 muestra la comprobación de la correcta configuración.

Figura 3.9: Comprobación de la ejecución de comandos SSI en apache.

Es importante advertir que si leemos el código fuente de la página Web de inicio[16] interpretada por nuestro cliente Web comprobamos que no contiene comandos `SSI`. El servidor `apache` antes de entregar la página Web solicitada por el cliente, analiza su contenido y todo aquel comando `SSI` que encuentra, lo ejecuta con la ayuda de los módulos `module_include` y `module_cgi`. El resultado de la ejecución, en formato HTML, sustituye al correspondiente comando.

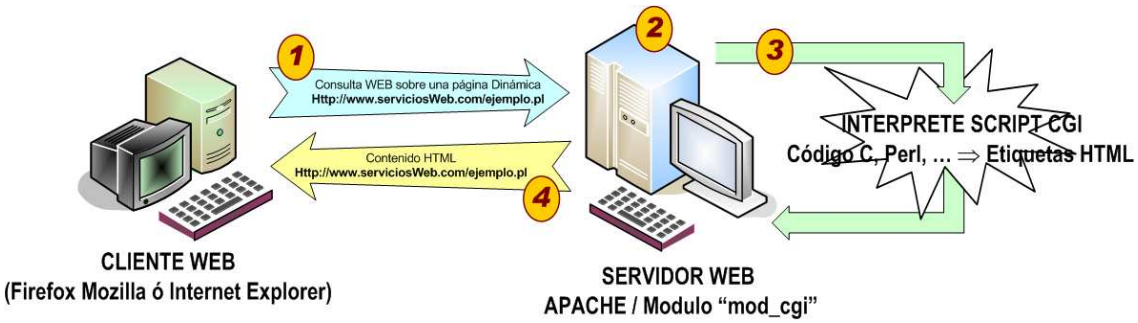
¡¡IMPORTANTE!!, la utilización de comandos `SSI` en las páginas Web alojadas en el servidor es totalmente transparente para el cliente al recibir este únicamente texto en formato HTML.

[16] Para ver el código fuente debes pulsar con el botón derecho del ratón sobre la página Web, y seleccionar la opción `Ver Código Fuente de la Página`.

3.11. Soporte para Webs dinámicas: scripts CGI

Una forma mucho más potente de dinamizar nuestro sitio Web es mediante el uso de `scripts CGI`, `Common Gateway Interface`. Estos scripts pueden corresponderse con aplicaciones desarrolladas mediante lenguajes tan dispares como `C`, `C++`, `Perl`, `Java` o el propio lenguaje de `ShellScripts de UNIX/Linux`.

En cualquier caso, al igual que ocurría con los comandos `SSI`, todo aquel `script CGI` que sea incluido dentro de una página Web servida por `apache`, deberá ser ejecutado previamente por el equipo servidor, y el resultado obtenido será incluido en formato HTML en la página Web antes de ser entregada al cliente que la solicitó.



PASOS SEGUIDOS EN LA SOLICITUD WEB DE UNA PÁGINA DINÁMICA (SCRIPT CGI):

1. El cliente Web solicita al servidor una página Web dinámica (`ScriptCGI`): `http://www.serviciosWeb.com/ejemplo.pl`
2. El servidor Web atiende la solicitud recibida, pero advierte a través de la extensión del archivo, `*.pl`, que se trata de una página Web cuyo contenido no puede ser interpretado por el cliente Web (sólo comprende etiquetas `HTML`) al tratarse de un `scriptCGI`.
3. Mediante la ayuda de un intérprete, ejecuta el `scriptCGI`, y su resultado lo incluye en la página Web en formato `HTML`.
4. La página Web solicitada, `ejemplo.pl`, en formato `HTML` es entregada al cliente.

Figura 3.10: Pasos seguidos por apache al servir una página dinámica (scripts CGI).

Para que el servidor `apache` sepa que la página que debe servir contiene `scripts CGI` se establece una distinción a través de la extensión del archivo[17]. La extensión que por convenio se utiliza, y que además está más extendida es `.pl`.

Tratar de abordar en el presente capítulo los diferentes lenguajes de programación de `scripts CGI` existentes sería desmesurado y está fuera de las pretensiones del libro, pero los ejemplos que se muestren estarán debidamente comentados.

En la mayoría de los ejemplos presentados se ha elegido el lenguaje `Perl` por tratarse, posiblemente, del lenguaje más estandarizado para este tipo de aplicaciones, aunque también se mostrará algún ejemplo mediante el uso de `shellscritps` al tratarse de la herramienta habitual de programación de tareas bajo GNU/Linux.

Para que `apache` ejecute los scripts tan sólo será necesario hacer uso de la directiva `ScriptAlias`. Esta nos permite crear un `alias` referente al directorio del sistema que contiene los `scripts CGI` que deseamos que sean ejecutados. Todos aquellos scripts que no se localicen en el directorio asociado al `alias` creado serán ignorados por `apache` y por tanto no ejecutados.

Por ejemplo, si quisiéramos poder ejecutar `scripts CGI` dentro de nuestro sitio Web, colocando como `URL` `http://www.escarbapedal.es/scripts/ejemplo1.pl`, sería necesario crear un `alias` mediante `ScriptAlias` llamado `scripts` que apunte al directorio donde se localiza el script `ejemplo1.pl` haciendo uso de la siguiente sintaxis:

```
ScriptAlias /alias/ directorio de los scripts CGI a ejecutar
```

[17] Esta es una situación similar a la que ocurría con los comandos `SSI`.

Ejemplo:

```
ScriptAlias /scripts/ /var/www/html/escarbapedal/programas/shellscript/
```

¡¡Advertencia!! Observar que el alias debe estar encerrado entre `/`, y que la ruta del directorio que alberga los `scripts CGI` debe acabar igualmente en `/`.

Por el contrario, si deseamos que no puedan ejecutarse `scripts CGI` dentro de una determinada ubicación, habrá que indicarlo explícitamente, ya que por defecto, todo aquel script localizado dentro del directorio apuntado por `ScriptAlias` será ejecutado si se solicita a `apache`. Para ello haremos uso de la directiva `Options -execCGI`, cuyo significado ya se ha visto en la página 127.

3.11.1. Scripts CGI: Shellscripts

Asumiendo que el sistema operativo sobre el cual está ejecutándose nuestro servidor Web `apache` es un GNU/Linux (o UNIX), la forma más sencilla de interactuar con este es haciendo uso del lenguaje que sabe interpretar GNU/Linux de manera nativa, los scripts de shell sh, csh, bash, ...

¡¡Ayuda!! Para saber qué intérpretes de comandos (shells) tenemos disponibles en nuestro equipo GNU/Linux es necesario listar el contenido del fichero de configuración del sistema `/etc/shells`:

```
[root@linux]# more /etc/shells
```

Debido a que nuestro equipo GNU/Linux puede trabajar con diversos intérpretes de comandos, al comienzo del script (primera línea) será necesario indicar que `shell` deseamos que lo ejecute mediante el uso de una línea comentada a través de la siguiente sintaxis:

```
#!ruta donde se encuentra el intérprete de comandos.
```

Por ejemplo:

```
#!/bin/sh
```

A continuación habrá que informar al intérprete de comandos de que la finalidad del script es que el resultado de su ejecución sea visualizado por un intérprete HTML (cliente Web). Es decir, que en lugar de visualizarse en la salida estándar (monitor/pantalla) de GNU/Linux, será enviado vía protocolo `http`.

Para ello será necesario agregar en nuestro script la línea, `echo "Content-type: text/html"`. En caso contrario su resultado será ignorado por el servidor, mostrándonos un error advirtiéndonos de que no comprende el resultado del script.

Además, deberán tenerse en cuenta dos cosas,

1. No debe intentarse visualizar ningún resultado antes de su declaración, admitiéndose únicamente la declaración de variables.

2. Debe respetarse una línea explícitamente vacía a continuación (`echo ""`):

```
#!/bin/sh
...#Declaracion de variables
echo "Content-type: text/html"
echo ""
...#Contenido de Shell Script
```

Ejercicio 3.12

Configura `apache` para que el sitio Web `www.escarbapedal.es` al que da servicio pueda albergar `scripts CGI` (`/var/www/html/escarbapedal/programas/shellscripts`) con la finalidad de poder consultar una pequeña base de datos (fichero plano de texto `/var/www/html/escarbapedal/informacion/ciclistas_btt.db`) a través de un formulario HTML al que se accede a través de la página de inicio del sitio Web (`/var/www/html/escarbapedal/formulario1.html`). En concreto, el script encargado de llevar a cabo la consulta será `consulta.pl`, y deberá poder ser ejecutado vía Web a través de la URL: `http:www.escarbapedal.es/scripts/consulta.pl`

Con respecto a la estructura del fichero `ciclistas_btt.db` asumiremos que es la siguiente:

```
<nº socio>:<nombre>: <apellidos>:<DNI>:<edad>:<telf>:
                          <dirección>:<ciudad>:<provincia>:<país>
```

Solución del ejercicio 3.12

A continuación, se mostrarán que modificaciones hay que realizar en la configuración de `apache` y el script necesario para llevar a cabo la consulta. En la solución proporcionada se asumirán conocimientos tanto del lenguaje de etiquetas HTML como de `shellscript`.

1. Comenzaremos configurando `apache` para que puedan ejecutarse `scripts CGI` en el sitio Web `www.escarbapedal.es` localizados en la ruta especificada en el enunciado del ejercicio práctico: `/var/www/html/escarbapedal/programas/shellscripts`. Asumiremos que este es uno de tantos sitios Web que pueden ser servidos simultáneamente desde `apache` haciendo uso de `Hosts Virtuales basados en nombre`.

```
#Contenido del fichero de configuración del servidor apache:
#                                        #/etc/httpd/conf/httpd2.conf
#1ER BLOQUE
ServerRoot /etc/httpd/2.0
PidFile /var/run/httpd.pid
...#Implementamos dos Hosts Virtuales Basados en Nombre bajo la IP 192.168.1.1
NameVirtualHost 192.168.1.1
...#Resto de Hosts Virtuales
<VirtualHost 192.168.1.1> #Host Virtual correspondiente a www.escarbapedal.es
  ServerName www.escarbapedal.es
  DocumentRoot /var/www/html/escarbapedal
  #Creamos un alias para que apache sepa localizar los scripts CGI a ejecutar:
  ScriptAlias /scripts/ /var/www/html/escarbapedal/programas/shellscripts/
  Options +execCGI #Este es su valor por defecto
  #Asumimos que la página de inicio sigue haciendo uso de SSI:
  DirectoryIndex inicio.shtml

  <Directory /var/www/html/escarbapedal>
    Options +Includes
    AddType text/html .shtml
    AddOutputFilter INCLUDES .shtml
  </Directory>

</VirtualHost>
...
#4º BLOQUE
```

Continúa en página siguiente

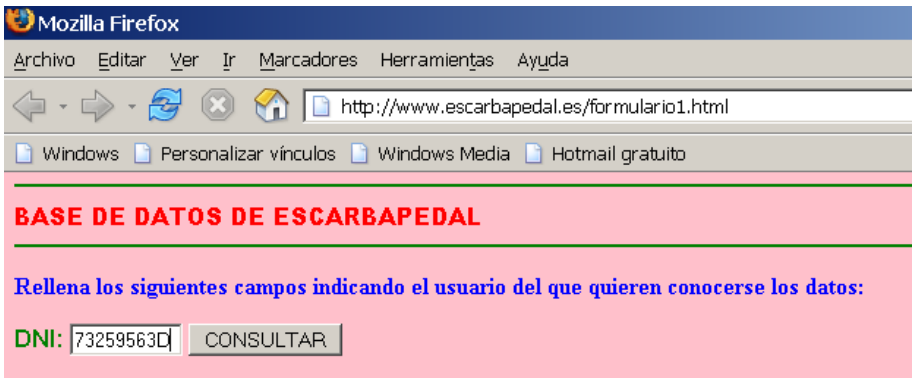
```
Listen 192.168.1.1:80

#El resto de parámetros se pueden dejar a su valor por defecto
```

Como puede observarse en la solución proporcionada se han añadido dos directivas dentro del `VirtualHost` asociado al sitio Web `www.escarbapedal.es`: `ScriptAlias` y `Options`, de las cuales solamente es obligatoria `ScriptAlias` ya que por defecto esta habilitada la opción de que se ejecuten los `scripts CGI` (`Options +execCGI`).

2. La forma más habitual de realizar consultas vía Web es mediante el uso de formularios Web HTML. A continuación se mostrará un formulario compuesto por un único campo (DNI), para que el usuario pueda introducir la clave primaria DNI correspondiente al ciclista del club `escarbapedal` del que desea conocer sus datos:

```
<!--#Contenido de formulario1.html, sitio Web: www.escarbapedal.es -->
<html> <head> </head> <body bgcolor="pink">
<hr color="green">
<font color="red" face="arial black">BASE DE DATOS DE ESCARBAPEDAL</font>
<hr color="green">
<p face="arial"> <font color="blue"> <B>
Rellena los siguientes campos indicando el usuario del que quieren
conocerse los datos:</font> </B> </p>
<form method="POST" action="/scripts/consulta.pl">
<font face="arial" color="green"> <B>DNI: </B> </font>
<input type="text" name="dni" maxlength="9" size="9">
<input type="submit" value="CONSULTAR">
</form> </body>
```

En la siguiente figura se muestra el aspecto del formulario HTML programado.



De todo el contenido de la página Web `formulario1.html` cabría destacar la línea que genera él formulario: `<form method="POST" action="/scripts/p3.pl">`.

En ella se define el método (method) a través del cual se enviará la información del formulario (GET o POST), y el destinatario (action). En relación al método de envió, por sus ventajas haremos uso de aquí en adelante del método `POST`.

Nota:Las ventajas del método POST son que no hay límite en la cantidad de información a enviar y que no es visible a través de la URL.

En cuanto al parámetro `action` aclarar que en ningún caso colocaremos la ruta real del shell-script encargado de manipular la información introducida en el formulario, sino la ruta virtual haciendo uso del `alias` definido mediante `ScriptAlias`. Es decir, si el `shellscript` se encuentra en `/var/www/html/escarbapedal/programas/shellscripts/consulta.pl` no escribiremos `action='/programas/shellscripts/consulta.pl'`, sino `action='/scripts/consulta.pl'`, ya que `/scripts/` es el `alias` asociado a la ruta real.

3. Por último, generaremos el `shellscript` `consulta.pl` encargado de recoger la información enviada desde el formulario anterior, a través de la cual consultará el fichero `ciclistas_btt.db` para extraer la información del socio solicitado, y mostrarlo en la pantalla.

```
Contenido del fichero de socios del club ciclista escarbapedal: ciclistas_btt.db:
⋮
22:Julian:Cerio Justo:4528125P:32:32786627:Pl. España,n15:Hijar:Teruel:Es
23:Arturo:Martin Romero:1242411Z:29:635613131:C/Mayor,n12-1ºA:Alcañiz:Teruel:Es
⋮
```

Antes de escribir el `shellscript` `consulta.pl` es conveniente conocer algunos de los comandos necesarios. Con este fin se ha incluido el apéndice B (página 293) en el que puedes encontrar una pequeña descripción de los mismos. Es muy recomendable que le *eches un vistazo*.

El shellScript, `consulta.pl`, encargado de consultar e informar del usuario especificado (DNI) sería:

```
#!/bin/bash
echo "Content-type: text/html"
echo ""
echo "<html> <head> </head> <body bgcolor='yellow'>"
echo "<hr color='red'> <font face='arial black' color='brown'>EL
RESULTADO DE LA CONSULTA ES EL SIGUIENTE </font> <hr color='red'>"
read DNI_SOCIO
DNI_SOCIO=`echo $DNI_SOCIO | cut -d"=" -f2`
if [ `cat /var/www/html/escarbapedal/informacion/ciclistas_btt.db | grep
"$DNI_SOCIO" | tee /tmp/resultado | wc -l` -eq 1 ]
then
NOMBRE_COMPLETO=`cat /tmp/resultado | cut -d":" -f1,2 | tr -s ":" " "`
TELEFONO=`cat /tmp/resultado | cut -d":" -f5`
DIRECCION=`cat /tmp/resultado | cut -d":" -f6,7,8 | tr -s ":" " "`
echo "<table border='2'>"
echo "<tr align='center' bgcolor='pink'>"
echo "<td> <font face='arial' color='blue'>NOMBRE COMPLETO</font> </td>"
echo "<td> <font face='arial' color='blue'>TELEFONO</font> </td>"
echo "<td> <font face='arial' color='blue'>DIRECCION</font> </td>
</tr>"
echo "<tr bgcolor='orange'>"
echo "<td> <font face='arial' color='blue'>$NOMBRE_COMPLETO</font>
</td>"
echo "<td> <font face='arial' color='blue'>$TELEFONO</font> </td>"
echo "<td> <font face='arial' color='blue'>$DIRECCION</font> </td>"
echo "</tr> </table>"
else
echo "Lo sentimos, pero el DNI $DNI_SOCIO no se corresponde con ninguno
de los socios registrados. GRACIAS."
fi
rm -f /tmp/resultado
echo "</body> </html>"
```

Como puede observarse, para que el contenido del `shellscript` pueda ser interpretado y visualizado por un cliente Web, es necesario indicar al comienzo del script su intérprete (`#!/bin/bash`), y el formato de salida (`echo "Content-type: text/html"`), atendiendo en todo momento a que si este es HTML, deberemos asegurarnos de que se impriman (echo) las correspondientes etiquetas, <html>,<head>, <body>, ...De esta forma, el servidor `apache` al recibir una solicitud sobre una página Web correspondiente con un `script CGI` se limita a leer la primera línea del script para conocer quien debe interpretar el código y cedérselo. El resultado de su ejecución (etiquetas HTML) será suministrado al cliente Web (Mozilla Firefox).

¡¡Muy Importante!! Para que la ejecución del `script CGI` resulte exitosa deberán tenerse en cuenta los siguientes puntos:

1. Nos aseguraremos de que el servicio `httpd` ha sido reiniciado una vez realizados los correspondientes cambios en su configuración (`apachectl restart`).
2. Garantizaremos que el propietario de los scripts es el usuario del sistema `apache` con la finalidad de evitar problemas de privilegios y seguridad.
3. Deberemos conceder permisos de ejecución (`+x`) al script para que sea visto por `apache` como un ejecutable.

Si repasamos el código anterior, las operaciones realizadas son las siguientes:

1. Recogemos la información enviada desde el formulario:

```
read DNI_SOCIO
```

Teniendo en cuenta que el formulario esta compuesto por un único campo de tipo `text` (`input text`) llamado `dni` (name=dni), y que el valor introducido es `1242411Z` (value=1242411Z), si visualizáramos el valor de la variable su resultado sería:

```
[user@linux]$ echo $DNI_SOCIO
dni=1242411Z
```

2. Extraemos el número de identificación de la variable. Para ello asumimos que el contenido de la variable `DNI_SOCIO` es un registro compuesto por campos separados por un carácter `=`, por lo que deberemos cortarlo (cut) quedándonos con la segunda columna:

```
DNI_SOCIO=`echo $DNI_SOCIO | cut -d"=" -f "2"`
```

De esta forma, si tras ejecutar la instrucción anterior volviésemos a visualizar el contenido de la variable `DNI_SOCIO` obtendríamos únicamente el número:

```
[user@linux]$ echo $DNI_SOCIO
1242411Z
```

¡¡Muy Importante!! En los Shell Scripts se distingue entre comillas dobles ("), comillas simples (') y comillas graves (`).
Comillas simples ('): Todos aquellos comandos de Linux que incorporen metacaracteres (*, ?, []) o caracteres especiales ($, \, <, >) perderán su funcionalidad.
Comillas dobles ("): Permiten interpretar los caracteres especiales introducidos en los comandos pero no los metacaracteres (* y ?). Además anulan el efecto de los espacios en blanco.
Comillas graves (`): Permite la ejecución de cualquier comandos pudiendo introducir en él tanto caracteres especiales como metacaracteres, pudiendo asignar su resultado a una variable.

3. Comprobamos si el DNI del socio introducido en el formulario se corresponde con uno de los socios que se encuentra registrado en la base de datos `ciclistas_btt.db`. Para ello hacemos uso del comando `grep`:

```
[user@linux]$ cat /var/www/html/escarbapedal/informacion/ciclistas_btt.db
| grep "$DNI_SOCIO" | tee /tmp/resultado | wc -l
```

Concretamente listamos el contenido de `ciclistas\_btt.db` haciendo uso de `cat` y a continuación pasamos su resultado mediante una tubería (|) al comando `grep`. Este filtra las líneas de la base de datos, dejando únicamente aquella que contiene el DNI introducido (DNI_SOCIO). El resultado del filtrado se almacena en un fichero temporal (`/tmp/resultado`), y observamos si se ha encontrado alguna coincidencia contando el número de líneas filtradas (`wc -l`). El resultado final, tan sólo puede ser 1 en el caso de que haya encontrado un socio con ese DNI, ó 0, en caso de no hallar ninguno (al tratarse de una clave primaria es imposible que se encuentren más de 1).

¡¡Aclaración!! El comando `tee` permite almacenar en el fichero especificado el resultado de un tratamiento de información parcial entre tuberías.

4. Comprobamos el número de coincidencias (nº de socios con el DNI especificado) mediante uso del comando `test` (equivalente a `[ <condición> ]`), y en función de su valor, mediante un `if` condicionamos el conjunto de instrucciones que serán ejecutadas posteriormente.

```
if [ `cat /var/www/html/escarbapedal/informacion/ciclistas_btt.db | grep
"$DNI_SOCIO" | tee /tmp/resultado | wc -l` -eq 1 ]
then
  Instrucciones ejecutadas en caso de coincidencia
else
  Instrucciones ejecutadas en caso de no coincidencia
fi
```

5. En el caso en que se haya encontrado un socio con el DNI indicado, el fichero temporal `/tmp/resultado` contendrá sus datos con el siguiente formato:

```
23:Arturo:Martin Romero:1242411Z:29:635613131:C/Mayor,n12-1ºA:Alcañiz:Teruel:Es
```

Suponiendo que tan sólo nos interesa el nombre del socio, su teléfono y dirección, filtramos el contenido del fichero `/tmp/resultado` mediante los siguientes comandos:

```
NOMBRE_COMPLETO=`cat /tmp/resultado | cut -d":" -f2,3 | tr -s ":" " "`
TELEFONO=`cat /tmp/resultado | cut -d":" -f6`
DIRECCION=`cat /tmp/resultado | cut -d":" -f7,8,9 | tr -s ":" " "`
```

El comando `cat` lista el contenido del fichero `/tmp/resultado` entregándoselo mediante el uso de una tubería (|) al comando `cut`, el cual, al informarle que su contenido está dividido en campos delimitados por el carácter `:` (`-d":"`) rescatará los campos deseados en función del número de columna que ocupen (-f*n^o columna*). En el caso de que el número de campos rescatados sea más de uno, será necesario eliminar el carácter delimitador que los separa mediante el comando `tr`, sustituyendo los `:` por un espacio en blanco.

Por el contrario, en el caso de que no exista ningún socio con el DNI indicado, se mostrará un mensaje advirtiendo de ello.

6. Por último, en caso exitoso, imprimimos (`echo`) una tabla en formato HTML cuyo contenido sean los campos filtrados con el comando `cut` anterior:

```
echo "<table border='2'>"
echo "<tr align='center' bgcolor='pink'>"
echo "<td> <font face='arial' color='blue'>NOMBRE COMPLETO</font> </td>"
echo "<td> <font face='arial' color='blue'>TELEFONO</font> </td>"
echo "<td> <font face='arial' color='blue'>DIRECCION</font> </td> </tr>"
echo "<tr bgcolor='orange'>"
echo "<td> <font face='arial' color='blue'>$NOMBRE_COMPLETO</font> </td>"
echo "<td> <font face='arial' color='blue'>$TELEFONO</font> </td>"
echo "<td> <font face='arial' color='blue'>$DIRECCION</font> </td>"
echo "</tr> </table>"
```

Ejercicio 3.13

Crear un `script CGI` llamado `gestionWeb.pl` ubicado en `/var/www/html/escarbapedal/programas/gestion/gestionweb.pl` basado en `shellscript` que permita administrar el sistema de ficheros del servidor (por ejemplo, crear y borrar directorios) mediante el uso de un formulario. Para acceder a él se añadirá en la página de inicio del sitio Web correspondiente un hipervínculo (`<a href.="/scripts-gestion/gestionweb.pl">GESTION WEB</a>`).

El acceso a dicho script deberá estar restringido a los administradores del servicio (no anónimo) con la finalidad de evitar acciones malintencionadas.

Solución del ejercicio 3.13

Mediante la creación y eliminación de directorios, se mostrará la manera de poder gestionar vía Web un equipo servidor (observa que esto es fácilmente extensible a realizar cualquier otra operación de control remoto).

1. Al tratarse de un `script CGI` de ejecución no anónima, deberemos configurar el servidor `apache` para que el acceso al directorio donde se encuentra quede restringido a los administradores del sistema (ver apartado 2.5). Además será necesario añadir un nuevo `ScriptAlias` que haga referencia al directorio que contiene los scripts de gestión del servidor:

```
#Contenido del fichero de configuración del servidor apache:
#                                                  #/etc/httpd/conf/httpd2.conf
#1ER BLOQUE
ServerRoot /etc/httpd/2.0
```

Continúa en página siguiente

```
PidFile /var/run/httpd.pid
...
#Implementamos dos Hosts Virtuales Basados en Nombre bajo la IP 192.168.1.1
NameVirtualHost 192.168.1.1
#Host Virtual correspondiente a www.escarbapedal.es:
<VirtualHost 192.168.1.1>
ServerName www.escarbapedal.es
DocumentRoot /var/www/html/escarbapedal
#Creamos dos alias para indicar a apache donde están los scripts CGI a ejecutar:
ScriptAlias /scripts/ /var/www/html/escarbapedal/programas/shellscripts/
ScriptAlias /scripts-gestion/ /var/www/html/escarbapedal/programas/gestion/
Options +execCGI #Le estamos dando su valor por defecto
#Asumimos que la página de inicio sigue haciendo uso de SSI
DirectoryIndex inicio.shtml
<Directory /var/www/html/escarbapedal>
Options +Includes
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
</Directory>
#Damos acceso restringido al directorio contenedor de los scripts de gestion:
<Directory /var/www/html/escarbapedal/programas/gestion>
AuthType Basic
AuthDBMType DB
AuthName "Solo Autorizados"
AuthDBMUserFile seguridad/administradoresWeb
Require valid-user
</Directory>
</VirtualHost>
...#4º BLOQUE
Listen 192.168.1.1:80
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

2. Una vez configurado `apache` y reiniciado el servicio (`apachectl restart`), crearemos el `shellscript` que se encargará de la gestión del sitio Web. A continuación se muestra el ShellScript `gestionweb.pl` encargado de la gestión de directorios:

```
#!/bin/bash
echo "Content-type: text/html"
echo ""
echo "<html> <head> </head> <body bgcolor='orange'>"
echo "<hr color='red'> <font face='arial black' color='brown'>GESTOR WEB DEL
SITIO WEB ESCARBAPEDAL </font> <hr color='red'>"
echo "<form method="POST" action='/scripts-gestion/gestionweb.pl'>"
echo "<font face='arial' color='brown'>"
echo "<B>SELECCIONA LA ACCION A REALIZAR: "
echo "<select name='opcion' size='1'>"
echo "<option selected> </option>"
echo "<option>CREAR DIRECTORIO</option>"
echo "<option>BORRAR DIRECTORIO</option> <br>"
echo "</select> <br> <br>"
echo "NOMBRE DEL DIRECTORIO: </B> </font> <input type='text'
name='nombre' size='30'> <br>"
echo "<p align='center'> <input type='submit' value='EJECUTAR'>
</p> <br> <br> </form>"
read DATOS_FORMULARIO
```

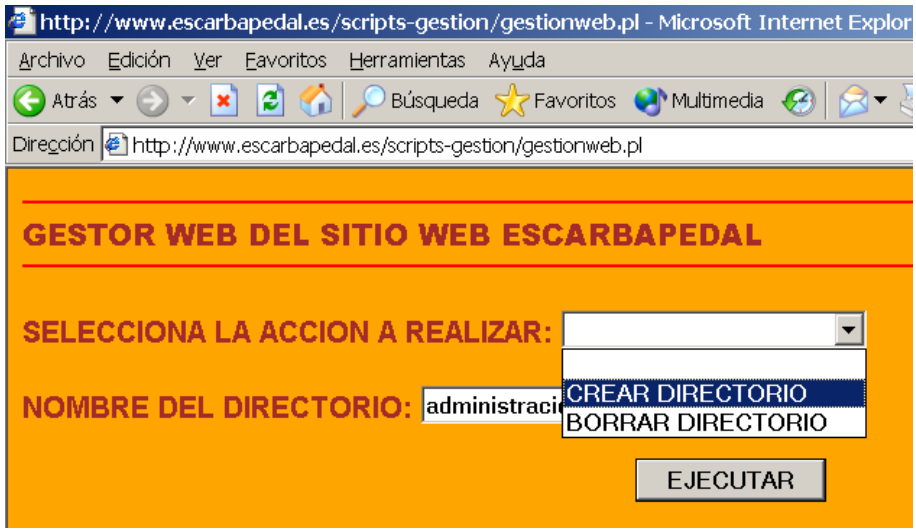
Continúa en página siguiente

```
OPCION=`echo $DATOS_FORMULARIO | cut -d"&" -f"1" | cut -d"=" -f"2" | tr -s "+" " "`
NOMBRE=`echo $DATOS_FORMULARIO | cut -d"&" -f"2" | cut -d"=" -f"2" | tr -s "+" " "`
if [ "$OPCION" != "" -a "$NOMBRE" != "" ]
then
  case "$OPCION" in
   "CREAR DIRECTORIO") mkdir $NOMBRE
     if [ $? -eq 0 ]
     then
       echo "<br> <B>SE HA CREADO PERFECTAMENTE...</B>"
     else
       echo "<br> <B>SE HA PRODUCIDO UN ERROR...</B>"
     fi ;;
   "BORRAR DIRECTORIO") rm -fR $NOMBRE ;;
     if [ $? -eq 0 ]
     then
       echo "<br> <B>SE HA BORRADO SIN PROBLEMAS...</B>"
     else
       echo "<br> <B>SE HA PRODUCIDO UN ERROR...</B>"
     fi ;;
  esac
else
echo "<br> <p align='center'>POR FAVOR RELLENA EL FORMULARIO
CORRECTAMENTE.</p>"
fi
echo "</body> </html>"
```

Al igual que en el ejercicio práctico anterior, el script comienza informando del intérprete que deberá encargarse de ejecutarlo y transformarlo en formato HTML, `#!/bin/bash`. A continuación añadiremos `echo "Content-type: text/html"` para informar al intérprete que el resultado de la ejecución será entregado al cliente Web en formato HTML que él comprende. Importante respetar la línea en blanco a continuación de la instrucción anterior, `echo ""`. En cuanto al resto del script diremos que:

a) El propio `shellscript` genera (`echo`) el formulario HTML que permitirá al usuario decidir la acción a llevar a cabo, `crear directorio` o `borrar directorio`. Para ello el formulario se compone de dos campos, un `select` (`combobox`) que nos dará a elegir entre las dos opciones disponibles (option), y un caja de texto (`input text`) a través del cual indicaremos el nombre del directorio a crear o borrar.

```
echo "<form method="POST" action='/scripts-gestion/gestionweb.pl'>"
echo "<font face='arial' color='brown'>"
echo "<B>SELECCIONA LA ACCION A REALIZAR: "
echo "<select name='opcion' size='1'>"
echo "<option selected> </option>"
echo "<option>CREAR DIRECTORIO</option>"
echo "<option>BORRAR DIRECTORIO</option> <br>"
echo "</select> <br> <br>"
echo "NOMBRE DEL DIRECTORIO: </B> </font> <input type='text' name='nombre'
                                    size='30'> <br>"
echo "<p align='center'> <input type='submit' value='EJECUTAR'> </p>
                                     <br> <br> </form>"
```



b) Tras el formulario, mediante un `read` recogemos en la variable `DATOS_FORMULARIO` la información enviada por éste. En caso de que no haya sido pulsado el botón `submit EJECUTAR` la variable estará vacía, y en caso contrario contendrá la opción seleccionada y el nombre del directorio indicado.

c) A continuación manipularemos el contenido de la variable `DATOS_FORMULARIO`. Si visualizáramos su contenido presentaría el siguiente formato (por ejemplo, si seleccionásemos la opción `CREAR DIRECTORIO` y escribieramos como nombre del directorio `administracion`):

```
[user@linux]$ echo $DATOS_FORMULARIO
"opcion=CREAR+DIRECTORIO&nombre=administracion"
```

Según esto, para recuperar a través de la variable `DATOS_FORMULARIO` la opción seleccionada y el nombre del directorio, deberemos filtrar su contenido como si se tratase de un fila con dos campos (columnas) divididas por el carácter `&`. Mediante el comando `cut` seleccionamos la columna, y con `tr` sustituímos los signos `+` por espacios en blanco.

```
OPCION=`echo $DATOS_FORMULARIO | cut -d"&" -f"1" | cut -d"=" -f"2" | tr -s "+" " "`
NOMBRE=`echo $DATOS_FORMULARIO | cut -d"&" -f"2" | cut -d"=" -f"2" | tr -s "+" " "`
```

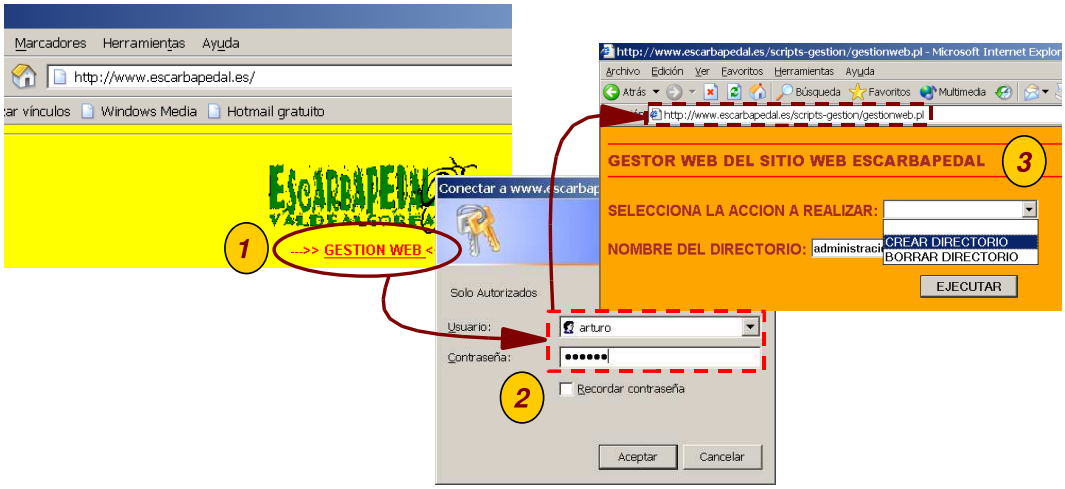
d) Para comprobar que la información introducida en el formulario no es incongruente ($OPCION y $NOMBRE) hacemos uso de la cláusula `if`. En concreto se comprueba mediante una `AND` lógica (`-a`) que se ha seleccionado una de las opciones, y que el campo asociado al nombre del directorio no esta vacio:

```
if [ "$OPCION" != "" -a "$NOMBRE" != "" ]
then
   Instrucciones a ejecutar en caso de rellenar correctamente el formulario
else
   Instrucciones a ejecutar en caso de no rellenar correctamente el formulario
fi
```

e) Mediante una cláusula `case` distinguiremos entre la opción seleccionada `CREAR DIRECTORIO` (que implica `mkdir $NOMBRE`) y `BORRAR DIRECTIRIO` (que implica `rm -fR $NOMBRE`). Para comprobar que la creación o borrado del directorio indicado se ha realizado con éxito, se analiza el contenido de la variable `$?` . En el caso en que la ruta del nombre del directorio sea relativa (si la ruta empieza por `/` será absoluta, y sino relativa) será en relación a la raíz del `shellscript`.

Nota: La variable `$?` toma el valor cero `0` en el caso de que se haya ejecutado con éxito el último comando. En caso contrario, contendrá un valor distinto.

```
case "$OPCION" in
  "CREAR DIRECTORIO") mkdir $NOMBRE
    if [ $? -eq 0 ]
    then
      echo "<br> <B>SE HA CREADO PERFECTAMENTE...</B>"
    else
      echo "<br> <B>SE HA PRODUCIDO UN ERROR...</B>"
    fi ;;
  "BORRAR DIRECTORIO") rm -fR $NOMBRE ;;
    if [ $? -eq 0 ]
    then
      echo "<br> <B>SE HA BORRADO SIN PROBLEMAS...</B>"
    else
      echo "<br> <B>SE HA PRODUCIDO UN ERROR...</B>"
    fi ;;
esac
```



Por último, aclarar que todas las acciones que se llevan a cabo sobre el sistema se hacen en nombre del usuario `apache`. Esto significa que el límite de tales acciones lo determinan los privilegios concedidos a este usuario dentro del sistema.

En el caso de desear aumentar la potencia de las acciones que puedan desencadenarse a través de un `script CGI` podríamos dar privilegios al usuario `apache` mediante `sudo` o suplantar a otro usuario con privilegios para llevar a cabo las tareas deseadas. En ambos casos hay que tener en cuenta que puede suponer un agujero de seguridad para nuestro servidor.

3.11.2. Scripts CGI: PERL

El lenguaje `perl` surgió como una alternativa a los lenguajes de shell `shellscript` (sh, csh, ksh, bash, ...) con la finalidad de aumentar sus posibilidades. Presenta una sintaxis similar a estos, pero ofreciendo una potencia comparable con lenguajes como `C`. Esto se traduce en que los scripts desarrollados en `perl` pueden llegar a ser más versátiles y potentes.

A continuación se mostrará algún ejemplo de programación en lenguaje `perl`. No se pretende explicar este lenguaje, sólo se desea familiarizar al lector con su sintaxis. Aunque la utilización de lenguajes como PHP está desbancando a `perl`, no deberíamos menospreciarlo al tratarse del

lenguaje más ampliamente usado en la realización de `scripts CGI`. En el caso de que no conozcas `PERL` es muy recomendable que leas el apéndice C que puedes encontrar en la página 297. En este apéndice se hace un pequeño resumen de algunos comandos `PERL` que emplearemos en los ejercicios propuestos.

Con respecto a la configuración de `apache`, señalar que no será necesario introducir nada nuevo. Esto se debe que la utilización de `scripts CGI` en `apache` es independiente del lenguaje utilizado para su realización (lo explicado en la sección anterior es válido).

Ejercicio 3.14

Crear una página Web haciendo uso de un `script CGI` escrito en lenguaje `perl` llamado `examenWeb.pl` que permita examinar a los usuarios que accedan a ella (acceso no anónimo). El examen Web estará formado por un formulario Web compuesto por 10 cuestiones relacionadas con la materia a examinar. En concreto el script deberá manipular tres ficheros de texto (la extensión `*.txt` en GNU/Linux no es significativa, sólo aclaratoria) que a continuación se describen.

> `cuestiones-examen.txt`: Contiene las cuestiones que componen el examen. Cada línea del fichero se corresponderá con una cuestión diferente. Estas estarán constituidas por diversos campos (columnas) separados por el carácter `";"`. Los campos contenidos en cada línea son: `<numero pregunta>;<cuestion>;<identificador de 1ª respuesta>;<1ª respuesta>;<identificador de 2ª respuesta>;<2ª respuesta>;<identificador de 3ª respuesta>;<3ª respuesta>`.
>
> Nota: Los símbolos < y > no deben aparecer en el fichero. Únicamente se han puesto aquí para que distingas cláramente cada uno de los campos.

Un ejemplo podría ser el siguiente:

```
pregunta1;P1.-Indica que características son ideales para todo amplificador
operacional:;R11;A) Impedancia de entrada y de salida muy altas.;R12;B) Ganancia
en tensión e impedancia de salida muy altas.;R13;C) Impedancia de salida nula
y de entrada infinita.

pregunta2;P2.-Indica qué zonas lineales de trabajo se distinguen dentro de las
curvas que caracterizan a un transistor bipolar:;R21;A) Zonas Ohmica, Saturación
y Corte.;R22;B) Zonas Saturacion, Actica y Corte.;R23;C) Zonas Óhmica, Activa
y Corte.

pregunta3;P3.-Indica cuál es el significado del parámetro Slew Rate:;R31;A) In-
forma de la ganacia típica a bajas frecuencias.;R32;B) Informa de las condiciones
bajo las cuales un AO se satura en intensidad.;R33;C) Informa de la velocidad
máxima a la que puede cambiar de posición la tensión de salida.
...
```

> `respuestas-examen.txt`: Contiene las respuestas correctas a las cuestiones planteadas en `cuestiones-examen.txt`. En concreto, el fichero está compuesto por tantas filas como respuestas, correspondiéndose estás con el `identificador` de la respuesta correcta.

Tomando el ejemplo anterior, el fichero `respuestas-examen.txt` sería de la forma:

```
R13
R22
R33
...
```

`notas-examen.txt`: Contiene los usuarios o alumnos que ya han sido evaluados vía Web, junto con su nota correspondiente. Este fichero además de registrar las notas obtenidas nos permitirá comprobar si el usuario o alumno que trata de acceder al examen vía Web ya lo ha realizado previamente, evitando que puedan examinarse más de una vez. Su contenido estará compuesto por tantas filas como usuarios examinados, estando formadas cada una de ellas por dos campos: <nombre de usuario/alumno>:<nota obtenida>.

Por ejemplo:

```
mariapineda:9
juliocastellano:8
...
```

Aunque la raíz del sitio Web que contiene dicha página (examenWeb.pl) deba ser `/var/www/html/ingemartin`, la cambiaremos para que esté localizada en `/var/www/html/ingemartin/examenes/curso1`. Por último, deberás configurar `apache` para que todos los usuarios a examinarse puedan acceder a través de la URL: `http://www.ingemartin.es/examen-curso1/examenWeb.pl`.

Solución del ejercicio 3.14

En primer lugar se mostrarán los cambios a realizar en fichero de configuración de `apache httpd2.conf` y a continuación el contenido del `script CGI` en lenguaje `perl` que permita examinar al usuario que se autentifique:

1. En primer lugar configuraremos `apache` para dar servicio al sitio Web `www.ingemartin.es` mediante un `Host Virtual basado en nombre` (por ejemplo, bajo la IP 192.168.1.1):

```
#Contenido del fichero de configuración del servidor apache:
#                                    #/etc/httpd/conf/httpd2.conf
#1ER BLOQUE
ServerRoot /etc/httpd/2.0
PidFile /var/run/httpd.pid
...
#Implementamos dos Hosts Virtuales Basados en Nombre bajo la IP 192.168.1.1
NameVirtualHost 192.168.1.1
...#Resto de Hosts Virtuales
<VirtualHost 192.168.1.1> #Host Virtual correspondiente a www.ingemartin.es
  ServerName www.ingemartin.es
  DocumentRoot /var/www/html/ingemartin
  #Creamos un alias que indique a apache donde se
  # localizan los scripts CGI a ejecutar:
  ScriptAlias /examen-curso1/ /var/www/html/ingemartin/examenes/curso1/
  Options +execCGI #Directiva con su valor por defecto
  #Acceso restringido al directorio que contiene los scripts de los examenes:
  <Directory /var/www/html/ingemartin/examenes/curso1>
    AuthType Basic
    AuthDBMType DB
    #Mensaje de advertencia al acceder al sitio Web:
    AuthName "Solo Alumnos de Primer Curso"
    AuthDBMUserFile seguridad/alumnos #Guardamos la BD en /etc/httpd/2.0/seguridad
    Require valid-user
  </Directory>
</VirtualHost>
```

Continúa en página siguiente

```
...
#4º BLOQUE
Listen 192.168.1.1:80
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

Asumiremos que se encontrarán registrados en la base de datos todos los alumnos o usuarios que deseamos que se examinen vía Web. En caso contrario será necesario crear la base de datos y rellenarla mediante el uso del comando `htdbm` (repasar apartado 3.9 a partir de la página 122):

```
[root@linux] htdbm -c -TDB /etc/httpd/2.0/seguridad/alumnos julianrodriguez
[root@linux] htdbm -TDB /etc/httpd/2.0/seguridad/alumnos juanjomartin
[root@linux] htdbm -TDB /etc/httpd/2.0/seguridad/alumnos arturomartin
...
[root@linux] htdbm -l -TDB /etc/httpd/2.0/seguridad/alumnos
```

A continuación reiniciaremos el servicio `httpd`:

```
[root@linux]# apachectl restart #service httpd restart o /etc/init.d/httpd restart
```

2. Una vez configurado `apache` crearemos la página Web `examenweb.pl` haciendo uso del lenguaje `perl`. Su contenido será similar al mostrado a continuación:

```
#!/usr/bin/perl -w

print "Content-type: text/html\n\n";

print qq~
<html> <head> <meta http-equiv="Content-Language" content="es"> </head>
<body bgcolor="#FF99FF"> <table border="0"> <tr> <td width="100%"
                                        bgcolor="#00FFFF">
<p align="center"> <font size="3" face="Arial Black">EXAMEN A TRAVES
                                        DE LA RED</font> </p>
<p align="center"> <font size="4" face="Arial Black" color="#FF0000">
                CUESTIONARIO DE ELECTRÓNICA ANALÓGICA </font> </p> </td>
</tr>
~;
open(FICHERO,"</etc/httpd/2.0/seguridad/notas-examen.txt");
@lineas=<FICHERO>;
close(FICHERO);
$usuario_ya_evaluado=0;
foreach $linea (@lineas){
chop($linea);
@usuario_evaluado=split(/:/,$linea);
if ($ENV{'REMOTE_USER'} eq $usuario_evaluado[0]) {
$usuario_ya_evaluado=1; } }
if ($usuario_ya_evaluado == 0) {
print "<tr> <td width='100%'> <p align='center'> <font size='2' face=
                'Arial Black' color='#14250F'>Bienvenido Sr/Sra. \"
                ".$ENV{'REMOTE_USER'}."\" le deseamos suerte con
                el examen ...</font>";
```

Continúa en página siguiente

```
print qq~
</td> </tr> <tr> <td width="100%" colspan="3"> <br>
<form method="POST" action="/examen-curso1/examenweb.pl"> <hr> <p>
                <font face='Arial Narrow'> <b>
~;
open(FICHERO1,"</etc/httpd/2.0/seguridad/cuestiones-examen.txt");
@lineas=<FICHERO1>;
close(FICHERO1);
foreach $linea (@lineas){
@campos=split(/;/,$linea);
print "$campos[1] <br>";
print "<input type='radio' value='$campos[2]' name='$campos[0]'> <font
                color='#008080'>$campos[3] </font> <br>";
print "<input type='radio' value='$campos[4]' name='$campos[0]'> <font
                color='#008080'>$campos[5] </font> <br>";
print "<input type='radio' value='$campos[6]' name='$campos[0]'> <font
                color='#008080'>$campos[7] </font> <hr>";
}
print "...</b> </font> </p> <p align="center"> <input type="submit"
                value="EVALUAR"> </p> </form> </td> </tr> </table>";
$datos_recibidos=<STDIN>;
if ($datos_recibidos ne "") {
#El formato de los datos recibidos es: "pregunta1=R11&pregunta2=R23&..."
@respuestas=split(/&/,$datos_recibidos);
open(FICHERO2,"</etc/httpd/2.0/seguridad/respuestas-examen.txt");
@respuestas_correctas=<FICHERO2>;
close(FICHERO2);
$nota=0;
$num_respuesta=0;
foreach $trozo(@respuestas) {
@respuesta=split(/=/,$trozo);
chop($respuestas_correctas[$num_respuesta]);
if ($respuesta[1] eq $respuestas_correctas[$num_respuesta]) {
$nota=$nota+1; }
$num_respuesta=$num_respuesta+1; }
print "<br>La nota obtenida es $nota.";
open(FICHERO3,">>/etc/httpd/2.0/seguridad/notas-examen.txt");
print FICHERO3 "$ENV{'REMOTE_USER'}:$nota\";
close(FICHERO3); } }
else {
print "</table> <br>Perdona \"$ENV{'REMOTE_USER'}\" pero ya tienes la
                nota puesta: $usuario_evaluado[1]"; }
print "</body> </html>";
```

Trataremos de explicar el script anterior con el mayor detalle posible:

a) Todo `script CGI` debe comenzar informando a `apache` del intérprete que se encargará de leer y ejecutar el script con la finalidad de transformar su resultado en etiquetas HTML y posteriormente enviárselas al cliente Web que llevó a cabo la solicitud Web sobre tal página.

```
#!/usr/bin/perl -w
```

Si en el caso de los `shellscripts` la ruta del intérprete era `#!/bin/bash`, ahora al tratarse de un script codificado en lenguaje `perl` deberemos indicarle la ruta del intérprete de `perl`: `#!/usr/bin/perl`. La opción `-w` informa al intérprete de `perl` que nos advierta de la existencia de errores en el cógido.

b) A continuación, al igual que se hacía con los `scripts CGI` escritos en `shellscript` indicaremos a `apache` y al intérprete que el resultado de su ejecución debe codificarse en lenguaje

HTML. Con ello damos a entender que la salida estándar encargada de mostrar los resultados deja de ser la pantalla, para pasar a ser un intérprete HTML, es decir, un cliente Web como por ejemplo `mozilla firefox`, `Internet Explorer`, `konqueror`, ...

```
print "Content-type: text/html\n\n";
```

c) Tras las dos líneas anteriores empieza el contenido del script. Este comienza imprimiendo el encabezado o título en formato HTML de la página Web. Para ello se hace uso del comando `print`. En el caso de que la cantidad de etiquetas HTML a imprimir ocupe más de una línea se puede hacer uso de varios `print` o de un único `print` utilizando las etiquetas `qq~` y `~;` las cuales informan del conjunto de líneas afectadas por el comando `print`. Ambos casos, equivalentes, se muestran a continuación:

```
print "<html> <head> <meta http-equiv='Content-Language' content='es'> </head>";

print "<body bgcolor='#FF99FF'> <table border='0'> <tr> <td width='100%'
               bgcolor='#00FFFF'>";
print "<p align='center'> <font size='3' face='Arial Black'>EXAMEN A TRAVES
               DE LA RED</font> </p>";

print "<p align='center'> <font size='4' face='Arial Black' color=
               '#FF0000'>CUESTIONARIO DE ";
print "ELECTRONICA ANALOGICA </font> </p> </td> </tr>";
```

o

```
print qq~
<html> <head> <meta http-equiv="Content-Language" content="es"> </head>
<body bgcolor="#FF99FF"> <table border="0"> <tr> <td width="100%"
                bgcolor="#00FFFF">
<p align="center"> <font size="3" face="Arial Black">EXAMEN A TRAVES
                DE LA RED</font> </p>

<p align="center"> <font size="4" face="Arial Black" color="#FF0000">CUES-
TIONARIO DE ELECTRONICA ANALOGICA </font> </p> </td> </tr>
~;
```

La razón de utilizar un formato u otro radica en cuestiones de comodidad.

¡¡Importante!! En el caso de hacer uso de varios `print` debe advertirse que aquellas etiquetas HTML que contengan parámetros que tengan asignados valores encerrados entre comillas dobles deben sustituirse por comillas simples para evitar errores de interpretación en relación a las comillas dobles del propio comando `print`. Por ejemplo:

`print "<p align="center">"` ⇒ `print "<p align='center'>"`

d) Tras el título de la página Web mostraremos el examen (formulario HTML) al usuario que se haya logado, pero antes deberemos comprobar que dicho usuario no ha realizado el examen antes. Para llevar a cabo esta comprobación rastrearemos el fichero `notas-examen.txt` ya que este contendrá a los usuarios que hayan realizado el examen junto con su correspondiente nota.

```
open(FICHERO,"</etc/httpd/2.0/seguridad/notas-examen.txt");
@lineas=<FICHERO>;
close(FICHERO);
$usuario_ya_evaluado=0;
foreach $linea (@lineas){
chop($linea);
@usuario_evaluado=split(/:/,$linea);
if ($ENV{'REMOTE_USER'} eq $usuario_evaluado[0]) {
$usuario_ya_evaluado=1; } }
```

En primer lugar abrimos el fichero en modo lectura (<) apuntando a su contenido a través del manejador de fichero llamado `FICHERO`:

```
open(FICHERO,"</etc/httpd/2.0/seguridad/notas-examen.txt");
```

Después volcamos sobre la variable `@lineas` el contenido del fichero apuntado por el manejador `FICHERO`. Al estar compuesto el fichero por varias líneas no se puede guardar su contenido en una variable normal; por ello que la variable es un vector o array (`@lineas`). Después de esto eliminamos el manejador:

```
@lineas=<FICHERO>;
close(FICHERO);
```

Por último, mediante la ayuda de un bucle `foreach` recorreremos el contenido del fichero (posiciones del vector `@lineas`) para comprobar que el usuario que se ha logado no ha realizado el examen previamente:

```
$usuario_ya_evaluado=0;
foreach $linea (@lineas){
chop($linea);
@usuario_evaluado=split(/:/,$linea);
if ($ENV{'REMOTE_USER'} eq $usuario_evaluado[0]) {
$usuario_ya_evaluado=1; } }
```

En cada uno de los ciclos del bucle `foreach` se va transfiriendo cada una de las líneas del fichero `notas-examen.txt` (@lineas) a la variable `$linea` para que sean, una a una, analizadas.

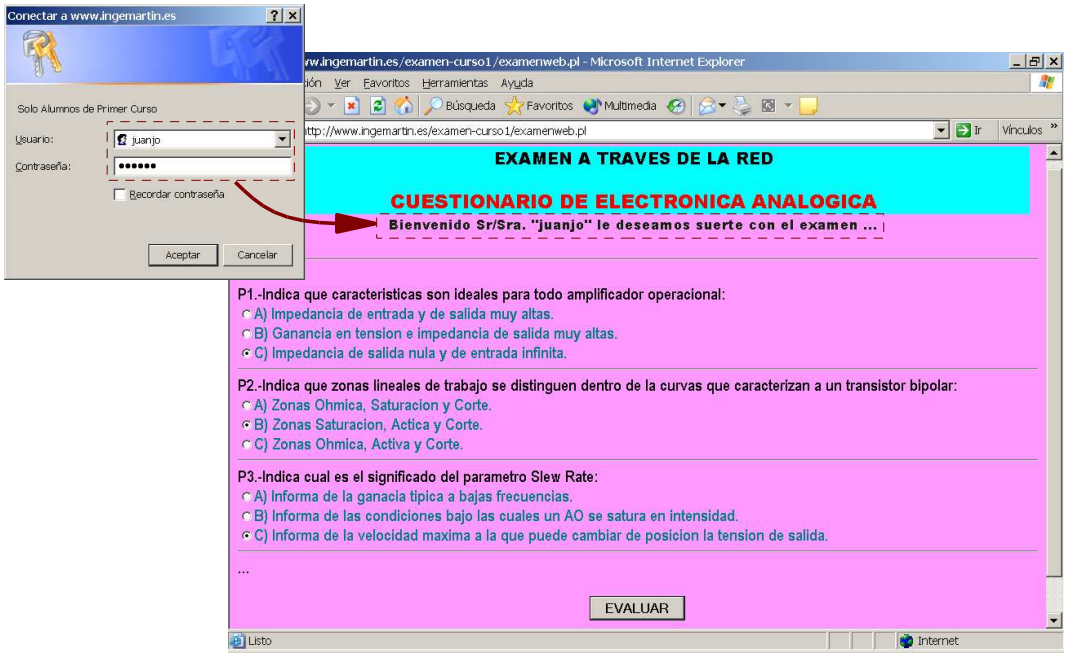
Teniendo en cuenta que cada línea del fichero `notas-examen.txt` esta compuesta por la pareja de datos <`usuario evaluado:nota obtenida`>, mediante el comando `split` separamos ambos campos quedándonos con el primero <`usuario evaluado`> (`$usuario_evaluado[0]`). Posteriormente comprobamos mediante un `if` si coincide con el `login` del usuario que accede al examen vía Web (la variable de entorno CGI `$ENV{'REMOTE_USER'}` contiene el nombre de la persona autentificada durante el login). Mediante una variable de tipo semáforo (`$usuario_ya_evaluado="0"` ó `"1"`) registramos si tal usuario ya se había examinado previamente, o si en cambio, permitimos que lo haga.

En cuanto al comando `chop($linea)` indicar que se encarga de eliminar el salto de línea (`\n`) que se localiza al final de cada una de las filas que se recuperan del fichero. En caso de no hacer uso de él tampoco ocurriría nada, ya que en la comparación entre `$ENV{'REMOTE_USER'}` y `$usuario_evaluado[0]` nunca afecta el carácter final de línea (si tuviésemos que comparar con `$usuario_evaluado[1]` sí que tendríamos problemas), pero es conveniente acostumbrarse a su uso.

e) En el caso de que el usuario que se ha autentificado no haya realizado el examen antes (es decir que `$usuario_ya_evaluado == 0`), pasaremos a mostrarle un mensaje de bienvenida y el formulario del examen a realizar.

```
if ($usuario_ya_evaluado == 0) {
print "<tr> <td width='100%'> <p align='center'> <font size='2' face=
                'Arial Black' color='#14250F'>Bienvenido Sr/Sra. \"
                ".$ENV{'REMOTE_USER'}."\" le deseamos suerte con
                el examen ...</font>";
print qq~
</td> </tr> <tr> <td width="100%" colspan="3"> <br>
<form method="POST" action="/examen-curso1/examenweb.pl"> <hr> <p>
                <font face='Arial Narrow'> <b>
~;
open(FICHERO1,"</etc/httpd/2.0/seguridad/cuestiones-examen.txt");
@lineas=<FICHERO1>;
close(FICHERO1);
foreach $linea (@lineas){
@campos=split(/;/,$linea);
print "$campos[1] <br>";
print "<input type='radio' value='$campos[2]' name='$campos[0]'> <font
                color='#008080'>$campos[3] </font> <br>";
print "<input type='radio' value='$campos[4]' name='$campos[0]'> <font
                color='#008080'>$campos[5] </font> <br>";
print "<input type='radio' value='$campos[6]' name='$campos[0]'> <font
                color='#008080'>$campos[7] </font> <hr>";
}
print "...</b> </font> </p> <p align="center"> <input type="submit"
                value="EVALUAR"> </p> </form> </td> </tr> </table>";
```

En el mensaje de bienvenida es necesario preceder a las comillas dobles de una barra invertida, \, para que estas puedan mostrarse . Las comillas dobles es un carácter significativo en lenguaje `perl`, para que sean tratadas como un carácter más sin ningún significado especial debemos indicárselo al intérprete mediante la \.



El formulario del examen está compuesto por tantos `radiolist` (`type="radio"`), como preguntas hay almacenadas en el fichero `cuestiones-examen.txt`. Por ello, para construir el formulario, abrimos en primer lugar el fichero `cuestiones-examen.txt` en modo lectura (<) y volcamos su contenido sobre la lista o array `@lineas`. Después, mediante un bucle `foreach` por cada una de las líneas formamos un `radiolist`, separando (carácter separador `;`) previamente estas por los distintos campos (`@campos`) que las forman según su sintaxis.

`$campos[0]` ⇒ <numero pregunta>
`$campos[1]` ⇒ <cuestion o pregunta examen>
`$campos[2]` ⇒ <identificador 1ª respuesta>
`$campos[3]` ⇒ <1ª respuesta>
`$campos[4]` ⇒ <identificador 2ª respuesta>
`$campos[5]` ⇒ <2ª respuesta>
`$campos[6]` ⇒ <identificador 3ª respuesta>
`$campos[7]` ⇒ <3ª respuesta>

El uso de los identificadores de respuesta (R11, Respuesta nº1 de la cuestión 1, R12, ...) nos permitirá conocer a posteriori el número de aciertos, y por tanto, la nota final.

f) Una vez mostrado el formulario, el usuario pasará a rellenarlo, y al pulsar sobre el botón `EVALUAR` (`type="submit"`), pasaremos a contabilizar el número de respuestas correctas:

```
$datos_recibidos=<STDIN>;
if ($datos_recibidos ne "") {
#El formato de los datos recibidos es: "pregunta1=R11&pregunta2=R23&..."
@respuestas=split(/&/,$datos_recibidos);
open(FICHERO2,"</etc/httpd/2.0/seguridad/respuestas-examen.txt");
@respuestas_correctas=<FICHERO2>;
close(FICHERO2);
$nota=0;
$num_respuesta=0;
foreach $trozo(@respuestas) {
@respuesta=split(/=/,$trozo);
chop($respuestas_correctas[$num_respuesta]);
if ($respuesta[1] eq $respuestas_correctas[$num_respuesta]) {
$nota=$nota+1; }
$num_respuesta=$num_respuesta+1; }
print "<br>La nota obtenida es $nota.";
```

A través de la variable `$datos_recibidos` recogeremos la información enviada por el formulario (`method="POST" action="/examen-curso1/examenweb.pl"`) al script `examenweb.pl`, al recibirla este por su entrada estándar <STDIN>:

```
$datos_recibidos=<STDIN>;
```

Teniendo en cuenta que el formato en que es recibida la información es: <`name campo1 formulario`>=<`value1`>&<`name campo2 formulario`>=<`value2`>&... (`pregunta1=R11&pregunta2=R23&...`), mediante el comando `split` separaremos la distintas respuestas dadas almacenándolas en el vector o array `@respuestas` para después compararlas con las respuestas correctas `@respuestas_correctas` que se encuentran dentro del fichero `respuestas-examen.txt`.

```
@respuestas=split(/&/,$datos_recibidos);
open(FICHERO2,"</etc/httpd/2.0/seguridad/respuestas-examen.txt");
@respuestas_correctas=<FICHERO2>;
close(FICHERO2);
```

Luego, mediante un bucle `foreach` iremos evaluando cada una de las respuestas proporcionadas comparándolas con las respuestas correctas. En caso de coincidencia incrementaremos un contador, `$nota`, que nos proporcionará la nota final.

```
$nota=0;
$num_respuesta=0;
foreach $trozo(@respuestas) {
@respuesta=split(/=/,$trozo);
chop($respuestas_correctas[$num_respuesta]);
```

Continúa en página siguiente

```
if ($respuesta[1] eq $respuestas_correctas[$num_respuesta]) {
$nota=$nota+1; }
$num_respuesta=$num_respuesta+1; }
print "<br>La nota obtenida es $nota.";
```

Como ya se ha advertido anteriormente, para que la comparación proporcione el resultado esperado, es necesario eliminar el salto de línea (`\n`) mediante el comando `chop` que se recoge en la variable `$respuestas_correctas[$num_respuesta]` al almacenar en ella cada una de las líneas del fichero `respuestas-examenes.txt`. Por este mismo motivo, es necesario asegurarse de que todas las líneas del fichero acaben en un salto de línea, en especial la última.

En cuanto a la condición inicial `if ($datos_recibidos ne "")` tan sólo trata de comprobar si se ha pulsado o no el botón `submit` del formulario, para que solamente se evalúen las respuestas en ese caso, no en caso de recargar la página.

g) Por último, registraremos la nota obtenida por el usuario en el fichero de notas `notas-examen.txt`. Para ello será necesario abrirlo en modo escritura/añadir (>>). Registraremos en nombre del usuario que ha accedido al examen vía Web, `$ENV{'REMOTE_USER'}`, y la nota final, `$nota`, separados ambos por el carácter `":"`. Para garantizar que cada línea de este fichero se corresponda con la nota obtenida por uno de los usuarios, deberemos incluir al final los dos campos anteriores el carácter salto de línea `"\n"`:

```
open(FICHERO3,">>/etc/httpd/2.0/seguridad/notas-examen.txt");
print FICHERO3 "$ENV{'REMOTE_USER'}:$nota\n";
close(FICHERO3); } }
```

h) En el caso de que el usuario que ha accedido ya haya sido examinado previamente, no le mostraremos el formulario del examen, sino un mensaje de advertencia como que el examen ya lo hizo, y cual fue su calificación, información obtenida del fichero `notas-examen.txt`.

```
else {
print "</table> <br>Perdona \"$ENV{'REMOTE_USER'}\" pero ya tienes la
                nota puesta: $usuario_evaluado[1]"; }
print "</body> </html>";
```

3. Tras haber programado el `script CGI examenweb.pl`, para que este pueda ser ejecutado vía Web por `apache` deberemos darle permisos de ejecución y garantizar que el usuario propietario de todo el contenido Web sea él:

```
[root@linux]# chown -R apache.apache /var/www/html
[root@linux]# chmod -R u+x /var/www/html/escarbapedal/programas
```

3.12. Soporte para Webs dinámicas: Scripts PHP

En la actualidad, posiblemente, el lenguaje más utilizado para la implementación de sitios Web dinámicos sea PHP. Este lenguaje es posterior a `perl`, por lo que recoge todo lo bueno de los lenguajes de programación que había hasta el momento tratando de resolver los problemas que presentaban sus antecesores.

A continuación se va a realizar un ejercicio con Webs dinámicas basadas en PHP. Como ya

sucedió con `PERL` es necesario que el lector tenga ligeros conocimientos de PHP. En el caso de no ser así en el apéndice D se ha recogido un resumen de comandos PHP; para hacer el ejercicio propuesto es más que suficiente con que leas este apéndice (página 309).

3.12.1. Instalación del intérprete PHP

Si deseamos que `apache` gestione páginas Web creadas vía PHP, será necesario instalar el módulo `apache2-mod_php` (`urpmi apache2-mod_php`).

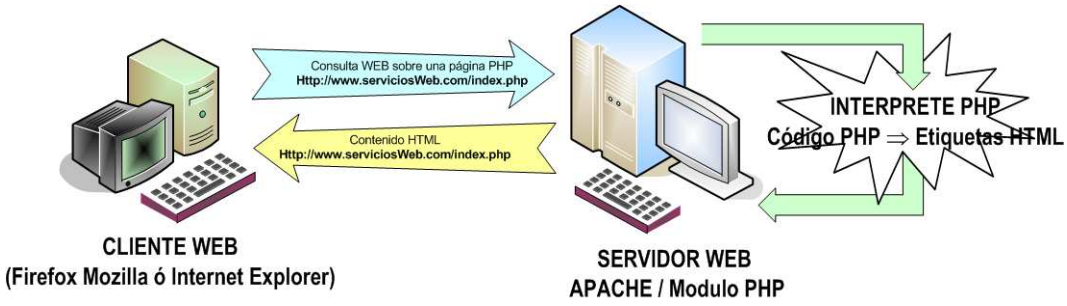


Figura 3.11: Pasos seguidos por apache ante una solicitud de una página PHP.

Una vez instalado el módulo, `apache` se autoconfigura por sí solo, y podremos pasar directamente a probar su funcionamiento. Para ello tan sólo será necesario editar alguna de las páginas Webs que formen parte de alguno de los sitios que cuelguen del servidor `apache`, e incrustar código PHP.

La página cambiada deberá ser renombrada, cambiando su extensión a `.php` (por ejemplo, `index.php`). Esto es muy importante, ya que, la única forma que tiene `apache` de saber que la página web que debe servir, contiene código PHP, es por medio de la extensión dada al archivo.

Una característica importante que presenta PHP en relación a otros lenguajes, como puede ser `perl`, es que el código PHP se encuentra embebido entre el código HTML. Dentro de una página Web (.php) para distinguir entre lo que es código HTML de lo que es código PHP, únicamente habrá que hacer uso de las etiquetas `<?` y `?>`. Lo contenido entre esos símbolos será examinado por el intérprete de PHP.

Un último aspecto importante que presenta PHP, es su magnífica relación con el gestor de bases de datos `MySQL`. Esto permite hacer consultas `SQL` sobre tablas, almacenadas en la base de datos, de una manera muy sencilla.

Ejercicio 3.15

Crear una página Web en PHP llamada `librovisitas.php` relacionada con una asociación (por ejemplo, `amigos del camino de Santiago`) que permita introducir a los visitantes que accedan sus impresiones quedando registradas en un fichero llamado `visitas.txt`.

Solución del ejercicio 3.15

Como en el resto de ejercicios prácticos, a continuación se mostrará una posible solución al problema planteado, y posteriormente se intentará explicar detalladamente.

1. En primer lugar se mostrará el contenido de la página `librovisitas.php` correspondiente a la solución adoptada:

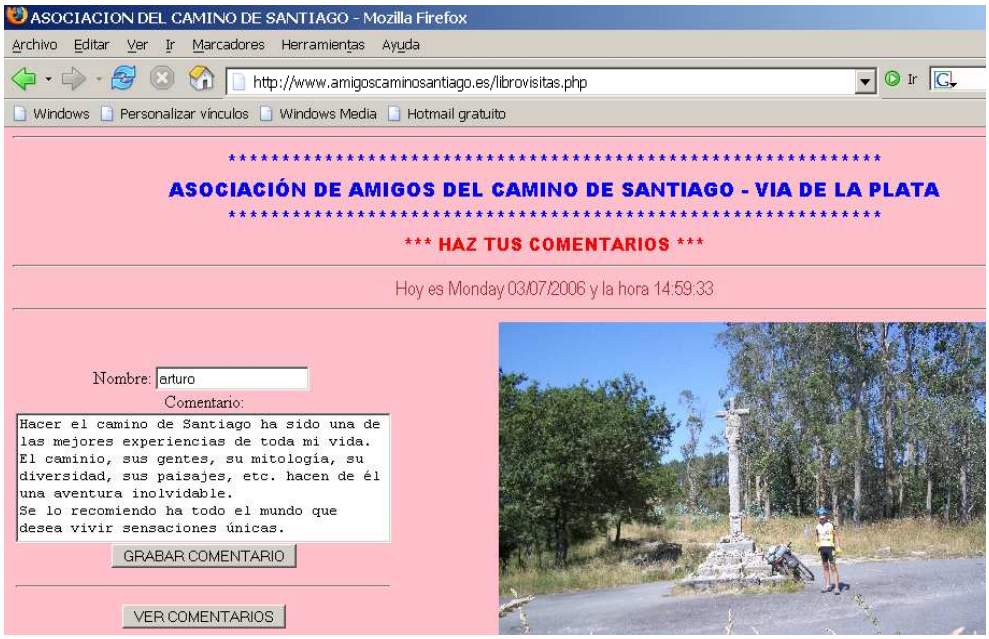


Figura 3.12: Aspecto del sitio Web dinámico propuesto en el ejercicio.

```
<? include "biblioteca.php"; ?>
<html> <head> <title>ASOCIACION DEL CAMINO DE SANTIAGO</title> </head>
<body bgcolor=pink> <hr> <center> <font color=blue face="arial black" size=4>
**********************************************************<BR>
ASOCIACIÓN DE AMIGOS DEL CAMINO DE SANTIAGO - VIA DE LA PLATA<BR>
**********************************************************<BR>
</font> <font color=red face="arial black" size=3>
** HAZ TUS COMENTARIOS ***</font> </center> <hr>
<center> <font color=brown face="arial narrow" size=4>
<? $fechayhora=mostrarfecha();
echo $fechayhora."<br>"; ?>
</font> </center> <hr>
<table border=0 align=center width=100%> <tr> <td> <center>
<form name="f1" action="" method=post>
Nombre: <input type=text name=nombre size=20> <br>
Comentario: <textarea name=comentario cols="40" rows="6"> </textarea> <br>
<input type=hidden name=oculto value="<?solofecha()?>">
<input type=submit name=grabar value="GRABAR COMENTARIO">
</form> <hr>
<form name="f2" action="" method=post>
<p align=center> <input type=submit name=ver value="VER COMENTARIOS"> </p>
</form> </center> </td> <td width=70%> <center>
<img src=finisterra.jpg width=400 height=340> </center> <hr align=center
                                width=30% color=brown>
<? if ($_POST['ver'] == "VER COMENTARIOS")
{ leer("visitas.txt"); }
if ($_POST['grabar'] == "GRABAR COMENTARIO")
{ grabar($_POST['nombre'],$_POST['comentario'],$_POST['oculto'],"visitas.txt");
leer("visitas.txt"); } ?>
<td> </tr> </table> </body> </html>
```

En la primera línea de la solución propuesta se observa que se hace uso de una biblioteca de funciones PHP (`biblioteca.php`). El contenido de la misma puede ser algo como lo mostrado a continuación:

```
<?
function mostrarfecha()
{ $fecha=mktime();
setlocale(LC_TIME,"es_ES");
return(strftime("Hoy es %A %d/%m/%Y y la hora %H:%M:%S",$fecha)); }

function solofecha()
{ $fecha=mktime();
setlocale(LC_TIME,"es_ES");
$fechayhora=strftime("%d/%m/%Y a las %H:%M:%S",$fecha);
echo $fechayhora; }

function leer($fichero)
{ $puntero=fopen($fichero,"r");
$tamaño_fichero=filesize($fichero);
$contenido_fichero=fread($puntero,$tamaño_fichero);
echo $contenido_fichero;
fclose ($puntero); }

function grabar($nombre,$comentario,$fecha,$fichero)
{ $puntero=fopen($fichero,"a");
fwrite($puntero,"<b>$nombre ($fecha) dijo:</b> <br>\n");
fwrite($puntero,"$comentario<br> <hr align=center width=50%color=brown>\n");
fclose($puntero); } ?>
```

2. Comenzaremos explicando las funciones de la biblioteca:

   `mostrarfecha()`: Es una función encargada de devolver (return) la hora en el formato indicado a través de la función `strftime()`.

   ```
   function mostrarfecha()
   { $fecha=mktime();
   setlocale(LC_TIME,"es_ES");
   return(strftime("Hoy es %A %d/%m/%Y y la hora %H:%M:%S",$fecha)); }
   ```

Dentro de esta, la función `mktime()` se encarga de rescatar la fecha y hora del sistema. `setlocale()` las transcribe en español (`es_ES`) y `strftime()` devuelve la hora en el formato especificado.

`solofecha()`: Es una función similar a la anterior, devuelve la hora y fecha del sistema para almacenarla en el libro de visitas. Son posibles otras opciones a las utilizadas en la función `mostrarfecha()`. Por ejemplo, en lugar de hacer uso de la función `return()` en la función `solofecha()` se utiliza directamente la función `echo()`.

```
function solofecha()
{ $fecha=mktime();
setlocale(LC_TIME,"es_ES");
$fechayhora=strftime("%d/%m/%Y a las %H:%M:%S",$fecha);
echo $fechayhora; }
```

`leer($fichero)`: Es la función encargada de leer todo el contenido del fichero pasado como parámetro (`$fichero`) y mostrarlo a continuación por pantalla.

```
function leer($fichero)
{ $puntero=fopen($fichero,"r");
```

Continúa en página siguiente

```
$tamaño_fichero=filesize($fichero);
$contenido_fichero=fread($puntero,$tamaño_fichero);
echo $contenido_fichero;
fclose ($puntero); }
```

Dentro de esta, la función `fopen($fichero,"r")` se encarga de abrir el fichero pasado como parámetro en modo lectura. La función `filesize()` nos informa de su tamaño y `fread()` lee todo su contenido almacenándolo en la variable `$contenido_fichero`. Finalmente mediante la función `echo()` mostramos por pantalla ese contenido y después cerramos el fichero (`fclose`).

`grabar($nombre,$comentario,$fecha,$fichero)`: Es la función encargada de escribir en el fichero pasado como parámetro (`$fichero`), el nombre (`$nombre`) y el comentario (`$comentario`) introducidos a través del formulario HTML junto con la fecha (`$fecha`) de cuándo se hizo.

```
function grabar($nombre,$comentario,$fecha,$fichero)
{ $puntero=fopen($fichero,"a");
fwrite($puntero,"<b>$nombre ($fecha) dijo:</b> <br>\n");
fwrite($puntero,"$comentario<br> <hr align=center width=50%color=brown>\n");
fclose($puntero); }
```

Dentro de esta, la función `fopen($fichero,"a")` abre el fichero en modo escritura colocando el puntero de escritura (`$puntero`) al final de su contenido (la opción `"a"` añade, no sobrescribe) y `fwrite()` escribe en él los valores pasados como parámetros (`$nombre`, `$comentario` y `$fecha`) en formato HTML teniendo en cuenta la forma en que deseamos que posteriormente sea visualizado. El carácter `"\n"` (salto de línea) tan sólo sirve para estructurar el contenido dentro del fichero de texto.

3. A continuación pasaremos a detallar la solución adoptada en `librovisitas.php`.

La página PHP comienza incluyendo la biblioteca de funciones comentada en el punto anterior:

```
<? include "biblioteca.php"; ?>
```

De esta forma, podremos hacer uso de sus funciones simplemente llamando al nombre de la función y pasándole los correspondientes parámetros.

A continuación empieza la página HTML colocando unos títulos junto con la fecha y hora actual. Para obtener la fecha y hora actual, llamamos a la función `mostrarfecha()` de la biblioteca recogiendo su valor devuelto (`return`) en la variable `$fechayhora` para después imprimirla (`echo`).

¡¡Ojo!! Observa que debido a que PHP es un lenguaje embebido en HTML, la única forma de distinguir lo que es PHP de lo que es HTML es mediante la utilización de las etiquetas <? ?>.

```
<html> <head> <title>ASOCIACION DEL CAMINO DE SANTIAGO</title> </head>
<body bgcolor=pink> <hr> <center> <font color=blue face="arial black" size=4>
**********************************************************<BR>
ASOCIACIÓN DE AMIGOS DEL CAMINO DE SANTIAGO - VIA DE LA PLATA<BR>
**********************************************************<BR>
```

Continúa en página siguiente

```
</font> <font color=red face="arial black" size=3>
** HAZ TUS COMENTARIOS ***</font> </center> <hr>
<center> <font color=brown face="arial narrow" size=4>
<? $fechayhora=mostrarfecha();
echo $fechayhora."<br>"; ?>
</font> </center> <hr>
```

Mediante un primer formulario invitamos al usario que accede a la página Web a escribir un identificador y una opinión a cerca de un tema en concreto (en este caso, `El camino de Santiago`).

```
<table border=0 align=center width=100%> <tr> <td> <center>
<form name="f1" action="" method=post>
Nombre: <input type=text name=nombre size=20> <br>
Comentario: <textarea name=comentario cols="40" rows="6"> </textarea> <br>
<input type=hidden name=oculto value="<?solofecha()?>">
<input type=submit name=grabar value="GRABAR COMENTARIO">
</form> <hr>
```

Puede observarse que el formulario consta de un campo oculto (`type=hidden`) con la finalidad de enviar (`submit`) junto con el la información introducida por el usuario la hora del sistema obtenida a través de la función de la biblioteca `solofecha()` (el valor de este campo oculto se encuentra entre comillas dobles, `"<?solofecha()?>"`, ya que la fecha consta de más de una palabra). De esta forma, quedará registrado en el fichero de visitas, tanto la opinión como la fecha en que se realizó.

Toda esta información será enviada a sí mismo tras pulsar el botón `submit` ya que al no colocar nada en el campo `action` del formulario se omite que el destinatario es él mismo (`librovisitas.php`).

Un segundo formulario (consta únicamente de un botón `submit`) nos permite consultar todas las opiniones que han sido registradas hasta el momento. Al igual que el formulario anterior, el destinatario de los datos del formulario es la propia página `librovisitas.php`.

```
<form name="f2" action="" method=post>
<p align=center> <input type=submit name=ver value="VER COMENTARIOS"> </p>
</form> </center> </td> <td width=70%> <center>
<img src=finisterra.jpg width=400 height=340> </center> <hr align=center
width=30% color=brown>
```

Por útimo, mediante código en PHP discriminamos qué botón `submit`, de los dos formularios que hay, fue pulsado. Esto es necesario para saber si hay que registrar (escribir en el fichero `visitas.txt`) la opinión de un visitante, o mostrar todas las opiniones registradas hasta el momento.

```
<? if ($_POST['ver'] == "VER COMENTARIOS")
{ leer("visitas.txt"); }
if ($_POST['grabar'] == "GRABAR COMENTARIO")
{ grabar($_POST['nombre'],$_POST['comentario'],$_POST['oculto'],"visitas.txt");
leer("visitas.txt"); } ?>
```

Para llevar a cabo la discriminación del botón `submit` que fue pulsado, simplemente se comprueba el valor de las variables enviadas a través de los formularios. Debemos recordar que en PHP el valor enviado desde un formulario se recoge mediante variables llamadas `$_POST['<nombre campo formulario>']`.

4. Para probar su funcionamiento será necesario darle permisos de ejecución a la página PHP para el usuario `apache` y después solicitar la página `librovisitas.php` al servidor apache desde nuestro cliente Web (por ejemplo, `Mozilla Firefox`).

```
[root@linux]# chown -R apache.apache /var/www/html
[root@linux]# chmod u+x librovisitas.php
```

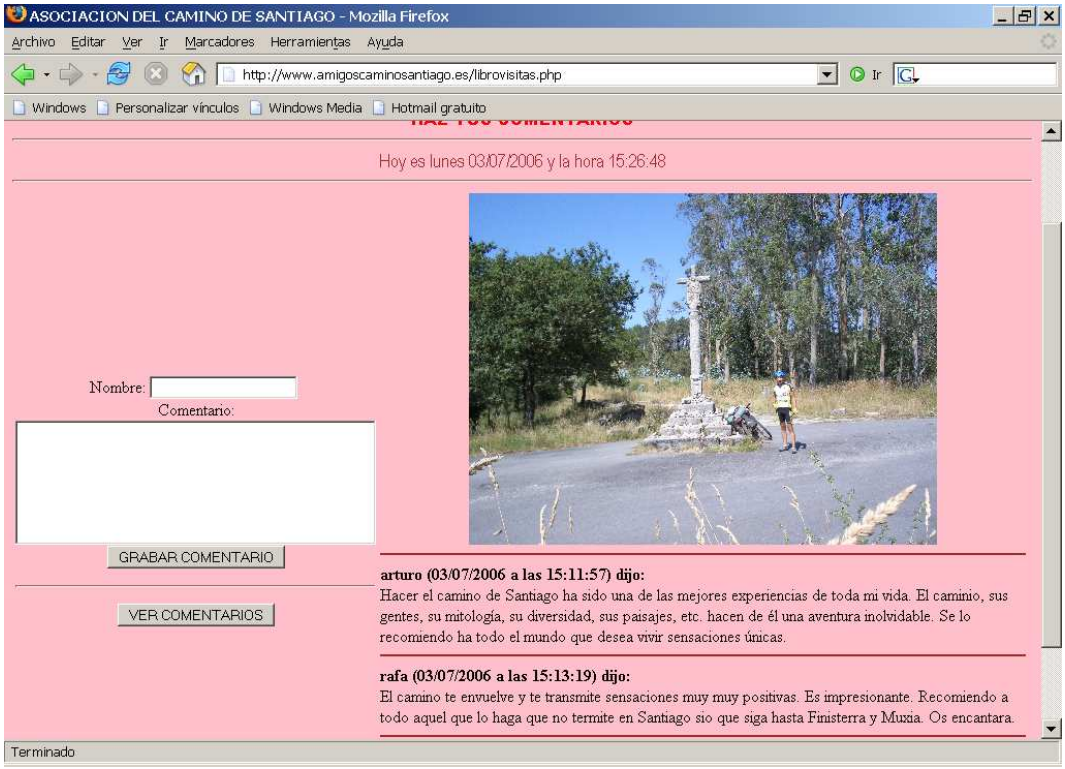


Figura 3.13: Comprobación del sitio Web dinámico escrito en PHP

3.12.2. Gestores de contenido (CMS)

Con la finalidad de facilitar la edición de sitios Webs y su posterior gestión, en la actualidad existen multitud de aplicaciones que permiten desarrollar sorprendentes entornos. Ofrecen la posibilidad de introducir detalles realmente complejos como calendarios, libros de vistias, foros, galerías de imágenes, ... de una manera muy sencilla y sin necesidad de tener conocimientos excesivos de PHP. En general, todo este tipo de aplicaciones reciben el nombre de sistemas gestores de contenido (CMS, `Content Management System`), entre los cuales cabría señalar a `PHP-Nuke`, `Mambo` o los novedosos `Jommla` y `Drupal`.

Independientemente de cuál sea mejor o peor, todos ellos facilitan de tal forma la labor del desarrollador, que en la actualidad carece totalmente de sentido tratar de implementar un sitio Web desde cero, escribiendo línea a línea el código HTML y/o PHP.

Figura 3.14: Los CMS son herramientas software que nos permiten implementar de una manera muy cómoda complejos sitios Web PHP.

3.13. Una posibilidad interesante: XAMMP

`Xampp` es un paquete de software libre (GPL) que permite instalar en nuestro equipo el servidor `apache`, `mysql` (gestor de bases de datos), `phpmyadmin` (herramienta de administración vía Web de `mysql`) y `proftpd` de una manera muy sencilla (4 pasos).

Se dice que `xampp` es un macroservicio que gestiona todo el software mencionado, evitándonos el tener que instalar uno a uno cada uno de los servicios. Además, otra de sus carácterísticas es su portabilidad ya que está disponible tanto para plataformas software Windows, GNU/Linux, ...

Si deseemos probar su funcionamiento, necesitaremos un equipo que no tenga activados ninguno de los servicios para evitar conflictos. Los pasos a seguir son los siguientes:

1. Descargar desde Internet el paquete `xampp-linux.tar.gz`. Puedes descargarlo a través de la página `http://www.apachefriends.org`.

Figura 3.15: Xampp puede descargarse de manera libre desde Internet

2. Siendo el usuario `root` descomprimimos el contenido de `xampp-linux.tar.gz` en `/opt`:

```
[root@linux]# tar xvfz xampp-linux-1.5.3a.tar.gz -C /opt
```

3. Arrancar el servicio `xampp`. Esto activará al mismo tiempo los servicios `apache`, `mysql` y `proftpd`:

```
[root@linux]# /opt/lampp/lampp start
```

Además de arrancar el servicio `xampp`, tenemos otras muchas opciones que convendría ojear ejecutando el comando `lampp` sin parámetros:

```
[root@linux]# /opt/lampp/lampp
```

.

4. Por último, para comprobar que el servicio funciona correctamente, desde un cliente Web realizaremos una solicitud de conexión al servidor `apache`:

```
[user@linux]$ konqueror http://localhost
```

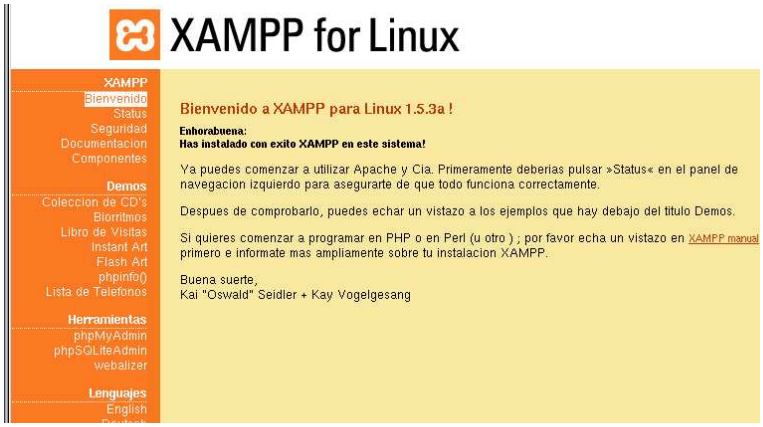


Figura 3.16: Comprobación del funcionamiento de `apache` a través de Xampp.

En el caso de que deseemos personalizar los servicios Web y FTP, deberemos editar los correspondientes ficheros de configuración `httpd.conf` y `proftpd.conf` respectivamente tal como hemos visto en sus capítulos respectivos. Estos los podemos encontrar en:

```
Fichero de configuración de apache: /opt/lampp/etc/httpd.conf
Fichero de configuración de apache: /opt/lampp/etc/proftpd.conf
```

En relación a `mysql`, para administrarlo y gestionarlo haremos uso del sitio Web `phpMyAdmin` que cuelga del servidor `apache`: `http://localhost/phpmyadmin/`.

Figura 3.17: phpMyAdmin nos permite gestionar vía web el DBM MySQL.

Capítulo 4

CONFIGURACIÓN DE UN SERVIDOR FTP: PROFTP

4.1. Introducción al servicio FTP

El servicio FTP[1] es, como bien describen sus siglas, el protocolo normalizado utilizado en la transferencia de ficheros conectados en red. Junto con `TELNET` (control remoto de equipos) y el correo electrónico (e-mail), fue uno de los primeros servicios ofrecidos en Internet que fueron normalizados. En concreto, su primera propuesta se expuso en abril de 1971 mediante el RFC114[2], lo que da cuenta de la larga evolución y continuas modificaciones que ha ido sufriendo reflejándose todo ello en nuevos RFC[3]. Es en febrero de 1973, cuando aparece el RFC454 como resultado de una extensión del RFC254 tratando de solventar todos los problemas conocidos hasta el momento. No obstante, su evolución siguió produciéndose con continuos cambios que se plasmaban en nuevos RFC: 542 (julio 1973), 607 y 614 (1974), 686 y 691(1975), ... Finalmente, es en octubre de 1985 cuando aparece el RFC959 estableciendo un consenso entre todos los aspectos estudiados hasta el momento, y dando una definición del protocolo que se conserva hasta nuestros días.

El conjunto de documentos RFC tan sólo proporcionan una serie de definiciones relacionadas con los aspectos involucrados en una transferencia de ficheros entre equipos conectados en red. Tratan de facilitar la forma en que se comparten los recursos, garantizando una compatibilidad entre los diferentes sistemas de archivos existentes, al mismo tiempo se garantiza una comunicación fiable y eficiente, junto con una protección del resto de los datos que no son compartidos.

En la actualidad la transferencia de ficheros sigue siendo ampliamente utilizada aunque haciendo uso de otra serie de servicios diferentes a FTP. Sólo hace falta observar la cantidad de tráfico de información en formato digital (películas, música, imágenes, ...) que viaja constantemente por Internet relacionada con los novedosos servicios P2P (Peer To Peer): `emule`, `amule`, `direct connect`, ... A pesar de ello, la implementación de los servicios FTP no ha parado, y podemos seguir encontrando multitud de sitios FTP en Internet. Un buen ejemplo de ello son los numerosos repositorios GNU/Linux disponibles en la Red que ofrecen sus paquetes software correspondientes a las distintas distribuciones (por ejemplo, `ftp://ftp.cica.es`).

En un servicio FTP pueden distinguirse dos tipos de equipos: servidores y clientes. Se distinguen entre sí por el software que utilizan. Por ejemplo, un software servidor FTP sería `proftpd`, y un software cliente FTP konqueror. Por un lado, los servidores se encargan de gestionar la información que se comparte desde el equipo, y las solicitudes de conexión procedentes de los clientes, garantizando una comunicación fiable, eficiente y segura. Por otro lado, los clientes son utilizados como intermediarios entre el usuario y el servidor facilitando el establecimiento de la conexión y el posterior control de la transferencia de los datos. En cualquier caso, ambos deben utilizar obli-

[1] Las siglas FTP significan File Transfer Protocol, o en español Protocolo de Transferencia de Ficheros.

[2] Si deseas leer el RFC114, puedes encontrarlo en `http://www.ietf.org/rfc/rfc114.txt`.

[3] Sólo en los primeros años de vida del servicio los RFC 172, 264, 265, 281, 294, 354, 385, 414, o 430 son un ejemplo de los muchos documentos que aparecen incidiendo en aspectos relacionados con el protocolo FTP.

gatoriamente el mismo protocolo para poder comunicarse, ya que en caso contrario no podrían entenderse. El protocolo utilizado es FTP, el cual da nombre al servicio. Este se corresponde con uno de los muchos protocolos que han sido implementados en la capa de aplicación del modelo de referencia TCP/IP.

A diferencia de otros servicios, FTP suele hacer uso tanto del protocolo TCP (Transmission Control Protocol o Protocolo de Control de Transfercia) orientado a conexión, como el UDP (User Datagram Protocol o Protocolo de transmisión de Datagramas de Usuario) no orientado a conexión dentro del nivel 4 o capa de transporte del modelo de referencia TCP/IP para la implementación del servicio. En concreto, suele hacerse uso del protocolo TCP para la transferencia de los datos y el UDP para la transmisión de la información relacionada con el control de la comunicación. El número de puerto TCP/UDP es el 21.

En concreto, los objetivos del protocolo FTP podrían resumirse en los siguientes puntos:

- Facilitar la compartición de archivos en red, facilitando de esta forma el acceso y su posterior transferencia entre equipos remotos.
- Garantizar una comunicación fiable, asegurándonos de que el acceso a la información compartida en el servidor queda acotado a aquellos ficheros que expresamente se haya decidido.
- Ofrecer la posibilidad de restringir el acceso a determinados ficheros que se consideren confidenciales (sitios FTP no anónimos).
- Garantizar al mismo tiempo una comunicación ágil, contribuyendo a aumentar la eficiencia de los enlaces de comunicaciones.

4.2. Tipos de sitios FTP: Anónimos y no anónimos

La clasificación de los sitios FTP, se suele hacer en función del control de acceso que se establece. Según esto, puede hablarse de sitios FTP anónimos y no anónimos:

Sitio FTP anónimo: Se caracteriza por permitir el acceso a los directorios que comparte el servidor de manera anónima sin necesidad de una autentificación previa. Este tipo de configuración se suele adoptar cuando la información compartida en el servidor no tiene carácter confidencial posibilitando un acceso público.

Sitio FTP no anónimo: Al contrario que el anterior se caracteriza por restringir el acceso a su contenido a determinadas cuentas de usuario, requiriendo una autentificación (login y password). Este tipo de configuración suele corresponde cuando los contenidos son de dominio privado, deseando preservar una confidencialidad.

4.3. Objetivos de la práctica del servidor FTP

En este capítulo se cubrirán los siguientes aspectos:

1. Transformar nuestro equipo GNU/Linux en un servidor FTP: instalación de `proftpd`.
2. Configuración de `proftpd` para dar servicio a sitios Web de ámbito público y privado: sitios FTP anónimos y no anónimos.
3. Posibilitar el alojamiento, en un mismo servidor, de distintos sitios FTP: Hosts Virtuales basados en dirección IP y en puerto.

A lo largo de la práctica se propondrán múltiples ejercicios prácticos, proporcionando junto a su enunciado una posible solución. Para la implementación y comprobación de todos estos ejercicios prácticos se asumirá la disposición de una configuración en red con la que poder realizar las pruebas.

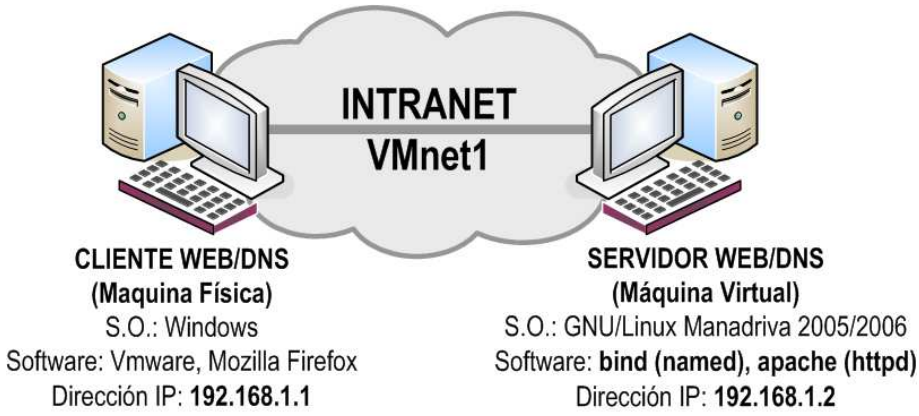


Figura 4.1: Vmware nos ofrece la posibilidad de hacer pruebas en red pudiendo probar las configuraciones realizadas a lo largo de la práctica.

En aquellos casos donde no se disponga de suficiente equipación para ello, cabría recordar que VMware es una muy buena opción en todo tipo de pruebas en red:

Como puede observarse en la figura anterior[4], tras instalar en la máquina VMware basta con configurarlo una de sus redes virtuales (VMnet1) para llevar a cabo las implementaciones propuestas. Será necesario instalar sobre una de sus máquinas virtuales una distribución GNU/Linux (por ejemplo, Mandriva 2005/2006) para configurarla posteriormente como servidor FTP (será necesario configurarlo igualmente como servidor DNS para resolver los nombres de dominio asignados a los sitios FTP).

Por último señalar, que aunque a lo largo de la práctica se tratarán de detallar absolutamente todos los pasos necesarios para su implementación, determinados aspectos, por cuestiones de extensión se darán por supuestos, como son los comandos asociados al sistema operativo GNU/Linux y los aspectos relacionados con los servidores vistos en capítulos anteriores (BIND y `apache`).

[4]Como la situación habitual de los Institutos de Enseñanza Secundaria, que son fundamentalmente a quienes va dirigido este libro, tienen los equipos funcionando con Windows, ha sido asumido que la máquina física funciona bajo Microsoft Windows.

PRÁCTICA: CONFIGURACIÓN DE GNU/LINUX COMO SERVIDOR FTP.

4.4. Instalación del software PROFTPD

Al igual que en anteriores ocasiones, en primer lugar será necesario instalar el software que permita a nuestro equipo GNU/Linux comportarse como un servidor de archivos (`FTP`). Aunque existen multitud de paquetes software que permiten conseguir esto, en la presente práctica se ha elegido `proftpd` (`PROfesional FTP Daemon`), por presentar unas características que a nuestro parecer lo hacen idóneo. Algunas de ellas son:

1. Sintaxis similar a la utilizada en el fichero de configuración de `apache` (`/etc/httpd/conf/httpd2.conf`), donde se distingue entre parámetros[5] y directivas.

2. Permite configurar Host Virtuales basados en IP, y en puerto TCP.

3. Es modular, lo que facilita su configuración, y posterior mantenimiento. Para conocer la lista de módulos contenidos ejecutar

```
[root@linux]# proftpd -l
```

4. Licencia GPL.

5. Alta portabilidad, al funcionar sobre multitud de sistemas operativos: UNIX, Linux, Solaris, FreeBSD, NetBSD, OpenBSD, SunOS, IRIX, HP/UX, Mac OS, ...).

6. Es más seguro que otros programas de similar aplicación.

La instalación es tan sencilla como escribir:

```
[root@linux]# urpmi proftpd
```

4.5. Configuración básica de PROFTPD. Sitio no anónimo

Una vez instalado `proftpd` ya podemos configurarlo. Para ello, y al igual que ocurre con el resto de servicios, `proftp` tiene un archivo de configuación situado en `/etc`, este fichero es `/etc/proftpd.conf`.

Es fácil adivinar que para que un sitio FTP no anónimo pueda ponerse en marcha necesitaremos al menos dos cosas, los usuarios a los que deseamos dar permiso de acceso al sitio FTP y el lugar donde se ubicará la información (el directorio raíz) del sito FTP:

- Los usuarios registrados en el sistema GNU/Linux, por defecto, pueden acceder al sitio FTP. Las cuentas de usuario registradas dentro del sistema se encuentran en `/etc/passwd`, observando el contenido de este fichero podemos conocer los usuarios permitidos.

[5] Mediante el comando

```
[root@linux]# proftpf -D parámetro
```

es posible definir parámetros que posteriormente pueden utilizarse a través de <IfDefine> </IfDefine>.

¡¡Aclaración!! El formato que presenta el fichero `/etc/passwd` es el siguiente:

```
nombre usuario:contraseña:UID:grupo principal:descripción:HOME:shell
```

```
[root@linux]# more /etc/passwd | tail -6
juanjo:x:500:500::/home/juanjo:/bin/bash
ftp:x:76:76:system user for proftpd:/var/ftp:/bin/false
apache:x:77:77:system user for apache-conf:/var/www:/bin/sh
carol:x:505:505::/home/carol:/bin/bash
arturo:x:506:506::/home/arturo:/bin/bash
ntp:x:79:79:system user for ntp:/etc/ntp:/bin/false
```

En el caso de querer restringir los usuarios con permiso de acceso vía FTP a los contenidos en una determinada lista, se deberá hacer uso de la directiva `<Limit LOGIN></Limit>`. Esta directiva, tal como indica su nombre, restringe las cuentas de usuario que pueden *logarse* vía FTP. Por ejemplo, en el caso en que deseásemos que solamente pudiesen acceder al sitio FTP los usuarios registrados `arturo` y `juanjo`, dentro del fichero de configuración de `proftpd`, `/etc/proftpd.conf`, deberíamos editar lo siguiente:

```
<Limit LOGIN>
  #Lista de usuarios a los que se les permite logarse:
  AllowUser arturo juanjo
  #Se Deniega el acceso a todo usuario no especificado en AllowUser
  #(Esto es lo que se denomina establecer una política por defecto)
  DenyAll
</Limit>
```

- En cuanto al directorio raíz del sitio FTP, a diferencia de `Apache`, no es necesario especificar éste[6]. El servicio FTP, por defecto, permite a los usuarios registrados dentro del sistema (los definidos en `/etc/passwd`) acceder a su `HOME` de una manera remota. Es decir, el parámetro `DocumentRoot` de `Apache` se corresponde con el `HOME` del usuario[7] en `proftpd`.

```
[root@linux]# useradd -d home -g grupo principal    nombre usuario
```

Además de `useradd` se puede emplear el alias `adduser` que por ser un vínculo al primero se comporta exactamente igual. Por último, otro comando relacionado con la gestión de los usuarios y que tiene opciones comunes a `useradd` es `usermod`. Este es utilizado para modificar características de usuarios ya creados.

Opciones de los comandos `useradd` y `usermod`

- **Opción -d**

Especifica la ruta del directorio raíz del HOME de la cuenta de usuario a crear, donde tendrá permisos totales sobre su contenido. En caso de no especificar una ruta éste será `/home/`*nombre_usuario*.

[6] Recuerda que en apache esto se hacía mediante el uso del parámetro `DocumentRoot`.
[7] Al crear un usuario se puede especificar el `home` del mismo utilizando la opción `-d`.

- **Opción -s**

Indicamos la `shell` o `intérprete de comandos` que se encargará de ejecutar las instrucciones o comandos especificados por el usuario tanto desde la consola (`LUI/CUI`), como a través de la interfaz gráfica (`GUI`). Los `shell` disponibles se encuentran listados en el archivo `/etc/shell`. En caso de no especificar uno concreto, se asigna por defecto el `bash`.

- **Opción -g**

Permite agregar la cuenta de usuario a crear, dentro de un grupo de usuarios concreto. Por ejemplo, si deseamos que el usuario a crear tenga permisos de administración, lo agregaríamos al grupo de `root` (`-g root`). En caso de no especificar un grupo concreto, el usuario será agregado a un nuevo grupo que se creará con el mismo nombre que el dado al usuario.

- **Opción -G**

Permite añadir la cuenta de usuario especificada a distintos grupos de usuarios ya creados en el sistema además del principal, acumulando los distintos privilegios de cada uno de ellos. Es lo que se denomina pertenecer a grupos secundarios.

En ocasiones puede resultar interesante crear cuentas de usuario que puedan acceder al sistema vía FTP con la finalidad de subir (`upload`) o descargar (`download`) cierta información del servidor, pero garantizando que no puedan iniciar una sesión dentro del sistema.

Debemos tener en cuenta que todo usuario que accede vía FTP debe estar previamente registrado dentro del sistema. Al crear la correspondiente cuenta, y si no se indica lo contrario, estamos concediendo permisos para que puedan iniciar una sesión en el propio sistema y por tanto se está dejando una puerta abierta para que accedan remotamente (`telnet`, `ssh`, `vnc`, ...). Esto puede convertirse en un agujero de seguridad. Para que eso no ocurra, podemos crear una cuenta de usuario para el sitio FTP, pero evitando que pueda logarse dentro del sistema. Para ello, en el momento de la creación, asignaremos al usuario una `shell` falsa (`/bin/false`):

```
[root@linux]# useradd -d DocumenRoot Sitio FTP -s /bin/false nombre usuario
```

Además, para que este tipo de usuarios puedan acceder al sitio FTP será necesario introducir un nuevo parámetro dentro del fichero `/etc/proftpd.conf` (`RequireValidShell on|off`):

```
#Indicamos que los usuarios no requieren una shell operativa
#para poder conectarse al sitio FTP:
RequireValidShell off
```

En el caso en que estemos conformes de que todos los usuarios del sistema puedan acceder (AllowUser) a su `HOME` (DocumentRoot), no será necesario especificar nada en el fichero de configuración `/etc/proftpd.conf` para poner en marcha un servidor FTP no anónimo. Sólo habrá que editarlo en el caso de no estar conformes con ello.

Para indicar nuestro deseo de que el servicio FTP presente o no las opciones por defecto utilizamos `DefaultServer on/off`:

```
# Activamos las opciones por defecto del servidor
DefaultServer on
```

El contenido del fichero `/etc/proftpd.conf` en este comienzo del capítulo será el siguiente (¡elimina el contenido original si fuera necesario!):

```
#Fichero de configuración del servicio proftpd: /etc/proftpd.conf
DefaultServer on
RequireValidShell off #Por defecto vale on
```

Para poder realizar conexiones FTP será necesario poner en marcha el servicio, al igual que con el resto de servicios (o reiniciarlo en el caso de llevar a cabo modificaciones) haciendo uso de uno de los siguientes comandos equivalentes:

```
[root@linux] service proftpd start|restart|stop|status
[root@linux] /etc/init.d/proftpd start|restart|stop|status
```

Ejercicio 4.1

Configurar el servicio FTP de manera no anónima para que tan solo puedan acceder los usuarios `celia` y `julia` vía FTP al sistema.

La raíz del sitio FTP deberá ser `/var/ftp/documentacion`, sobre el cual tendrán permisos tanto de lectura como de escritura. Además, las cuentas de usuario anteriores deberán configurarse para que únicamente puedan acceder al servicio FTP pero en ningún caso iniciar sesión dentro del sistema.

Solución del ejercicio 4.1

En primer lugar debemos asegurarnos de que el directorio raíz del sitio FTP existe. En caso contrario, lo crearemos, y modificaremos su propietario, `UID/GID`. El propietario recomendado es el usuario `ftp`, que tiene asignada una `shell` falsa.

```
[root@linux] mkdir /var/ftp/documentacion
[root@linux] chown ftp.ftp /var/ftp/documentacion
```

A continuación configuraremos/crearemos las cuentas de usuario para cumplir los requerimientos especificados. En caso de que no existan dichas cuentas de usuario las crearemos previamente, para lo cual:

```
[root@linux] useradd -d /var/ftp/documentacion -g ftp -s /bin/false celia
[root@linux] passwd celia
[root@linux] useradd -d /var/ftp/documentacion -g ftp -s /bin/false julia
[root@linux] passwd julia
```

Si las cuentas de usuario ya existieran, lo más sencillo sería modificarlas:

```
[root@linux] usermod -d /var/ftp/documentacion -g ftp -s /bin/false celia
[root@linux] usermod -d /var/ftp/documentacion -g ftp -s /bin/false julia
```

Observa que al hacer que los usuarios `celia` y `julia` pertenezcan al grupo FTP estamos concediéndoles los permisos de grupo sobre el directorio que se les ha asignado. En cualquier caso, para asegurar que esos usuarios tengan permisos tanto de lectura como de escritura sobre el directorio `documentacion`, modificaremos los permisos de grupo:

```
[root@linux]# chmod g+rxw /var/ftp/documentacion
```

Por último configuraremos el fichero `/etc/proftpd.conf` para que las únicas cuentas de usuario que puedan acceder al servicio FTP sean `celia` y `julia`:

```
DefaultServer on
RequireValidShell off
<Limit LOGIN>
  #Lista de usuarios a los que se les permite logarse:
  AllowUser celia julia
  #Se Deniega el acceso a todo usuario no especificado en AllowUser
  DenyAll
</Limit>
```

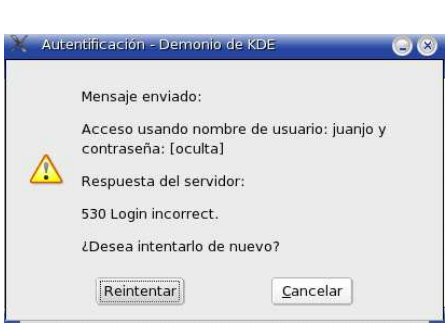
Antes de comprobar la configuración realizada reiniciaremos el servicio `proftpd`:

```
[root@linux]# service proftpd restart
```

Nota: Aunque al iniciar el servicio se realiza de forma automática, puede testearse el contenido del `proftpd.conf` por medio del comando

```
[root@linux]# proftpd -t
```

Seguidamente, desde un equipo (Linux/Windows) ejecuta tu aplicación cliente FTP preferida, por ejemplo `konqueror`, `Mozilla Firefox`, `Internet Explorer`, ...Después invoca al servicio anterior: `URL→ ftp://`*`direccionIP servidor FTP`*. Podrá comprobarse que en caso de tratar de acceder con una cuenta de usuario diferente a las indicadas, será denegado el acceso, y que éste solo será satisfactorio en caso de que se trate de `celia` o `julia`.



Comprobación del ejercicio:

Una vez dentro del sitio FTP, es interesante comprobar el buen funcionamiento de la configuación realizada. Para ello realiza las siguientes actuaciones:

1. Trata de descargar un documento y crear una carpeta dentro del sitio FTP, lo que os permitirá comprobar los permisos de lectura/escritura concedidos.

   Si se anularan los permisos de escritura, podría comprobarse que ya no es posible escribir en el sitio FTP:

   ```
   [root@linux]# chmod g-w /var/ftp/documentacion
   ```

2. En caso de no colocar el parámetro `RequireValidShell` a `off` (por defecto, `on`), comprueba que las cuentas de usuario correspondientes a `celia` y `julia` no pueden acceder vía FTP.

Figura 4.2: Comprobación de los permisos tras crear una carpeta.

```
RequireValidShell on
```

3. Así mismo, si prescindiéramos de la directiva `<Limit LOGIN></Limit>` podría acceder al sistema vía FTP cualquier usuario que estuviera registrado en `/etc/passwd`.

4.6. Configuración avanzada de PROFTPD. Sitio FTP no anónimo

Como se ha podido comprobar es muy sencillo poner en marcha un sitio FTP. Con un pequeño número de parámetros el servicio FTP responde a la perfección. No obstante, como vamos a poder comprobar a continuación, existen una serie de inconvenientes que no se han tenido en cuenta en la configuración anterior.

4.6.1. Evitando la navegación por el sistema de ficheros.

En el ejercicio práctico del apartado anterior se puede observar, que una vez dentro del servidor, el usuario puede navegar por todo el sistema de ficheros de GNU/Linux. Así, después de logarse vía FTP, el usario puede rastrear todo el sistema de ficheros leyendo información que puede ser confidencial, o realizar otra serie de acciones indeseadas.

A priori, esto no debería conllevar ningún tipo de riesgo de seguridad. Los usuarios que acceden vía FTP suelen ser usuarios del sistema y la conexión vía FTP permite, al igual que cuando se inicia sesión en sistema, navegar libremente por el sistema de ficheros de GNU/Linux según los permisos de este.

Figura 4.3: Muestra de que los pulsadores de navegación están activos (`konqueror`).

En cualquier caso, puede interesarnos que esto no sea así, y que una vez que hemos accedido al sistema a través de una conexión, los usuarios no puedan acceder más arriba de la raíz del sitio FTP dentro del árbol de directorios de GNU/Linux. Es decir, si por ejemplo la raíz (DocumentRoot) es `/var/ftp/documentacion`, permitiríamos acceder a sus subdirectorios, pero en ningún caso a su directorio padre, `/var/ftp/`, y superiores.

Para garantizar esto se debe hacer uso de un nuevo parámetro dentro de `/etc/proftpd.conf` denominado `DefaultRoot`, el cual restringe al acceso al sitio FTP al directorio especificado y subdirectorios.

```
#Raíz del sitio FTP, evita navegar por encima de él
DefaultRoot /var/ftp/documentacion
```

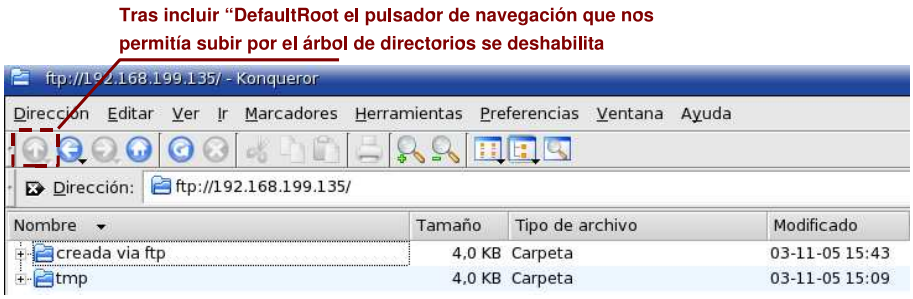


Figura 4.4: Comprobación de que los pulsadores de navegación se desactivan.

Por defecto su valor es `DefaultRoot /`, es decir, la raíz del propio sistema de ficheros de GNU/Linux, de ahí que si no especificamos otra cosa, podamos navegar por todo él.

No obstante esto puede crearnos un conflicto, ya que no todos los usuarios que tienen permiso de acceso al sitio FTP tienen porque tener la misma raíz (HOME). Para evitar esta confusión existe un valor comodín, ~ (código ASCII 126, AltGr+4 ó AltGr+ñ), que hace referencia al HOME de cada usuario.

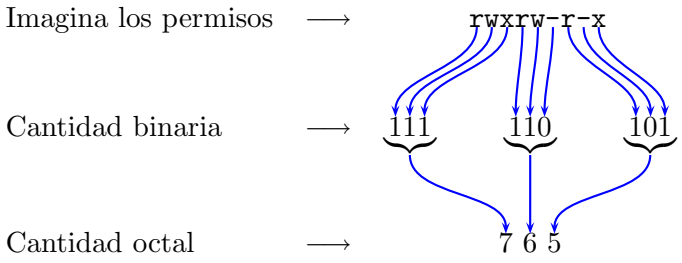
```
#La Raíz del sitio FTP será el HOME de cada usuario
DefaultRoot ~
```

4.6.2. Permisos de ficheros y directorios creados vía FTP

Para poder modificar los permisos con los que son creados los directorios y archivos dentro de GNU/Linux surge el concepto de mascara. Por ejemplo, cuando un usuario crea un directorio dentro del sistema lo hace con los permisos, `rwxr-xr-x`, y en el caso de que sea un fichero con `rw-r--r--`. Esto es así porque el comando `umask` está configurado en nuestro GNU/Linux de esta forma por defecto. Este comando es quien dictamina con qué permisos se crean los archivos y directorios.

A `umask` se le asigna una cantidad octal, que está relacionada con los permisos de un archivo o directorio. Los permisos de archivos y directorios también se pueden expresar por medio de una cantidad octal, tal y como se muestra en el siguiente ejemplo.

Ejemplo:



Los permisos resultantes con los que se crea un directorio o un fichero son el resultado de realizar una `AND` (producto lógico) entre los permisos 777 (⇒ `rwxrwxrwx`) para directorios y 666 (⇒ `rw-rw-rw-`) para ficheros con el *complemento*[8] *a 1* de la mascara definida con `umask` en formato octal.

Inicialmente, `umask` tiene asignado el valor 022. Esto significa que los permisos con los que se crearán los directorios serán:

[8] El complemento a 1 de una cantidad lógica consiste en invertir los valores lógicos. Es decir, los `0` pasan a ser `1` y los `1` se convierten en `0`

$$777 \ \& \ \overline{022} = 777 \ \& \ 755 = 755 \quad \Longrightarrow \quad \texttt{rwxr-xr-x}$$

Y los permisos con que serán creados los archivos:

$$666 \ \& \ \overline{022} = 666 \ \& \ 755 = 644 \quad \Longrightarrow \quad \texttt{rw-r--r--}$$

En el caso de que la creación se lleve a cabo vía FTP también existe una máscara, llamada en este caso `Umask` y que por defecto también vale `022` tanto para directorios como para ficheros (sintaxis: `Umask` *`máscara ficheros  máscara directorios`*). Esto da como resultado unos permisos de creación similares a los expuestos anteriormente (755 ó `rwxr-xr-x` para directorios y 644 ó `rw-r--r--` para ficheros).

En la figura 4.5 se muestra un ejemplo de comprobación. Observa los permisos que se otorgan a un directorio y a un fichero que hayan sido creados vía FTP.

Figura 4.5: Permisos de escritura, lectura y ejecución por defecto.

Ejemplo: Imagina que deseamos unos permisos de creación para directorios de la forma `rwxrw----` y para archivos `rw-------`. La máscara debería ser modificada introduciendo el parámetro `Umask` en el fichero de configuración `/etc/proftpd.conf`:

```
#Permisos resultantes: 666&077=600 y 777&017=760
Umask 077 017 #Ficheros: 077, Directorios: 017
```

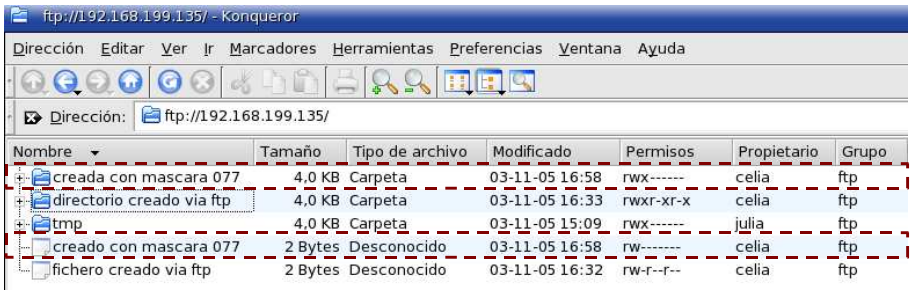


Figura 4.6: Modificación en los permisos de creación tras utilizar `Umask`.

4.6.3. Control de las conexiones al servicio FTP

En este apartado va a mostrarse cómo se puede limitar el número de conexiones permitidas a nuestro servicio FTP y cómo restringir el acceso a dicho servicio desde máquinas concretas.

En ocasiones puede resultar interesante limitar el número de conexiones FTP a nuestro servidor para evitar sobrecargas. Para ello disponemos de un conjunto de parámetros que al agregarlos en el fichero de configuración del servicio `proftpd /etc/proftpd.conf` nos permiten evitarlo. De entre todos los parámetros caben destacar `MaxIstances` y `MaxClients`, que tienen un significado general común, pero con diferentes connotaciones.

Mientras que `MaxIstances` limita el número de conexiones al servicio FTP sin objeciones, `MaxClients` limita el número máximo de conexiones diferenciando entre el número de accesos desde un equipo y/o el número de conexiones realizadas por un usuario.

Parámetros de configuración `MaxIstances` y `MaxClients`

- **MaxInstances** ***cantidad***

Establece el número máximo de conexiones que van a ser aceptadas por el servidor FTP.
Ejemplo:

```
MaxInstances 34
```

- **MaxClients** ***cantidad*** ***mensajes***

Tiene un significado similar a `MaxInstances`, pero suele ir acompañado de los parámetros `MaxClientsPerHost` y `MaxClientsPerUser` que nos permiten una mayor personalización en los límites.

De manera opcional, podemos indicar el mensaje que mostrará el cliente FTP en el momento del acceso si se supera dicho límite. Si nosotros no proporcionamos este mensaje `proftp` dará el mensaje que tiene configurado por defecto.

MaxClientsPerHost *cantidad* *mensaje*
Evita que desde un mismo equipo puedan establecerse una cantidad excesiva de conexiones FTP, limitándolo al valor especificado. De manera opcional, podemos indicar el mensaje que mostrará el cliente FTP en el caso de superar dicho límite.
Ejemplo:

```
MaxClientsPerHost 3 "Número maximo de conexiones superado"
```

MaxClientsPerUser *cantidad* *mensaje*
Evita que una misma cuenta de usuario `ftp` pueda establecer más de una cierta cantidad de conexiones a la vez.
Ejemplo:

```
MaxClientsPerUser 2 "Ya has superado tu número máximo de conexiones"
```

Figura 4.7: Mensaje que aparece al superar el límite de conexiones.

Según esto y para poder controlar el número de conexiones, dentro de `/etc/proftpd.conf` haremos uso del parámetro `MaxInstances` o de los parámetros `MaxClients/MaxClientsPerHost/`

`MaxClientsPerUser`, pero carecería de sentido usar ambos. Por ejemplo, si nuestra única pretensión fuera simplemente limitar el número de conexiones a nuestro servicio FTP el fichero `/etc/proftpd.conf` quedaría de la siguiente forma:

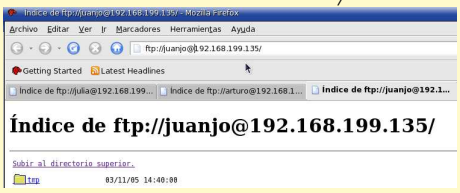
```
#Fichero de configuración del servicio proftpd: /etc/proftpd.conf
DefaultServer       on
RequireValidShell   off
DefaultRoot         ~
#Evita que puedan establecerse más de 10 conexiones al servicio FTP
MaxInstances        10
```

Si se hubiera deseado establecer un criterio limitador más específico se podría haber configurado como sigue:

```
#Fichero de configuración del servicio proftpd: /etc/proftpd.conf
DefaultServer       on
RequireValidShell   off
DefaultRoot         ~
#Evita que puedan establecerse más de 10 conexiones al servicio FTP:
MaxClients          10
#Garantiza que un cuenta de usuario solo establezca una única conexión:
MaxClientsPerUser   1
#Evita que desde un mismo equipo se realicen más de 3 conexiones
MaxClientsPerHost   3
```

¡¡Nota!! Para poder hacer las comprobaciones pertinentes sobre la correcta configuración de los límites `MaxInstances/MaxClients`, podemos hacer uso de clientes FTP como por ejemplo `mozilla-firefox`. Para establecer una conexión FTP sobre un servidor desde `mozilla-firefox` será necesario utilizar la siguiente sintaxis en la URL:

URL ⟶ ftp://*usuario*@*direccionIPservidor*/*NombreDominioServidor*



Para conocer el número de conexiones FTP que esta atendiendo nuestro servidor puede utilizarse el comando `ftpwho`, el cual nos informará de qué usuarios son, y el tiempo que llevan conectados.

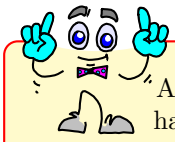
```
[root@linux]# ftpwho
standalone FTP daemon [8933], up for 0 min
  9033 julia [0m15s] 0m11s idle
  9084 celia [0m3s] 0m3s idle
Service class              2 users
```

Limitar el acceso a determinadas máquinas

De forma similar a como ocurría con el servidor `apache`, en `proftp` pueden evitarse conexiones indeseadas al servidor desde equipos concretos, utilizando una sintaxis común: `Order/Deny/Allow`.

Se debe hacer uso de la directiva <Limit LOGIN> </Limit> a través de la cual indicaremos qué equipos están autorizados a conectarse al servicio FTP y cuáles no. Por defecto y si no se indica lo contrario, la política es permitir conexiones desde cualquier equipo cliente.

```
#Directiva encargada de limitar quien y desde donde pueden logarse vía FTP:
<Limit LOGIN>
  #Indicamos que primero mire quienes no están autorizados:
  Order deny,allow
  #Lista de equipos no autorizados:
  Deny from direccionIP/NombreDominio
  #Autorizamos accesos si el cliente no se encuentre en Deny:
  Allow from all
</Limit>
```



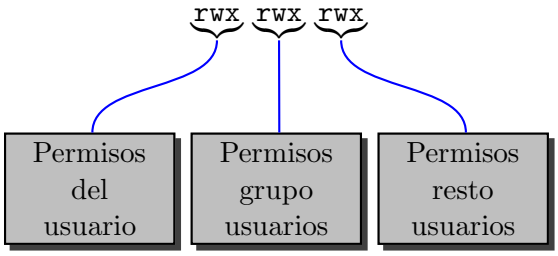
Al igual que cuando se configura `apache`, al especificar los equipos se puede hacer uso de:
La dirección IP del equipo, de la red de equipos o del nombre de dominio del equipo o del dominio que engloba a todos los equipos que le pertenecen.

4.6.4. Permisos de lectura/escritura en el sito FTP

Controlar los permisos de lectura/escritura sobre el sitio FTP puede ser tan sencillo, como lo es configurar adecuadamente los permisos del propio sistema de ficheros de GNU/Linux. Por tanto, tal como se ha hecho en el ejercicio 4.1, antes de poner en marcha el servicio `proftpd` será necesario configurar los permisos de los directorios del sitio FTP mediante el uso de `chmod`. Si al listar la información de los directorios que forman el sitio FTP (comando `ls -l` *`directorio padre`*, los permisos que aparecen no concuerdan con los deseados, será necesario modificarlos.

Por ejemplo, si sobre un directorio queremos conceder al usuario propietario (`u`) del directorio todos los permisos, excepto el de ejecución, al grupo (`g`) asignado al directorio todos los permisos excepto el de escritura, y al resto de usuarios ningún tipo de permisos ejecutaríamos[9]:

```
[root@linux]# chmod u+rw-x,g+r-wx,o-rwx /var/ftp/sitioftp
```



No obstante existe otro modo de controlar estos permisos desde el propio servicio `proftpd`: hacer uso de la directiva `<Limit READ WRITE STOR></Limit>` (`READ` ⇒ `lectura`, `WRITE` ⇒ `modificar`, `STOR/STORe` ⇒ `almacenar/permitir transferencia`).

De esta forma, podemos controlar los permisos sobre el conjunto del sitio FTP haciendo uso de una sintaxis similar a la utilizada antes para controlar las cuentas de usuario autorizadas (`<Limit`

[9] Observa que el comando mostrado equivale a

```
[root@linux]# chmod 650 /var/ftp/sitioftp
```

LOGIN></Limit>). También es posible acotar su efecto limitador a un directorio específico mediante el uso de la directiva <Directory *ruta/path*></Directory>.

¡Importante! Por defecto, si no indicamos lo contrario, no se establecerá ningún tipo de límite, y se tendrán en cuenta únicamente los permisos concedidos sobre el sistema de ficheros (chmod). Pero en caso de indicar límites dentro de /etc/proftpd.conf, se establecerán como permisos resultantes aquellos que sean más restrictivos entre los indicados mediante la directiva Limit y los concedidos a través del sistema de ficheros.

Ejercicio 4.2

Edita el fichero /etc/proftpd.conf y configura el servicio proftpd de manera no anónima para que tan sólo puedan acceder al servidor FTP los usuarios manolo, cecilia y pedro pertenecientes al grupo depdesarrollo. Debe restringirse el acceso al servidor para que sólo pueda realizarse desde los equipos pertenecientes a una red interna 192.168.100.0/255.255.255.0. El número de conexiones deberá limitarse a 5, garantizando que desde un equipo tan sólo se pueda establecer una única conexión de manera simultánea. Podrán realizarse dos conexiones por cuenta de usuario al mismo tiempo, pero evidentemente, desde equipos diferentes.

La raíz del sitio FTP para cada uno de los usuarios autorizados deberá ser /var/ftp/manolodepdes, /var/ftp/ceciliadepdes, y /var/ftp/pedrodepdes respectivamente, pudiendo navegar únicamente por sus subdirectorios, sobre los cuales tendrán permisos tanto de lectura como de escritura. Podrán crear directorios y ficheros con permisos rwxrwx--- y rw-rw---- respectivamente. No obstante dentro de cada uno de los sitios FTP deberá existir un subdirectorio llamado sololectura sobre el que tendrán exclusivamente permisos de sólo lectura.

Además, deberán configurarse correctamente las cuentas de usuario anteriores para que puedan únicamente acceder al equipo servidor vía FTP, pero en ningún caso, iniciar sesión dentro de éste.

Configurar una zone en BIND para que el acceso al servidor sea a través del nombre de dominio ftp.tusitioftp$_x$.com (la x se corresponde con el número del equipo desde el que realizas la práctica).

Solución del ejercicio 4.2

Comenzaremos creando los directorios y cuentas de usuario en el servidor:

```
[root@linux]  mkdir -p /var/ftp/manolodepdes/sololectura
              /var/ftp/ceciliadepdes/sololectura /var/ftp/pedrodepdes/sololectura
```

Con la opción -p crea al mismo tiempo los directorios del departamento de desarrollo de cada usuario y sus correspondientes subdirectorios sololectura.

```
[root@linux]# chown -R ftp.ftp /var/ftp
```

De manera recursiva (-R) modificamos el *propietario.grupo* (ftp.ftp) de todo el contenido que cuelga (subdirectorios/archivos) de /var/ftp

```
[root@linux]# chmod -R u+rwx,g+rwx,o-rwx /var/ftp/*depdes
```

Concedemos todos los permisos sobre los directorios anteriores al usuario ftp y al grupo de usuarios ftp, y ninguno al resto. Advertir que al grupo de usuarios ftp al que pertenecerán

nuestros usuarios (opcional) también se les ha concedido permisos de escritura, por lo que los acotaremos con `Limit`.

Creamos el grupo de cuentas del supuesto departamento de desarrollo y los correspondientes usuarios. La opción `-s /bin/false` garantizará que no pueden iniciar sesión en el sistema directamente, vía `telnet`, `VNC` u otra forma de control que no sea vía FTP:

```
[root@linux] groupadd depdesarrollo
[root@linux] useradd -d /var/ftp/manolodepdes -g ftp -G depdesarrollo
             -s /bin/false manolo
[root@linux] useradd -d /var/ftp/ceciliadepdes -g ftp -G depdesarrollo
             -s /bin/false cecilia
[root@linux] useradd -d /var/ftp/pedrodepdes -g ftp -G depdesarrollo
             -s /bin/false pedro
[root@linux] passwd manolo
[root@linux] passwd cecilia
[root@linux] passwd pedro
```

A continuación editamos el fichero `/etc/proftpd.conf` para configurar el servicio FTP dado por `proftpd` y así cumplir los requerimientos impuestos en el enunciado de la práctica (¡sobreescribe si es necesario su contenido!):

```
#Fichero de configuración del servicio proftpd: /etc/proftpd.conf
DefaultServer on
RequireValidShell off #Por defecto vale on
DefaultRoot ~ #Evitamos la navegación fuera del HOME
MaxClients 5 "No se admiten más conexiones FTP: max=5"
MaxClientsPerHost 1 "Ya tienes abierta una conexion al servidor, lo siento"
MaxClientsPerUser 2 "Esta cuenta ya esta siendo utilizada por otros 2 clientes"
#Los directorios/archivos creados por usuarios vía FTP tendrán los
#permisos rwxrwx--- y rw-rw----, su máscara será el complemento a 1
Umask 007 006
<Limit LOGIN> #Limita que usuarios pueden acceder al servidor FTP
  #Autorizamos el acceso al servidor FTP a los usuarios pertenecientes
  #al grupo de cuentas de usuarios depdesarrollo
  AllowGroup depdesarrollo
  DenyAll #Denegamos el acceso a todo usuario no especificado en AllowGroup
</Limit>
<Limit LOGIN>
  #Primero mira qué equipos están autorizados, y después los denegados
  order allow,deny
  #Autorizamos el acceso a todos los equipos con dirección IP perteneciente
  #a la red 192.168.100.0/255.255.255.0 (24 indica el número de
  #unos de la máscara):
  allow from 192.168.100.0/24
  deny from all #Denegamos el acceso a cualquier otro equipo
</Limit>
#Todo lo que encierra la directiva Directory afectará únicamente
#al directorio indicado
<Directory /var/ftp/manolodepdes/sololectura>
   <Limit WRITE>
     DenyAll
   </Limit>
</Directory>
<Directory /var/ftp/ceciliadepdes/sololectura>
```

Continúa en página siguiente

```
    <Limit WRITE>
      DenyAll
    </Limit>
  </Directory>
  <Directory /var/ftp/pedrodepdes/sololectura>
    <Limit WRITE>
      DenyAll
    </Limit>
  </Directory>
```

Por último, sólo quedaría reiniciar el servicio,

```
[root@linux]# service proftpd restart
```

Aclaremos algunos aspectos de la solución anterior:

1. Si hubiésemos querido autorizar el acceso a más de un grupo de usuarios, sería necesario añadir el parámetro `AllowGroup` tantas veces como grupos sean. Es decir, si además del grupo de usuarios `depdesarrollo` hubiéramos querido autorizar el acceso al grupo `depinvestigacion` no sería correcto escribir lo siguiente:
   Nota: Decimos esto porque con `AllowUser` sí era posible utilizar esa sintaxis para varios usuarios

```
  <Limit LOGIN>
    AllowGroup depdesarrollo depinvestigacion
    DenyAll
  </Limit>
```

   Para que la configuración sea operativa, deberemos hacer lo siguiente:

```
  <Limit LOGIN>
    AllowGroup depdesarrollo
    AllowGroup depinvestigacion
    DenyAll
  </Limit>
```

2. A la hora de limitar los permisos a sólo lectura sobre los directorios `sololectura` de todos los usuarios, en lugar de escribir tres veces la directiva `Directory` podría pensarse en utilizar el carácter comodín `*` (asterisco), ampliamente usado en GNU/Linux, para así tan sólo escribirlo una vez, pero no funcionaría:

```
  <Directory /var/ftp/*depdes/sololectura>
    <Limit WRITE>
      DenyAll
    </Limit>
  </Directory>
```

El uso de los comodines sólo es posible en el último directorio de la ruta especificada. Esto significa que, en nuestro ejemplo, los comodines sólo podrían ser usados en el directorio `sololectura`. En este caso esto no sirve de nada, pero podría ayudarnos en futuras situaciones.

4.6.5. Control del tiempo de conexión al servicio FTP

Con el fin de evitar problemas relacionados con la seguridad y minimizar el consumo de recursos por establecimiento de conexiones en el servidor, `proftpd` nos permite configurar una serie tiempos; transcurridos los cuales se produce la desconexión automática de sesión FTP.

Parámetros de configuración de tiempos de conexión

- **TimeoutIdle** ***segundos***

Indica el tiempo máximo de inactividad (logarse, explorar, transferir, . . .) por parte del usuario, tras el cual se producirá la desconexión automática del servidor. Se contabiliza desde el mismo momento en que el servidor acepta la petición de conexión, y nos pide que nos loguemos.

Garantiza que no se desaprovechen recursos en el servidor por mantener una conexión activa no utilizada. También evita un posible uso mal intencionado por parte de otro usuario diferente al que inició la sesión, en caso de que este último olvidara cerrarla. Por defecto vale 600 segundos.

- **TimeoutStalled** ***segundos***

Tiempo máximo que se mantendría la conexión activa en caso de producirse un atasco o bloqueo de la transferencia. Por defecto vale 3600 segundos. Si lo colocamos a 0 segundos, la espera al posible desatasco sería indefinida.

- **TimeoutNoTransfer** ***segundos***

Tiempo máximo tras el cual se perderá la conexión en caso de no producirse ninguna transferencia de información (datos que no sean de control). Por defecto está en 300 segundos.

4.6.6. Auditoría del servicio PROFTPD

La auditoria nos permite reflejar en un documento quién, qué y cuándo llevó a cabo acciones sobre el servidor FTP. En concreto podemos auditar qué usuarios establecieron conexión con el servicio `proftpd` y en que momento, si se llevaron a cabo operaciones de lectura, de escritura, de almacenamiento, . . .

En caso de no indicar lo contrario, esta información se registra por defecto en un `log` del sistema `/var/log/xferlog`. Es posible modificar la ubicación y nombre de este fichero, así como forzar que sólo se registren ciertas acciones con el formato deseado.

Parámetros de configuración para la auditoría

- **TransferLog** ***fichero***

Fichero donde se almacena la información de auditoría. Recoge las transferencias realizadas entre clientes/servidor FTP bajo un formato específico que posteriormente describiremos. El formato utilizado sigue el estilo clásico de servidores FTP anteriores a `proftpd`. Si no se indica lo contrario el fichero de salida será `/var/log/xferlog`.

Si por el contrario, no nos interesa la información suministrada a través de `TransferLog` lo indicaremos de la siguiente forma:

```
TransferLog NONE
```

- **LogFormat *nickname formato***

Permite personalizar el formato de la información que se registra en los ficheros de auditoria. En el caso de que la información aportada a través de `TransferLog` sea excesiva o insuficiente, `LogFormat` nos permite definir con exactitud la información que deseamos auditar.

Se utiliza conjuntamente con el parámetro de configuración `ExtendedLog` ya que el formato utilizado por `TransferLog` no es configurable. El formato creado se referencia por medio de un identificador (*nickname*).

El valor por defecto del formato es:

```
LogFormat "%h%l%u%t \"%r \"%s%b"
```

Observa que las comillas dobles son un carácter reservado y para que aparezca de manera textual, debemos indicarlo precediéndolo de una contrabarra, \.
Nota: Recuerda que los archivos de configuración de los servicios son ejecutados por una `shell`. Para desactivar los carácteres reservados se debe colocar la contrabarra. Por ejemplo la propia contrabarra es un caracter reservado, si deseamos que aparezca debe ir precedida por ella misma \\.

Para definir un formato concreto se hace uso de variables predefinidas, *%nombre_variable*, pudiendo introducir cualquier texto en claro que se desee. Entre las distintas variables disponibles podrían destacarse las siguientes:

1. Dirección IP del equipo cliente FTP que accede al servidor: `%a`.
2. Hora/Fecha Local: `%t`.
3. Tiempo que dura la transferencia en segundos: `%T`.
4. Nombre del usuario tras la autentificación: `%l`. En caso de que la fase de autentificación falle se auditará como `UnKnow`.
5. Identificador del usuario (UID)/Nombre usuario que le loga: `%u`.
6. Nombre del usuario que accede al servicio, Anonymous o UnKnow: `%A`.
7. Cantidad de Bytes transferidos durante la solicitud de conexión: `%b`.
8. Dirección IP/Nombre de dominio del cliente FTP: `%h`.
9. Acción llevada a cabo desde el cliente: `%r`.
10. Código emitido por el servidor ante una solicitud: `%s` (`status`).

- **ExtendedLog *fichero acciones nickname***

Haciendo uso de uno de los formatos definidos con `LogFormat` nos permite especificar la ubicación del fichero que auditará el servicio `proftpd`. También indica que acciones realizadas desde el cliente serán registradas. Entre las distintas acciones cabría destacar las siguientes:

1. No auditar ninguna acción: `NONE`.
2. Auditar operaciones de autentificación: `AUTH` (USER, PASS).
3. Acciones de lectura: `READ` (RETR).
4. Acciones de escritura: `WRITE`.
5. Operaciones sobre directorios: `DIRS` (MKD, LIST, CWD, ...).

6. Información variada: `MISC` (Miscellaneous Commands).
7. Información adicional: `INFO` (PWD, SYST, ...).
8. Auditar todo: `ALL`.

4.6.7. Otros parámetros de configuración del servicio

Existe una multitud de parámetros para configurar `proftp`. Es imposible describirlos todos en este libro, pero puedes encontrarlos completamente explicados en `http://www.proftpd.org/localsite/Userguide/linked/userguide.html`.

En cualquier caso a continuación se describirán algunos de los parámetros más importantes que debes conocer.

Parámetros generales de configuración

- **ServerType inetd|standalone**

Permite configurar el servicio `proftpd` para que funcione de manera autónoma (`standalone`), o si por el contrario queremos que sea gestionado por el superdemonio/superservicio `inetd`. La elección de uno u otro dependerá de la cantidad de tráfico o solicitudes de conexión que sean recibidas, siendo aconsejable que funcione de manera autónoma contra mayor sea este número.

- **ServerName *nombre***

Este parámetro se utiliza para identificar al servidor. Recuerda que en el servidor `apache` también existía este parámetro y permitía la implementación de `Host Virtuales` basados en nombre de dominio. El protocolo FTP no permite esto y, por tanto, no es posible implementar `Host Virtuales` basados en nombre de dominio con `proftp`.

Por esta razón, este parámetro tan sólo tiene interés descriptivo.

- **ServerIdent on|off *descripción***

Si está a `on` al establecer la conexión muestra el mensaje colocado en *descripción*. Si está a `off` muestra el mensaje por defecto.

Nota: Para visualizar el mensaje es necesario establecer la conexión con el servidor desde la línea de comandos (mediante el comando `ftp`).

- **port *nº de puerto***

Este parámetro informa del `puertoTCP` a través del cual se atenderán las solicitudes de servicio FTP por parte de `proftpd`. Aunque es un parámetro fundamental no es necesario indicarlo en caso de utilizar el puerto por defecto `21`.

Debido a que no es posible la implementación de `Host Virtuales` basados en nombre de dominio, la utilización de `port` nos ofrecerá una alternativa que en la sección 4.10 desarrollaremos para implementar `Host Virtuales` basados en `puertoTCP`, utilizando aquellos puertos que el servidor tenga libres.

- **User *usuario*, Group *grupo* y UserAlias *alias usuario***

Estos parámetros están relacionados entre sí, siendo especialmente útiles cuando se configura un servidor FTP anónimo. La definición de los mismos nos permitirá decidir a que usuario del sistema suplanta el usuario anónimo que hace la conexión (cualquier usuario que se conecta al servicio debe estar registrado en el sistema; por esa razón, el usuario `anónimo` entrará en la máquina haciendose pasar por otro que lo esté).

El usuario `anónimo` (o `anonymous`) no será un usuario del sistema y a través de `UserAlias` podremos identificarlo con un usuario existente, por ejemplo, el usuario `ftp`.

- **AccessGrantMsg *mensaje***

Mensaje enviado al usuario una vez que se haya logado satisfactoriamente. Este mensaje se visualiza únicamente en el caso de que la conexión se lleve a cabo desde una consola (comando `ftp`).

- **AccessDenyMsg *mensaje***

Mensaje enviado al usuario en caso de que éste no se autentifique correctamente. Este mensaje se visualiza únicamente en el caso de que la conexión se lleve a cabo desde una consola (comando `ftp`).

- **MultilineRFC2228 on|off**

Permite garantizar compatibilidad (`on`) con todo tipo de clientes FTP.

- **ShowSymlinks on|off**

En el caso de que en nuestro sitio FTP se localicen enlaces simbólicos (ln -s) a otras ubicaciones del sistema de ficheros, mediante este parámetro podemos hacer que sean visibles (`on`), o hacer que pasen inadvertidos para los usuarios (`off`), haciéndoles creer que son directorios cuyo contenido es la ubicación a la que enlazan.

- **AllowOverWrite on|off**

Parámetro que establece permiso para sobreescribir los datos contenidos en el sitio FTP.

- **AuthUserFile *fichero***

Permite especificar la ruta del fichero de cuentas de usuario que serán utilizadas en la configuración de sitios FTP no anónimos. Como ya fue comentado, por defecto, este fichero es el de cuentas de usuarios del sistema `/etc/passwd`.

- **AuthGroupFile *fichero***

Permite especificar la ruta del fichero de grupos de usuarios que serán utilizados en la configuración de sitios FTP no anónimos. Por defecto, este fichero se corresponde con el de grupos del sistema `/etc/group`.

4.7. Configuración de un sitio anónimo con PROFTPD

Se entiende por servidor FTP anónimo, aquel al que puede conectarse cualquier usuario a través de un login común y sin necesidad de conocer una contraseña especial. Por lo general el login usuado es anonymous[10] y la contraseña puede ser cualquier cadena de caracteres.

Para construir un sitio anónimo habrá que tener en cuenta los siguientes aspectos:

1. Si se desea que el servicio sea totalmente anónimo, es necesario evitar que pueda entrar cualquier usuario del sistema a su `HOME` logándose vía FTP. Para ello, en el fichero de configuración `/etc/proftpd.conf` se deberá incluir la directiva `<Limit LOGIN></Limit>` denegando el acceso a todos los usuarios:

```
<Limit LOGIN>
  #Evitar acceso al HOME de los usuarios vía FTP (logándose):
  DenyAll
</Limit>
```

2. Tras escribir la directiva anterior, ningún usuario del sistema se podrá logar en él. Esto es un inconveniente porque los usuarios anómimos deberán hacerlo. Para conseguir esto existe una directiva que permitirá el logueo de los usuarios anónimos y que además especifica cual es la raíz (DocumentRoot) del sitio FTP anónimo para todos los usuarios que acceden al servicio. Ya no tiene sentido decir que el HOME de los usuarios es la raíz del sitio FTP al que acceden, debido a que estos son usuarios anónimos. Para hacer esto se usa la directiva `<Anonymous` *`ruta raíz`*`></Anonymous>`, la cual encerrará todos aquellos parámetros que deseamos que afecten de manera expresa al sitio FTP anónimo:

```
<Anonymous /var/ftp/pub>
  #Parámetros de configuración
</Anonymous>
```

   Por defecto, en el momento en que se instala `proftpd` en el equipo GNU/Linux, se crea el HOME del usuario `ftp`, `/var/ftp`, y dentro de este un directorio pensado para uso público, `/var/ftp/pub`. Si no se desea algo especial este directorio podría ser perfectamente válido como `HOME` del sitio anónimo.

3. En un sistema GNU/Linux vía FTP los permisos de archivos y directorios (`rwx`) hacen referencia a las cuentas de usuario registradas en el sistema. Esto significa que cuando un usuario anónimo se conecte al sistema lo hará suplantando a otro que sí está registrado. Es decir, se debe indicar en nombre de que usuario existente en el sistema llevan a cabo las acciones los usuarios anónimos que accedan a él. A este usuario suplantado se le deberán asignar unos privilegios que determinarán lo que se le permite hacer dentro del sitio FTP a él, y por tanto, a los usuarios anónimos que lo suplantan.

   Lo habitual es que el usuario suplantado sea `ftp`. Si aceptamos esa opción y queremos que en el momento en que acceda un usuario anónimo lo haga en calidad del usuario `ftp` con todos los permisos que se le han concedido dentro del sistema, deberemos agregar la siguiente línea:

[10] El login por lo general también suele poder ser `ftp` o `anonimo` en los sitios en lengua española.

```
<Anonymous /var/ftp/pub>
  User ftp
</Anonymous>
```

4. Que el servicio sea anónimo no significa que para acceder al servicio `proftpd` no haga falta logarse por parte del usuario anónimo. En realidad, lo que significa es que puede entrar todo aquel que se autentifique como usuario de ese servicio anónimo, aunque sin la necesidad de una contraseña registrada en el sistema.

   El usuario que puede acceder al servicio es el definido por `User`, por ejemplo en el caso anterior este usuario sería `ftp`. Por convenio, los clientes FTP, utilizan un usuario llamado `anonymous` para acceder a un sitio FTP anónimo. El usuario `anonymous` en realidad no tiene una cuenta en el sistema sino que es un `alias` de la cuenta de usuario a la que van a suplantar al acceder al sitio FTP anónimo. La forma de definir un alías es por medio del parámetro `UserAlias`. Así el ejemplo anterior quedaría:

```
<Anonymous /var/ftp/pub>
  User ftp
  #Creamos un alias denominado anonymous del usuario ftp:
  UserAlias anonymous ftp
</Anonymous>
```

   Esto signifca que permitimos entrar al servicio a aquellos que se loguen como `ftp` o como `anonymous`.

   Puedes comprobar, listando el contenido de `/etc/passwd`, que el usuario `ftp` no tiene una shell válida. Esto hace necesario el uso de `RequireValidShell off`.

5. Por último para que cualquiera que lo desee, usando como login `ftp` o `anonymous`, pueda establecer la comunicación FTP será necesario indicarlo mediante la directiva `<Limit LOGIN>` tal como se muestra en el siguiente ejemplo.

```
<Anonymous /var/ftp/pub>
  User ftp
  #Creamos un alias denominado anonymous del usuario ftp:
  UserAlias anonymous ftp
  RequireValidShell off
  <Limit LOGIN>
    #Se permite acceso a los logados como anonymous o como ftp
    AllowAll
  </Limit>
</Anonymous>
```

Ejercicio 4.3

Configurar un servicio FTP anónimo mediante `proftpd` para que usuarios anónimos puedan acceder al equipo servidor vía FTP en calidad del usuario `ftp`. La raíz del sitio FTP anónimo será el directorio `/var/ftp/`, donde no se podrá sobreescribir o almacenar nada. Contendrá dos subcarpetas `/var/ftp/descargas` y `/var/ftp/libre`. El contenido de `/var/ftp/descargas` no podrá ser borrado, aunque sí se podrán leer, descargar o subir ficheros. En el subdirectorio

`/var/ftp/libre` se podrá leer y alojar información con permisos de escritura, es decir, podrán borrar su contenido.

Para evitar sobrecargar al servidor permitiremos un máximo de 25 conexiones desde equipos diferentes.

Solución del ejercicio 4.3

En primer lugar se deberá crear el directorio raíz del sitio FTP anónimo, y sus subdirectorios.

```
[root@linux] mkdir /var/ftp/descargas
[root@linux] mkdir /var/ftp/libre
[root@linux] chown -R ftp.ftp /var/ftp
```

A continuación se editará el fichero `/etc/proftpd.conf` y se efectuará la configuración del servicio `proftpd` como anónimo.

```
#Fichero de configuración del servicio proftpd: /etc/proftpd.conf
DefaultServer on
#La cuenta de usuario ftp tiene asignada una shell falsa:
RequireValidShell off
#Evitamos que los usuarios anónimos naveguen por el sistema de ficheros:
DefaultRoot ~
#Limitamos la conexiones a 25:
MaxClients   25   "Se ha superado el nº máximo de conexiones abiertas"

#Evita que desde un mismo equipo se realicen más de 1 conexión
MaxClientsPerHost   1   "Ya tienes abierta una conexion al servidor, lo siento"
<Limit LOGIN>
  DenyAll #Evitamos que los usuarios registrados accedan vía ftp
</Limit>
<Anonymous /var/ftp/>
   User ftp #Los usuarios anónimos suplantan al usuario ftp
   UserAlias anonymous ftp #Al logarse como anonymous suplantan al usuario ftp
   <Limit LOGIN>
     AllowAll #Obligamos que los accesos sean como usuario anonymous
   </Limit>
   <Limit WRITE>
     #Evitamos que pueda escribirse sobre el sitio FTP, aunque el usuario
     #ftp tenga permisos para ello:
     DenyAll
   </Limit>
   <Directory /var/ftp/descargas>
     <Limit STOR>
       AllowAll #Permitimos sólo leer y almacenar
     </Limit>
   </Directory>
   <Directory /var/ftp/libre>
     <Limit WRITE>
       AllowAll #Damos permisos de escritura y borrado
     </Limit>
   </Directory>
</Anonymous>
```

Por último reiniciaremos el servicio para que los cambios surtan efecto,

```
[root@linux]# service proftpd restart
```

Se deben dejar claros algunos aspectos de la solución anterior:

1. En el directorio `/var/ftp/descargas` se pueden almacenar únicamente archivos, el sistema no dejará crear carpetas.
2. En el directorio `/var/ftp/libre` será posible almacenar archivos y crear carpetas nuevas.
3. Si el directorio `/var/ftp/` contiene otras carpetas que no están contenidas en la configuración del archivo `/etc/proftpd.conf`, estas serán de sólo lectura.

4.8. Combinación de servicios ftp: anónimo y no anónimo

En muchas ocasiones es interesante que el servicio `proftpd` combine las configuraciones vistas y permita al mismo tiempo el acceso no anónimo a los distintos usuarios registrados en el equipo y anónimo a todos aquellos que lo deseen. Evidentemente los recursos a los que se accederá de forma anónima y no anónima podrán ser diferentes.

Conseguir esto es tan sencillo como mantener el bloque <`Anonymous` *raíz sitio ftp anónimo*> </`Anonymous`> en el fichero de configuración y eliminar la directiva que evita todo acceso al servicio por parte de los usuarios del sistema, `<Limit LOGIN>DenyAll</Limit>`.

La estructura del archivo de configuración en este caso sería:

```
#Fichero de configuración del servicio proftpd: /etc/proftpd.conf
Aquí los parámetros generales de configuración
<Limit LOGIN>
  Aquí se indicarán los usuarios permitidos y no permitidos
</Limit>
<Anonymous /var/ftp/>
  User ftp
  UserAlias anonymous ftp
  RequireValidShell off
  Parámetros de configuración del sitio anónimo
</Anonymous>
```

Ejercicio 4.4

Configurar `proftpd` para que pueda dar tanto servicio anónimo, con las mismas características del ejercicio anterior, como no anónimo con la finalidad de que los usuarios pertenecientes al grupo `privilegiados` puedan acceder a su `HOME`.

Se deberá garantizar que tan sólo se podrá acceder al servidor FTP desde los equipos pertenecientes a las redes internas de una supuesta empresa: `192.168.1.0/24`, `192.168.2.0/24` y `192.168.3.0/24`.

Solución del ejercicio 4.4

En esta solución se asumirá que ha sido creado el grupo de usuarios `privilegiados` (`groupadd privilegiados`) y que se han modificado adecuadamente las cuentas de usuario (`usermod -G privilegiados` *`usuario`*) a las que se les desea permitir el acceso de manera no anónima al servidor FTP.

```
#Fichero de configuración del servicio proftpd: /etc/proftpd.conf
DefaultServer on
DefaultRoot ~ #No se perimite naveguar fuera de la raiz
<Limit LOGIN> #Decimos quienes pueden logarse de manera no anónima
  #Permitimos únicamente al grupo privilegiados logarse:
  AllowGroup privilegiados
  DenyAll #Evitamos que puedan acceder otros usuarios
</Limit>
<Limit LOGIN>
  order allow,deny #Primero mira autorizados y después los denegados
  allow from 192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24
  deny from all #Denegamos el acceso a cualquier otro equipo
</Limit>
<Anonymous /var/ftp/>
   User ftp #Los usuarios anónimos suplantan al usuario ftp
   UserAlias anonymous ftp
   #Como la cuenta del usuario ftp tiene asignada una shell falsa entonces:
   RequireValidShell off
   <Limit LOGIN>
     AllowAll #Sólo acceden los logados como usuario anonymous
   </Limit>
   <Limit WRITE>
     DenyAll #Evitamos que pueda escribirse sobre el sitio FTP
   </Limit>
   <Directory /var/ftp/descargas/subidas>
     <Limit WRITE>
       AllowAll #Permitimos la escritura en directorio subidas
     </Limit>
  </Directory>
<Anonymous>
```

4.9. Hosts Virtuales basados en IP

Con una sintaxis similar a la utilizada en la configuración del servidor Web `apache`, puede configurarse `proftpd` para implementar distintos sitios FTP mediante el uso de `HostVirtuales` basados en IP:

```
<VirtualHost direccionIP>
#Parámetros/directivas
</VirtualHost>
```

Esto va a permitirnos albergar dentro de un mismo equipo diferentes sitios FTP anónimos y no anónimos mediante la asignación a cada uno de ellos de una IP diferente.

4.10. Hosts Virtuales basados en puerto

En la configuración de `apache` se admitían tanto la creación de `Hosts Virtuales basados en dirección IP`, como `Hosts Virtuales basados en nombre`. Por contra, en este caso no es posible asignar a una misma dirección IP distintos nombres de dominio, y servir un sitio FTP independiente bajo cada uno de los nombres (Host Virtuales basados en nombre).

Esto es debido a que el protocolo FTP tan sólo observa la dirección IP ante una solicitud de servicio, y no se fija en el nombre de dominio que fue resuelto por el servidor `DNS` (el protocolo `http` tenía en cuenta todo `dirección IP/nombre dominio` a la hora determinar que sitio Web servir).

Una alternativa a esta opción (Host Virtuales basados en nombre) sería servir diferentes sitios FTP mediante la misma dirección IP pero por puertos TCP/UDP diferentes. Dentro de cada <`VirtualHost`> tan sólo será necesario especificar el puerto por donde serán escuchadas las peticiones de servicio realizadas por los clientes:

```
Port nº de puerto
```

En relación al número de puerto TCP/UDP a especificar, hay que tener en cuenta, que por defecto, si no se indica lo contrario se asumirá el puerto 21. En caso de indicar uno diferente, deberemos asegurarnos que se trata de un puerto no utilizado por otro servicio. El fichero `/etc/services` nos informa de los distintos servicios que existen y el puerto TCP/UDP por el que se ofrece, lo que puede servirnos para descartar que puertos no asignar. No obstante, se sugiere utilizar aquellos puertos que estén por encima del 1024.

Para comprender con una mayor profundidad todos estos aspectos se propone implementar la siguiente configuración del servicio `proftpd` que se especifica en el siguiente ejercicio práctico.

Ejercicio 4.5

Configurar `proftpd` para dar servicio a tres sitios FTP independientes mediante el uso de Host Virtuales basados en IP y puerto TCP/UDP:

Dirección IP	Puerto TCP	DocumentRoot	Permisos	Usuarios
192.168.100.x	21	/var/ftp/descargas1	Lectura	Anónimo
192.168.100.x	10001	HOME	Lectura/Escritura	No Anónimo
192.168.100.x+50	21	/var/ftp/descargas2	Lectura/Escritura	Anónimo
		HOME	Lectura/Escritura	Grupo `ies`

En el primer sitio FTP anónimo los usuarios anónimos accederán suplantando una cuenta de usuario con permisos limitados llamada `invitado`. En el segundo sitio FTP anónimo suplantaremos al usuario `ftp`.

Además, para evitar tener que hacer uso de direcciones IP, configuraremos `BIND` para acceder a cada uno de los sitios FTP como `ftp.sitioftp1`x`.com`, `ftp.sitioftp2`x`.com` y `ftp.sitioftp3`x`.com`. Nombres en los que la x representa el nº de equipo desde el que se lleva a cabo la práctica.

Solución del ejercicio 4.5

El fichero de configuración de `profptp` sería en este caso:

```
#Fichero de configuración del servicio proftpd: /etc/proftpd.conf
DefaultServer on
<VirtualHost 192.168.100.1>
  Port 21
  #La cuenta de usuario invitado tiene asignada una shell falsa
  RequireValidShell off
```

Continúa en página siguiente

```
  DefaultRoot ~
  <Limit LOGIN> #Limitamos el logueo no anónimo
   DenyAll #Evitamos el acceso de usuarios registrados en el sistema
  </Limit>
  <Anonymous /var/ftp/descargas1>    User invitado
   UserAlias anonymous invitado
   <Limit LOGIN>
      AllowAll
   </Limit>
   <Limit WRITE>
    DenyAll #Evitamos la escritura en el sitio FTP
   </Limit>
  </Anonymous>
</VirtualHost>
<VirtualHost 192.168.100.1>
 #El puerto 10001 permite la conexión con este sitio FTP no anónimo
 Port 10001
 DefaultRoot ~
</VirtualHost>

<VirtualHost 192.168.100.51>
 Port 21
 DefaultRoot ~
 <Limit LOGIN>
  AllowGroup ies #Permitimos únicamente al grupo ies logarse
  DenyAll #El resto no pueden acceder
 </Limit>
 <Anonymous /var/ftp/descargas1>
  User ftp
  UserAlias anonymous ftp
 RequireValidShell off
  <Limit LOGIN>
     AllowAll
  </Limit>
 <Anonymous>
</VirtualHost>
```

Para crear la cuenta de invitado y el HOME asociado a ella bastaría con ejecutar,

```
[root@linux]]# mkdir /var/ftp/descargas1
[root@linux]]# chown -R ftp.ftp /var/ftp
[root@linux]]# useradd -d /var/ftp/descargas1 -s /bin/false invitado
```

Por último reiniciamos el servicio,

```
[root@linux]# service restart
```

Aclaraciones sobre la configuración propuesta:

1. Para asignar más de una dirección IP a una interfaz de red haremos uso del comando `ifconfig`, exactamente igual que cuando se configuraban en `apache Host Virtuales basados en dirección IP`:

   ```
   [root@linux]# ifconfig ethN:M direccionIP   resto parámetros
   ```

   siendo `N` el número de interfaz de red a configurar (0,1,2,...) y `M` el número de alias IP que se le quiere asignar (1,2,3,...).

```
[root@linux]# ifconfig eth0:1 192.168.100.51
```

2. Excepto el parámetro `DefaultServer on`, de importancia ya comentada al principio de la presente práctica, que debe ser único dentro del fichero de configuración, el resto de parámetros es conveniente que se asignen de manera independiente a cada uno de los <`VirtualHost`> que se definan, lo que evitará confusiones, y posibles defectos de funcionamiento.

3. Para acceder al sitio FTP no anónimo servido por `proftpd` a través del puerto 10001 es necesario especificarlo expresamente en la URL del cliente FTP. Por ejemplo si un usuario llamado `arturo` quiere acceder lo indicará como:

```
ftp://arturo@ftp.sitioftp21.com:10001
```

o lo que es lo mismo

```
ftp://arturo@192.168.100.1:10001
```

Ejercicio 4.6

Configura los servicios `proftpd`, `named` y `httpd`, para dar servicio a cuatro sitios FTP independientes dentro del equipo GNU/Linux haciendo uso de `Hosts Virtuales` basados en dirección IP y puerto TCP/UDP. Las especificaciones a cumplir se detallan a continuación:

1. Configura `proftpd` para que el servicio que ofrezca presente las siguientes características:

VirtualHost	DocumentRoot	Permisos	Umask	Usuarios
192.168.10.x Puerto 21	/var/ftp/desarrollo	Leer/Escribir	F: rw-r--r-- D: rwxrwxr-x	desarrollo1, desarrollo2, desarrollo3 (grupo de usuarios → dep-desarrollo)
192.168.10.x Puerto 50000	/var/ftp/informacion	Leer	–	Anónimo
192.168.10.x+50 Puerto 21	/var/ftp/empleados	Leer	–	Grupo de usuarios: empleados (emp1, emp2, emp3, emp4, . . . , emp22)
	/var/ftp/empleados/informes	Leer/Escribir	F: rw-rw---- D: rwxrwx---	Grupo de usuarios: empleados (emp1, emp2, emp3, emp4, . . . , emp22)
192.168.10.x+50 Puerto 55555	/var/ftp/zonadescargas	Leer	–	Anónimo
	/var/ftp/zonadescargas/subidas	Leer/Escribir	F: rw-r----- D: rwxr-x---	Anónimo
	HOME	Leer/Escribir	F: rw-rw-r-- D: rwxrwxr-x	secre1, secre2, secre3 y grupo direccion: director1, director2

Nota: La x utilizada se corresponde con el número de tu equipo. El libro, y por tanto los ejercicios están pensados para que ser realizados en una clase. Las aulas suelen tener los ordenadores numerados, es ese número el que se está señalando con la x. La "F:" hace referencia a ficheros (permisos para ficheros) y la "D:" a los directorios.

En resumen, este ejercicio nos pide configurar cuatro sitios: un primer sitio FTP no anónimo, un segundo anónimo, un tercero también no anónimo y un último híbrido anónimo/no anónimo.

2. Con el fin de establecer un control sobre el acceso al servicio `proftpd` deberán tenerse en cuenta igualmente las siguientes características:

VirtualHost	Permitidos	Conexiones max.	Nombre Dominio	Auditoría
192.168.10.*x* Puerto 21	Red: 192.168.1.0/24	6 conexiones máx. 2 por usuario 1 por equipo	desarrollo.empresa.com	/var/log/desftp
192.168.10.*x* Puerto 50000	Redes: 192.168.*z*.0/24	50 conexiones máx.	anonimo.empresa.com	/var/log/anoftp
192.168.10.*x*+50 Puerto 21	SubRed: 192.168.2.64/27 Equipos: 192.168.3.1, 192.168.3.2, 192.168.4.1	40 conexiones máx. 4 por usuario 3 por equipo	empleados.empresa.com	/var/log/emplog
192.168.10.*x*+50 Puerto 55555	Cualquier host o red	15 conexiones máx.	ftp.empresa.com	/var/log/serftp

Nota: La z es una variable que toma los valores 1, 2, 3, 4 y 5. Luego 192.168.z.0/24 hace referencia a las 5 redes: 192.168.1.0, 192.168.2.0, 192.168.3.0, 192.168.4.0 y 192.168.5.0.

El formato en que se deberán auditar los accesos/transferencias sobre el servidor no será el suministrado por `TranferLog` sino que lo personalizaremos de la siguiente forma:

```
Cliente: IP cliente   Usuario: usuario   Fecha/Hora: fecha/hora   Accion: accion
```

Además:

a) Deberá evitarse que pueda navegarse por todo el sistema de ficheros del servidor excepto en el primer sitio FTP no anónimo.
b) No deberá advertirse la existencia de enlaces simbólicos.
c) Deberá garantizarse la máxima compatibilidad posible con todos los clientes FTP existentes.
d) Las sesiones abiertas desde los clientes serán rechazadas en caso de inactividad continuada durante 5 minutos, o en caso de no recibir una orden de transferencia en 8 minutos para todos los sitios FTP.
e) Excepto las cuentas de los `desarrolladores`, `secretarios` y de los usuarios del grupo de `direccion`, el resto de usuarios no deberán tener privilegios para iniciar sesión en el servidor.

3. Configura el servicio `named` para que pueda accederse a cada uno de los servicios anteriores a través de un nombre de dominio en lugar de tener que conocer la dirección IP del servidor. Advertir que el dominio es único, `empresa.com`.

4. Configura el servicio `httpd` (apache) para que el primero de los sitios FTP, gestionado por el personal de desarrollo de la empresa, haga las veces de alojamiento para dos sitios `Web`:

a) La raíz del primer sitio `Web` será `/var/ftp/desarrollo/webempresa`, al cual deberán poder acceder de manera `anónima` cualquier usuario a través del nombre de dominio `www.empresa.com`. A su vez dispondrá de una subcarpeta que contendrá `scripts perl` llamada `cgi-bin` que podrán ser ejecutados a través de la URL: `http://www.empresa.com/perl/`*nombre_script.pl*.

b) La raíz del segundo sitio `Web` será `/var/ftp/desarrollo/webconfidencial`, siendo este `no anónimo`, debiendo permitir el acceso únicamente a los usuarios `secretario`, `jefeseccion1`, `jefeseccion2`, `director1` y `director2`, englobados todos ellos en un grupo de cuentas de usuarios permitidos llamado `confidentes`. Para acceder a esta parte `no anónima` del sitio Web de la empresa debe configurarse `apache` para hacerlo a través de la siguiente URL: `http://www.empresa.com/confidencial`.

Solución del ejercicio 4.6

En primer lugar hay que hacer operaciones para crear los directorios (DocumentRoot), configurar sus permisos, generar las cuentas de usuario y grupos, y asignar las direcciones IP (suponemos que únicamente disponemos de una interfaz de red). Para ello, podríamos ejecutar uno a uno los comandos necesarios, o ejecutar el siguiente script para hacerlo todo de una vez (debe ejecutarlo `root`, ya que es el único que tiene privilegios para llevar a cabo la configuración del sistema):

```
#Script de configuración del sistema servidor
# 1) Creamos los directorios raíz (DocumentRoot) de los distintos sitios FTP
mkdir -p /var/ftp/desarrollo /var/ftp/informacion /var/ftp/empleados/informes
/var/ftp/zonadescargas/subidas
#Con la opción -p se crea al mismo tiempo los directorios que cuelgan de
#/var/ftp y sus subdirectorios

# 2) Creamos los grupos y cuentas de usuario en el servidor
groupadd depdesarrollo #Creamos el grupo departamento desarrollo
useradd -d /var/ftp/desarrollo -g depdesarrollo -s /bin/bash desarrollo1
useradd -d /var/ftp/desarrollo -g depdesarrollo -s /bin/bash desarrollo2
useradd -d /var/ftp/desarrollo -g depdesarrollo -s /bin/bash desarrollo3
echo "Indica las password para los tres desarrolladores:"
passwd desarrollo1
passwd desarrollo2
passwd desarrollo3
groupadd empleados #Creamos el grupo de usuarios empleados
#Creamos una variable contador que informara del numero de empleados creados
CONTADOR=1
#Mediante un bucle creamos las 22 cuentas de empleados:
while test $CONTADOR -le 22
do
useradd -d /var/ftp/empleados -g ftp -G empleados -s /bin/false emp$CONTADOR
passwd "emp$CONTADOR"
CONTADOR=‘expr $CONTADOR + 1‘
done
#Ahora haremos los mismo con las cuentas de usuario de los secretarios
$CONTADOR=1
while [ $CONTADOR -le 3 ]
do #Creamos las cuentas con las opciones por defecto:
useradd "secre$CONTADOR"
passwd "secre$CONTADOR"
done
groupadd direccion #Creamos el grupo de direccion
mkdir -p /home/direccion/director1 /home/direccion/director2
useradd -d /home/direccion/director1 -g direccion -s /bin/bash director1
chown -R director1.direccion /home/direccion/director1
echo "Indica la password para el usuario director1:"
passwd director1
useradd -d /home/direccion/director2 -g direccion -s /bin/bash director2
```

Continúa en página siguiente

```
echo "Indica la password para el usuario director2:"
passwd director2
chown -R director2.direccion /home/direccion/director2

# 3) Cambiamos los propietarios de los directorios, ajustando sus permisos
#Damos la propiedad al usuario ftp, grupo ftp (ftp.ftp) de todo el
#contenido que cuelga del HOME de: /var/ftp
chown -R ftp.ftp /var/ftp
#Modificamos el grupo asociado /var/ftp/desarrollo:
chgrp -R depdesarrollo /var/ftp/desarrollo
#Concedemos todos los permisos al grupo de desarrolladores sobre el
#directorio desarrollo para que puedan tanto leer como escribir:
chmod -R g+rwx /var/ftp/desarrollo
#Hacemos lo mismo con el grupo empleados
chgrp -R empleados /var/ftp/empleados
#Concedemos todos los permisos sobre informes
chmod -R g+rwx /var/ftp/empleados/informes
#Permitimos que cualquier usuario que acceda de manera anónima, al
#suplantar al usuario propietario FTP tenga todos los permisos
chmod -R u+rwx,g-rwx,o-rwx /var/ftp/zonadescargas/subidas

# 4) A continuación mostraremos el contenido de /etc/group y
#/etc/passwd para comprobar que los grupos y las cuentas de
#usuario han sido creados correctamente:
echo "A continuación se muestran los 3 grupos de usuarios creados:"
more /etc/group | tail -3
echo "Y la lista de las últimas 30 cuentas de usuario creadas:"
more /etc/passwd | tail -30
# 5) Una vez terminada la gestión de las cuentas de usuarios,
#crearemos los ficheros de auditoria:
touch /var/log/desftp /var/log/anoftp /var/log/emplog /var/log/serftp
# 6) Configuramos la interfaz de red del servidor para asignarle
# las direcciones IP
ifconfig eth0 192.168.10.x
ifconfig eth0:1 192.168.10.50+x
```

Nota: Recuerda que la x empleada en las dos últimas líneas del script indica el número del equipo con el que trabajas.

En segundo lugar, tras haber configurado GNU/Linux para cumplir con las especificaciones que se especifican en el enunciado del ejercicio práctico, pasaremos a la configuración del servicio `proftpd`. En adelante se supondrá que el equipo que se está configurando es el nº 1, por lo que la x de las direcciones IP será 1.

```
#Fichero de configuración del servicio proftpd: /etc/proftpd.conf
DefaultServer on
#A continuación definimos un formato personalizado para la auditoria
LogFormat formato_personal "Cliente: \"%h\" Usuario: \"%u\"
                      Fecha/Hora: \"%t\" Accion: \"%r\" "
#Garantizamos compatibilidad con el mayor número posible de clientes
MultilineRFC2228 on
#Configuración correspondiente al primer Virtual Host: No Anónimo
<VirtualHost 192.168.10.1>
  Port 21
  #Los desarrolladores que accedan requieren una shell válida
  RequireValidShell on
  Umask 033 002 #Mascara para archivos y directorios
```

Continúa en página siguiente

```
  #En caso de inactividad durante 300 sg se rechazará la conexión:
  TimeoutIdle 300
  #Se perderá la conexión por estar más de 480 sg sin hacer transferencias:
  TimeoutNoTransfer 480
  MaxClients   6   "No es posible la conexión. El servidor esta saturado. Ya
                              existen 6 conexiones activas"
  MaxClientsPerUser  2  "Lo siento, pero ya tienes abiertas 2 sesiones FTP
                             con tu cuenta de usuario"
  MaxClientsPerHost  1  "¡¡No se admite más de 1 conexión al servidor FTP
                              por equipo!!"
  #Inhabilitamos la auditoria por defecto guardada en /var/log/xferlog :
  TranferLog NONE
  #Auditoria personalizada:
  ExtendedLog /var/log/desftp read,write,auth formato_personal
  #Hacemos que los enlaces simbólicos sean transparentes al cliente FTP
  ShowSymlinks off
  #Limitamos el acceso de tal forma que solo accedan los desarrolladores:
  <Limit LOGIN>
    AllowUser desarrollo1 desarrollo2 desarrollo3
    DenyAll
  </Limit>
  #Permitimos el acceso a equipos de la red 192.168.1.0/24
  <Limit LOGIN>
    Order allow,deny
    Allow from 192.168.1.0/24
    Deny from all
  </Limit>

</VirtualHost>

#Configuración correspondiente al segundo Virtual Host: Anónimo
<VirtualHost 192.168.10.1>
  Port 50000
  #En caso de inactividad durante 300 sg se rechazará la conexión/sesión
  TimeoutIdle 300
  #Se perderá la conexión por estar más de 480 sg sin transferir:
  TimeoutNoTransfer 480
  MaxClients 50 ¡¡El servidor ya tiene las 50 conexiones anónimas establecidas!!
  #Inhabilitamos la auditoria por defecto:
  TranferLog NONE
  #Auditoria personalizada:
  ExtendedLog /var/log/anoftp read,write,auth formato_personal
  #Evitamos que acceda cualquier usuario logándose:
  <Limit LOGIN>
    DenyAll
  </Limit>
  <Anonymous /var/ftp/informacion>
    User ftp
    UserAlias anonymous ftp
    #Los usuarios anónimos acceden vía FTP al servidor suplantando
    # a FTP el cual no dispone de una shell válida
    RequireValidShell off
    <Limit LOGIN>
      Allow All
    </Limit>
    #Permitimos acceder a cualquier equipo perteneciente a cualquier red interna
    <Limit LOGIN>
```

Continúa en página siguiente

```
      Order allow,deny
      Allow from 192.168.0.0/16
      Deny from all
    </Limit>
    <Limit WRITE>
      DenyAll
    </Limit>
  </Anonymous>
</VirtualHost>

#Configuración correspondiente al tercer Virtual Host: No anonimo
<VirtualHost 192.168.10.51>
  Port 21
  #Las cuentas de usuario del grupo empleados no se les ha
  #asignado una shell válida en el momento de la creación de la cuentas
  #para evitar que inicien sesión en el servidor
  RequireValidShell off
  #En caso de inactividad durante 300 sg se rechazará la conexión/sesión:
  TimeoutIdle 300
  #Se perderá la conexión por estar más de 480 sg sin transferir:
  TimeoutNoTransfer 480
  MaxClients  40  "¡¡Perdona. El servidor FTP no admite más de
                              40 conexiones simultáneas!!"
  MaxClientsPerUser  4  "¡¡Atención!! Tienes abiertas 4 sesiones FTP
                              con tu cuenta de usuario"
  MaxClientsPerHost   3  "¡¡No se admiten más de 3 conexiones al
                             servidor FTP por equipo!!"
  #Inhabilitamos la auditoria por defecto
  TranferLog NONE
  #Auditoria personalizada:
  ExtendedLog /var/log/emplog read,write,auth formato_personal
  #Permitimos el acceso no anónimo a empleados
  <Limit LOGIN>
    AllowGroup empleados
    DenyAll
  </Limit>
  <Limit LOGIN>
    Order allow,deny
    Allow from 192.168.2.64/27,192.168.3.1,192.168.3.2,192.168.4.1
    Deny from all
  </Limit>
  <Limit WRITE>
    DenyAll #No se permite escribir sobre /var/ftp/empleados
  </Limit>
  <Directory /var/ftp/empleados/informes>
    #Permisos de creación de archivos y directorios dentro de informes:
    Umask 007 007
    <Limit WRITE>
      AllowGroup empleados
      DenyAll
    </Limit>
  </Directory>
</VirtualHost>

#Configuración correspondiente al cuarto Virtual Host: No Anónimo / Anónimo
<VirtualHost 192.168.10.51>
  Port 55555
```

Continúa en página siguiente

```
  RequireValidShell  on #Los usuarios no anónimos requieren una shell
  TimeoutIdle 300
  TimeoutNoTransfer 480
  MaxClients  15  "El servidor FTP no admite más de 15 conexiones simultáneas"
  TranferLog NONE #Inhabilitamos la auditoria por defecto
  #Auditoria personalizada:
  ExtendedLog /var/log/serftp read,write,auth formato_personal
  #Permitamos el acceso no anónimo a los secretarios y grupo direccion
  <Limit LOGIN>
    AllowUser secre1 secre2 secre3
    AllowGroup direccion
    DenyAll
  </Limit>
  #Explícitamente indicamos que puede accederse desde cualquier equipo
  <Limit LOGIN>
    Allow from all
  </Limit>
  <Limit WRITE>
    #Explícitamente indicamos a proftpd que si el sistema
    #de ficheros lo permite, que los usuarios no anónimos puedan
    #llevar a cabo escrituras en disco vía FTP
    AllowAll
  </Limit>
  Umask 113 002
  <Anonymous /var/ftp/zonadescargas>
    #Cuenta de usuario que será suplantada por los usuarios anónimos
    #al conectarse al servidor vía FTP
    User ftp
    #Alias de la cuenta de usuario suplantada, anonymous:
    UserAlias anonymous ftp
    #Para que anónimamente accedan al servidor vía FTP deberemos
    #indicar a proftpd que no requieren una shell valida,
    # ya que el usuario suplantado, FTP, no la tiene:
    RequireValidShell   off
    <Limit WRITE>
      #No se permite que los usuarios anónimos puedan escribir en la
      # raíz del sitio FTP
      DenyAll
    </Limit>
    <Directory /var/ftp/zonadescargas/subidas>
      #Permisos de creación de archivos y directorios dentro de subidas
      Umask 137 027
      <Limit WRITE>
       #Sobre el directorio subidas permitimos escribir a todos:
       AllowAll
      </Limit>
    </Directory>
  </Anonymous>
</VirtualHost>
```

Tras editar el fichero `/etc/proftpd.conf` tan sólo nos queda reiniciar el servicio `proftpd` para comprobar que el servidor FTP funciona de la manera esperada:

```
[root@linux]# service proftpd restart
```

A continuación configuraremos BIND para que de servicio de resolución de nombres al dominio `empresa.com`. Para ello, crearemos una nueva zona dentro `/etc/named.conf` correspondiente al dominio `empresa.com`, y editaremos el correspondiente fichero de zona `/var/named/empresa.com`

para que resuelva los nombres de equipos dentro del dominio especificados, `desarrollo`, `anonimo`, `empleados`, FTP y `www`, todos ellos correspondientes al mismo equipo servidor (el mismo equipo hace las veces de servidor FTP, `DNS` y `HTTP`).

El contenido de `/etc/named.conf` sería:

```
#Contenido mínimo del fichero /etc/named.conf
#para la realización del ejercicio práctico
options {
                        directory "/var/named";
};

zone "empresa.com"in {
                        type master;
                        file "maestra.empresa.com";
};
```

y el archivo de zona llamado `maestra.empresa.com` tendría el siguiente contenido:

```
empresa.com.              IN SOA   servidor.empresa.com.  root.localhost.
                                                          (3010200602;
                                                          21600;
                                                          10800;
                                                          604800;
                                                          21600; )
empresa.com.              IN NS    servidor.empresa.com.
desarrollo.empresa.com.   IN A     192.168.10.1
anonimo.empresa.com.      IN A     192.168.10.1
empleados.empresa.com.    IN A     192.168.10.51
ftp.empresa.com.          IN A     192.168.10.51
www.empresa.com.          IN A     192.168.10.1
```

Como es lógico, tras relizar las correspondientes modificaciones en la configuración del servidor BIND, deberemos reiniciarlo para que los cambios surtan efecto:

```
[root@linux]# service named restart
```

Para comprobar la correcta configuración de BIND una posibilidad es realizar un `dig` a alguno de los nombres de los equipos del dominio `empresa.com` y observar si se lleva a cabo la resolución a su correspondiente dirección IP:

```
[root@linux]# dig desarrollo.empresa.com
```

Nota: Recuerda que la dirección IP del ordenador que hace de DNS debemos colocarla enel fichero `/etc/resolv.conf` ⇒ `nameserver 192.168.10.1`

En el caso de que todo funcione de la manera esperada, a continuación pueden realizarse las comprobaciones sobre el servicio `proftpd`. En este caso podemos emplear los nombres de dominio:

```
ftp://anonimo.empresa.com:50000
```

.

Por último, configuraremos el servicio `httpd` para que los directorios del primer sitio FTP se correspondan con sitios Web servidos por `apache`.

En la configuración que se presenta a continuación asumiremos que `apache` no sirve un único sitio Web, sino que sirve diferentes sitios de manera independiente mediante el uso de Host Virtuales basados en nombre e IP. Además asumiremos que las cuentas de usuario de `apache` son diferentes de las que hace uso del sistema GNU/Linux, por lo será necesario crearlas previamente mediante el uso de `htpasswd`:

```
[root@linux] htpasswd -c /usr/local/apache/passwd/passwebconfidencial
                                                    cuenta_usuario_apache
```

⇓

```
[root@linux] htpasswd -c /usr/local/apache/passwd/passwebconfidencial secretario
[root@linux] htpasswd /usr/local/apache/passwd/passwebconfidencial jefeseccion1
[root@linux] htpasswd /usr/local/apache/passwd/passwebconfidencial jefeseccion2
[root@linux] htpasswd /usr/local/apache/passwd/passwebconfidencial director1
[root@linux] htpasswd /usr/local/apache/passwd/passwebconfidencial director2
```

El contenido del archivo `/etc/httpd/conf/httpd2.conf` sería:

```
# Fichero de configuración de la versión 2 de apache:
                    #/etc/httpd/conf/httpd2.conf (Mandriva 2005)
ServerRoot /etc/httpd/2.0
...
NameVirtualHost 192.168.10.1
<VirtualHost 192.168.10.1>
   ServerName www.empresa.com
  DocumentRoot /var/ftp/desarrollo/webempresa
  ScriptAlias /perl/ /var/ftp/desarrollo/webempresa/cgi-bin/
  Alias /confidencial /var/ftp/desarrollo/webconfidencial
  <Directory /var/ftp/desarrollo/webconfidencial>
    AuthType Basic # Tipo de Autentificación por parte de los usuarios
    AuthName "Zona Restringida a los Usuarios"
    AuthUserFile /usr/local/apache/passwd/passwebconfidencial
    Require user secretario jefeseccion1 jefeseccion2 director1 director2
  </Directory>
</VirtualHost>
# Resto de HostVirtuales
...
```

Al igual que en el resto de servicios, una vez realizados los cambios en la configuración de `apache` reiniciaremos el servicio `httpd` y realizaremos las comprobaciones necesarias mediante el uso de un cliente Web:

```
[root@linux]# service httpd restart
```

Nota: Recuerda que debe existir al menos un `index.html` en la carpeta raíz y algún `script perl` en `cgi-bin`.

Capítulo 5

CONFIGURACIÓN DE UN SERVIDOR DE CORREO: POSTFIX

5.1. Introducción al servicio de correo electrónico

Desde los inicios de Internet hasta la fecha, uno de los servicios ofrecidos con mayor aceptación sin duda es el correo electrónico (e-mail). El servicio email, junto con la navegación Web y los nuevos servicios P2P (emule, amule, direct connect, ...), forma parte de los servicios más exitosos y utilizados de todos los existentes. Nos permite enviar todo tipo de información en formato digital (texto, imágenes y archivos) aprovechándonos de las infraestructuras de telecomunicaciones existentes, con todas las ventajas respecto al correo tradicional que eso conlleva.

Con la finalidad de normalizar su funcionamiento están disponibles los documentos RFC-821, RFC-822 (protocolo SMTP), RFC-1939 (protocolo POP3), RFC-3501 (protocolo IMAP), aunque existen otros muchos que detallan aspectos más concretos de cada uno de los protocolos[1]. En concreto, el documento RFC-821 describe el protocolo SMTP (Simple Mail Transfer Protocol o Protocolo Simple de Trasnsferencia de Correos), utilizado en Internet para establecer la forma en que deben transportarse los emails entre los distintos servidores de correo públicos existentes. Así un correo emitido desde un equipo cualquiera, que esté conectado en red, puede llegar a su destino. Por otro lado, el documento RFC-822 normaliza el formato que deben utilizar los emails. El RFC-1939 describe el protocolo POP3 (Post Office Protocol o Protocolo de Oficina de Correos). En él se detalla la forma en que los correos deben entregarse al equipo destinatario desde el servidor de correo donde hayan quedado almacenados. Por último el RFC-3501 hace lo mismo con protocolo IMAP.

A lo largo del capítulo se describirán de una forma más detenida los protocolos involucadros en el servicio de correo electrónico, SMTP, POP e IMAP. Veremos como se instalan, configuran y se comprueba su funcionamiento bajo GNU/Linux.

5.2. Arquitectura del servicio de correo electrónico

Al igual que en el resto de servicios ofrecidos en red, en primer lugar cabría distinguir entre aplicaciones software cliente (por ejemplo, `kmail`, `Thunderbird` o `Microsoft Outlook Express`) y aplicaciones servidor (por ejemplo, `Postfix`, `SendMail` o `Microsoft Exchange`).

Aplicaciones cliente: Las aplicaciones software cliente, denominadas en el argot informático aplicaciones MUA (Mail User Agent o Agente de Usuario de Correo), son las que están en contacto directo con los usuarios que hacen uso del correo electrónico. Se caracterizan por permitir escribir o leer email, aunque en la actualidad ofrecen al usuario otra serie de opciones como son la gestión

[1] Se recomienda echar un vistazo a la página web `www.rfc-es.org`

de libretas de direcciones, operaciones de filtrado de correo o la distribución en carpetas para mantenerlo organizado.

Aunque este tipo de aplicaciones (MUA) siguen siendo utilizadas, en la actualidad todo servidor de correo ofrece simultáneamente un servicio WebMail, el cual nos permite vía Web llevar a cabo funciones similares a las de un MUA. Este servicio WebMail es posible gracias a la instalación y configuración de un servidor Web (por ejemplo, `apache`) junto con la del servidor de correo.

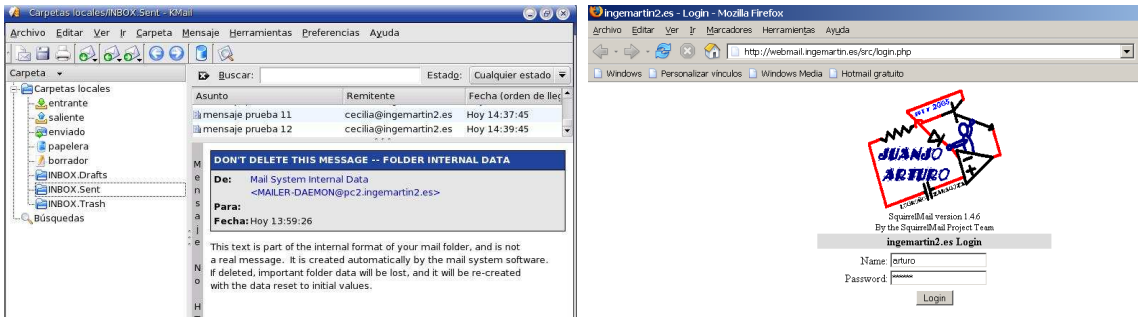


Figura 5.1: Las aplicaciones MUA y el servicio WebMail son los interemediarios entre el usuario y el servidor permitiéndonos tanto enviar como recibir correos.

Aplicaciones servidor: Las aplicaciones de software servidor, denominadas en el argot informático aplicaciones MTA (Mail Transport Agent o Agente de Transporte de Correo), están fuera del alcance de los usuarios finales, y son las encargadas de recoger, almacenar y en su caso reenviar los email con la finalidad de alcanzar el destinatario del correo.

De esta forma, mediante el uso de aplicaciones MUA (cliente) y MTA (servidor) se consigue poner en funcionamiento una infraestructura de correo electrónico completa, tal como se muestra en la figura 5.2

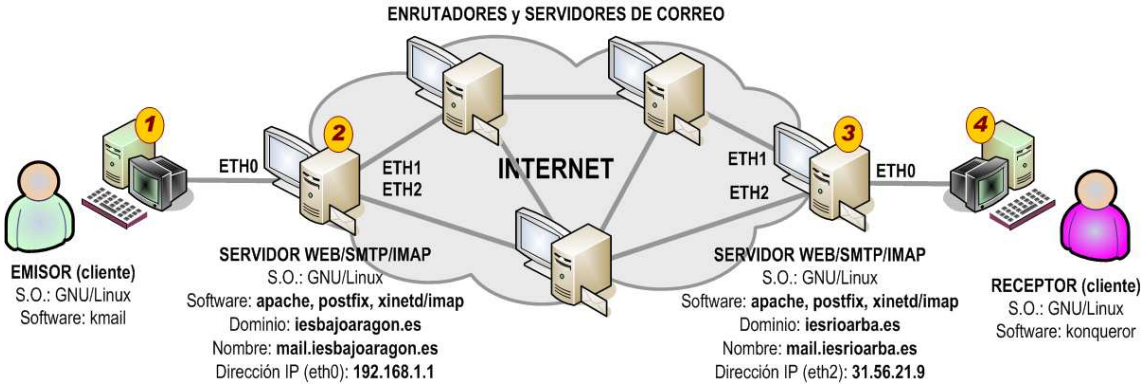


Figura 5.2: Arquitectura del servicio de correo.

5.3. La transmisión y recepción de un e-mail

La forma en que se lleva a cabo el encaminamiento/enrutado de los emails a través de Internet, desde el origen hasta el destino, es muy similar a la utilizada en la transmisión de cualquier otro paquete TCP/IP asociado a cualquier otro servicio ofrecido en Internet (http, ftp, ...). Siguiendo la figura 5.2, en la que `profesor@iesbajoaragon.es` manda un mail a `director@iesbajoaragon.es`, los pasos serían:

1. Una vez que el cliente mostrado en la figura 5.2, llamémosle `profesor@iesbajoaragon.es`, pincha sobre el botón `ENVIAR` de su editor (MUA), el correo sale destinado al equipo servidor de correo electrónico del dominio al que pertenece el usuario[2]. Para poder enviarlo, el equipo cliente necesita conocer la dirección IP asociada a su servidor de correo, y lo consigue mediante una resolución de nombres, que en este caso proporcionaría la dirección IP `192.168.1.1`. De esta forma el e-mail sale del equipo cliente hacia el servidor haciendo uso del protocolo SMTP.

2. Cuando el equipo servidor, `mail.iesbajoaragon.es` (con dirección `192.168.1.1`), recibe el correo procedente del equipo cliente, analiza en primer lugar la dirección del remitente y comprueba que tal usuario, en este caso `profesor`, es uno de los usuarios dados de alta en el equipo servidor con permisos suficientes para hacer uso del servicio de correo electrónico. Tras esta verificación el servidor analiza nuevamente el email y observa la dirección de correo del destinatario, supongamos en este caso que es `director@iesrioarba.es`. Esto lo hace para conocer el dominio del servidor de correo del destinatario, en este caso `iesrioarba.es`. Mediante una solicitud de resolución de nombres[3] el servidor `mail.iesbajoaragon.es` podrá conocer la dirección IP del servidor de correo del destinatario, que en este caso es `mail.iesrioarba.es` (dirección `31.56.21.9`). A continuación, haciendo uso del protocolo SMTP y mediante la ayuda de los router y sus tablas de enrutamiento, el e-mail sale del servidor origen hasta alcanzar su destino.

   Si el e-mail atraviesa a otros servidores de correo antes de alcanzar el servidor destino se dice que hacen la función de `relay` (retransmisor), al limitarse a reenviar los correos por su interfaz de red correspondiente, encaminando estos hacia su destino.

3. Al recibir el equipo servidor `mail.iesrioarba.es` el correo, lo recogerá y analizará la dirección de correo del usuario destinatario, `director@iesrioarba.es`. A continuación comprobará si el usuario `director` está dado de alta en el servicio. En caso de ser así, lo almacenará en el buzón de correo del usuario correspondiente (`director`) y esperará a que este lo solicite.

4. El usuario destinatario (`director`), mediante un navegador Web (en la figura es el `konqueror`) y haciendo uso del servicio WebMail que ofrece su servidor de correo (`mail.iesrioarba.es`) se autentifica en él y abre sus correos.

   Otra opción hubiera sido acceder desde una aplicación MUA a su servidor, y haciendo uso del protocolo POP o IMAP traerse los correos que le pertenezcan.

5.4. Contenido del capítulo

A través de las explicaciones y ejercicios de este capítulo se verá como,

1. Instalar en nuestro equipo el software que lo convertirá en un servidor de correo.
2. Configurar e implementar un servidor SMTP (postfix).
3. Configurar un servidor IMAP (xinetd/imap).
4. Configurar un servidor con servicio WebMail (apache/squirrelmail) y un cliente de correo (kmail).

En definitiva como montar un sistema completo de servicio de correo electrónico.

[2] Observa que en este caso el dominio es `iesbajoaragon.es` y el servidor de correo es `mail.iesbajoaragon.es`
[3] Repasar el capítulo 2 en el que se puede encontrar la explicación del registro de recurso MX.

PRÁCTICA: CONFIGURACIÓN DE UN SERVIDOR DE CORREO

5.5. Instalación del servidor de correo saliente (SMTP): POSTFIX

Comenzaremos la práctica convirtiendo nuestro equipo GNU/Linux en un servidor de correo (MTA) saliente con capacidad de manejo del protocolo SMTP (Simple Mail Transfer Protocol o Protocolo Simple de Transferencia de Correos). Aunque existen muchos paquetes software disponibles en la actualidad que nos permiten desempeñar esta función, en este documento se ha elegido `postfix`.

5.5.1. Extructura interna de POSTFIX

Existen programas que convierten un equipo en un servidor de correo para todos los gustos. Evitando entrar en polémicas sobre la bondad de unos u otros aquí se ha elegido `postfix` porque se caracteriza por una gran organización, potencia y sencillez de configuración, junto con un excelente comportamiento en materias de seguridad. Algunas de las características más importantes de postfix son:

- Los correos recogidos por `postfix` se organizan en cuatro colas de correos:

  1ª Cola: `maildrop` → Cola donde se almacenan los correos generados en el equipo local a través del proceso de envío de correo llamado `sendmail`.

  2ª Cola: `incoming` → Zona donde quedan encolados tanto los correos generados de manera local, como los procedentes del exterior (extranet/Internet).

  3ª Cola: `active` → Los correos son transferidos desde `incoming` hasta `active` donde son encolados con la finalidad de decidir como encaminarlos hacia su destino.

  4ª Cola: `deferred` → A esta cola van a parar aquellos correos que, previamente estando encolados en `active`, no han podido ser encaminados por alguna razón o están pendientes de ello.

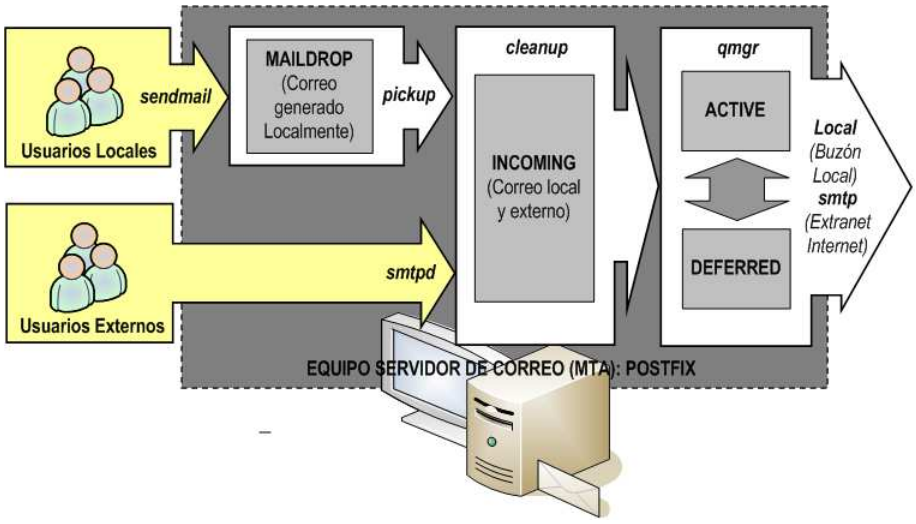


Figura 5.3: Arquitectura de `postfix`.

- Existen seis tablas que facilitan información a los procesos relacionados con `postfix`. La información recogida en ellas permite a los procesos decidir el tratamiento que deben dar a los correos almacenados en las cuatro colas anteriores:

1ª Tabla: `access` → Contiene las reglas que permiten establecer una política de control de acceso de los correos procedentes del exterior (extranet/Internet).

2ª Tabla: `aliases` → Por defecto las cuentas de correo que están dadas de alta en todo equipo servidor de correo tienen el siguiente formato: `<nombre de usuario local>@<nombre dominio>` (por ejemplo, `julian@iespabloserrano.es`). A veces nos puede interesar asociar más de una cuenta de correo a un determinado usuario, para lo cual será necesario generar `alias` (por ejemplo, que el usuario `julian` pueda emitir y recibir mediante la cuenta `jefedepinf@iespabloserrano.es`). Esta tabla nos permite establecer la asociación entre los nombres de los usuarios, y los alias de correo que se les desea asignar.

3ª Tabla: `canonical` → De manera similar a `aliases` nos permite establecer relaciones entre nombres reales y alias, pero con la diferencia de que estas relaciones pueden afectar tanto a usuarios locales como externos.

4ª Tabla: `relocated` → Esta tabla informa a los procesos `postfix` de los posibles cambios que se hayan podido dar en las direcciones de correo de los usuarios.

5ª Tabla: `transport` → Establece la forma en que se encaminarán los correos en función del dominio de destino. En caso de que no aparezca en esta tabla ningún registro, por defecto, los correos se encaminarán como si se tratase de un paquete TCP/IP cualquiera mediante la información suministrada por los DNS a través de sus registros de recurso `MX`.

6ª Tabla: `virtual` → Tabla utilizada en la creación y gestión de dominios virtuales.

- Al diferencia de otros servicios donde un único proceso gestiona el servicio ofrecido, `postfix` se caracteriza por estar formado por un conjunto de ellos, cada uno especializado en una labor concreta. Podrían destacarse los siguientes:

  Proceso Nº1: `sendmail` → Proceso encargado de facilitar la generación de correos en el equipo local. Los correos generados a través de este proceso se encolan en `maildrop`.

  Proceso Nº2: `pickup` → Gestiona los correos encolados en `maildrop` entregándoselos al proceso `cleanup` para que los analice más detenidamente.

  Proceso Nº3: `smtpd` → Este proceso se encarga de recoger los correos procedentes del exterior (extranet/Internet) haciendo uso del protocolo SMTP. La decisión de aceptar o denegar los correos se establece a través de las reglas establecidas en la tabla `access`. En caso de ser aceptados, se entregan al proceso `cleanup` para que los analice más detenidamente.

  Proceso Nº4: `cleanup` → Recoge los correos gestionados por los procesos `pickup` (correos generados localmente) y `smtpd` (correos procedentes del exterior), los analiza y los encola en `incoming`.

  Proceso Nº5: `qmgr` → Recoge los correos encolados en `active` y decide su posterior encaminamiento hacia su destino: local o externo.

5.5.2. Instalación del software POSTFIX

En cuanto a la instalación del paquete software `postfix` en GNU/Linux Mandriva tenemos las dos opciones de costumbre. Desde una consola, terminal o interfaz de línea de comandos haciendo uso de `urpmi`:

```
[root@linux]# urpmi postfix
```

Desde la interfaz gráfica de usuario (GUI) haciendo uso de `rpmdrake`.

```
[root@linux]# rpmdrake &
```

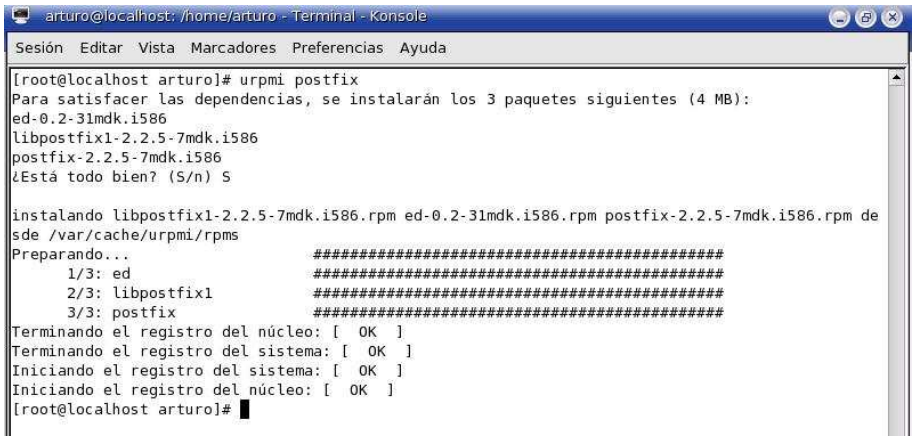


Figura 5.4: Instalación de `postfix` desde la línea de comandos.

5.6. Configuración de POSTFIX

Para configurar y personalizar el funcionamiento de `postfix` es necesario modificar el conjunto de parámetros que pueden ser contenidos en el fichero `/etc/postfix/main.cf`. El comportamiento del servidor de correo dependerá del valor que se les adjudique.

Este fichero, al igual que ocurre tras la instalación de otros servicios, contiene ya una lista de parámetros con valores asignados por defecto. De entre todos, no es conveniente modificar aquellos que están relacionados con la post-instalación[4]. Lo correcto será modificar únicamente los parámetros configurables de usuario[5]. Además de estos, pueden introducirse nuevos parámetros de configuración para una mejor personalización del servicio. Existen alrededor de cuatrocientos parámetros de configuración, lo cual da cuenta de la gran versatilidad que nos ofrece `postfix`.

Por otro lado, no es necesario hacer referencia explícita a la mayor parte de los parámetros de configuración dentro de `/etc/postfix/main.cf`, ya que poseen un valor por defecto que normalmente se ajusta a la mayoría de las configuraciones estándar[6].

En cualquier caso, `main.cf` no es el único archivo ligado a la configuración de `postfix`, dentro del directorio `/etc/postfix/` podemos encontrar un conjunto de ellos, de entre los cuales es interesante conocer los siguientes:

main.cf.defaults → Dentro de este fichero aparece la lista completa de todos los parámetros de configuración de `postfix` junto con su valor por defecto[7]. Es importante conocer estos valores, ya que sólo será necesario hacer referencia a ellos dentro de `main.cf` cuando dicho valor no se corresponda con el valor deseado.

main.cf.dist → Este fichero contiene un prototipo de posible fichero de configuración con extensas explicaciones relativas a la utilidad de cada uno de los parámetros, junto con los posibles valores que se les puede asignar. Es muy recomendable ver su contenido.

aliases → Este fichero contiene la lista con los alias de cuentas de correo asignados a las cuentas de usuario de nuestro sistema.

access → Es un fichero que está pensado para generar a partir de su contenido una tabla o base datos relacionada con el control de acceso al servidor. Su sintaxis es de la siguiente forma:

[4] La cabecera de estos parámetros dentro del fichero `/etc/postfix/main.cf` es "`These are changed by postfix install script`"

[5] La cabecera de estos parámetros dentro del fichero `/etc/postfix/main.cf` es "`User configurable parameters`"

[6] Comprobaremos que con introducir entre cinco y diez parámetros en la mayor parte de los casos suele ser suficiente

[7] Puedes observar que muchos de ellos no tienen valor asignado (`empty`)

```
# Con la finalidad de controlar el acceso de equipos:
# 'direccion IP / nombre dominio'   REJECT|OK
192.168.199.142                     OK
# 'cuenta de correo'                REJECT|OK
amartinr@iesrioarba.es              OK
```

Para generar la tabla o base de datos a partir este fichero se hace uso del comando `postmap`:

```
[root@linux]# postmap /etc/postfix/access
```

`transport` → Este fichero nos permite establecer la forma en que serán encaminados los correos salientes del servidor de correo. Sólo tiene sentido su uso en el caso de que no queramos utilizar la información de los registros de recurso `MX`, suministrada por los DNS. Su sintaxis es de la siguiente forma:

```
# En caso de aceptar el correo destinado a un dominio como local:
# 'nombre del dominio'   #'protocolo':'servidor'
localhost.midominio.es   local:
midominio.es             local:
# Reenviar a "servidorcorreo.otrodominio.es" los correos que lleguen
# para el dominio "otrodominio.es":
# 'nombre del dominio'   'protocolo':'servidor'
otrodominio.es           smtp:servidorcorreo.otrodominio.es
```

En el caso de que el dominio de destino del correo que se reenvía no figure en la tabla se derivará directamente hacia la extranet/Internet como si se tratase de un paquete TCP/IP más.

5.6.1. Parámetros de configuración de POSTFIX

Para poder llevar a cabo una configuración básica del servidor de correo, podríamos destacar los siguientes parámetros de configuración:

Parámetros asociados a la identificación del equipo

- **myhostname**

Se le asigna el nombre de dominio asignado al equipo servidor.
Ejemplo:

```
myhostname = servidormail.ingemartin.es
```

- **mydomain**

Se le asigna el nombre del dominio al que pertenece el equipo servidor.
Ejemplo:

```
mydomain = ingemartin.es
```

- **myorigin**

Identifica la ubicación desde donde son generados los correos. Es decir, si un usuario llamado `javier` edita un correo, la dirección de correo de este remitente figurará como `javier@`*`'myorigin'`*. Por este motivo, el valor de este parámetro se suele corresponder con el nombre del dominio al que pertenece el servidor de correo asignado a la directiva `mydomain`. Para hacer referencia al contenido de algún parámetro ya definido previamente debe precederse del carácter `$` al nombre del parámetro. En caso de no indicar nada, su valor coincidirá con el valor asignado a `$myhostname` provocando que los correos generados tengan el formato: *`'nombre usuario o alias'`*`@$myhostname`.
Ejemplo:

```
myorigin = $mydomain
```

Parámetros relacionados con el correo entrante y saliente

- **mydestination**

El valor de este parámetro informa a `postfix` de los nombres de dominio que serán asociados como locales en lugar de ser reenviados hacia otros servidores de correo. Por defecto nuestro servidor reenviará hacia el exterior todo aquel correo que no vaya dirigido explícitamente al equipo local. Si lo que deseamos es que nuestro servidor recoja y almacene todos los correos dirigidos a otros equipos deberemos indicarlo a través de esta directiva. Pueden especificarse varios nombres separados por espacios en blanco y/o comas.

Ejemplo:

```
mydestination =  localhost $myhostname localhost.$mydomain www.$mydomain
                 ftp.$mydomain mail.$mydomain $mydomain
```

- **mynetworks**

Informa de los equipos y redes (rangos de direcciones IP en formato CIDR) en las que se confía (trusted networks) y se les permite hacer uso del servicio de correo, aceptando su reenvío de correos. Pueden especificarse más de una dirección separando estas por espacios en blanco o por comas:
Ejemplo:

```
mynetworks = 127.0.0.0/8, 192.168.1.0/24, 192.168.2.0/24
```

Si quisieramos que nuestro equipo únicamente reenviara los correos generados localmente (direccion IP del servidor de correo ⇒ `192.168.1.1`):
Ejemplo:

```
mynetworks = 127.0.0.0/8, 192.168.1.1/32
```

.

- **mynetworks_style**

Establece el formato de direcciones IP aceptadas para el parámetro `mynetworks`. Como valores posibles a asignar tenemos `host`, `subnet` (opción por defecto) o `class`.

Asignaremos `host` cuando sólo permitamos el reenvío de correos desde el equipo local, `subnet` cuando el permiso deseamos concederlo a todas las subredes a las que pertenece el equipo (equipos conectados directamente a las interfaces de red) y `class` cuando deseamos especificar direcciones de red de clase A/B/C.
Ejemplo:

```
mynetworks_style = class
```

- **relay_domains**

Especifica los dominios y subdominios bajo los cuales serán aceptados los correos procedentes del exterior. Por defecto, su valor es el contenido de la variable `mydestination`. De esta forma todos los correos externos (extranet e Internet) que tengan como destinatario una dirección de correo asociada a los dominios especificados serán aceptados. Si por ejemplo, sólo deseáramos aceptar los correos asociados a las direcciones de usuario dadas de alta dentro de nuestro dominio escribiríamos:

```
relay_domains = $mydomain
```

- **relayhost**

Por defecto, si no indicamos lo contrario todo aquel correo que no vaya destinado al equipo local es derivado directamente hacia Internet, para que desde allí sea encaminado hacia su correspondiente destino. En el caso de que deseemos derivarlo directamente a otro equipo servidor, para que sea este quien decida que hacer, deberemos asignarle un valor a este parámetro.
Ejemplo:

```
relayhost = [servermail.$domain]
```

El `relayhost` puede especificarse entre corchetes, [ ], o sin ellos, dependiendo de si queremos que el enrutamiento de los correos hacia ese `host` sea estático mediante el uso de las tablas de enrutamiento locales, o sea consultado el registro `MX` en el servidor DNS asociado para `host` especificado.

- **disable_dns_lookups**

Su valor puede ser `yes` o `no` (valor por defecto) dependiendo de si queremos deshabilitar o no las consultas a los registros `MX` de nuestro servidor DNS. Su valor suele depender del tipo de encaminamiento (estático o dinámico), y de si existe o no un `relayhost` al cual derivar los correos.
Ejemplo:

```
disable_dns_lookups = yes
```

- **inet_interfaces**

Especifica las interfaces de red de nuestro equipo servidor de correo a través de las cuales se recibirán correos electrónicos. Por defecto su valor es `all`, aceptando de esta forma todos los correos independientemente de la interfaz por la que entren. En el caso de que solamente estemos interesados en aceptar correos a través de una interfaz de red concreta, podremos indicarlo a través de su dirección IP (o alias). Por ejemplo, si deseamos que `postfix`, únicamente, de servicio local a las cuentas de usuario registradas en la propia máquina para que se envíen correos entre ellas, escribiríamos:

```
inet_interfaces = localhost
```

- **proxy_interfaces**

En aquellos casos en que nuestro servidor de correo se encuentre tras un proxy/nat, debemos especificar su dirección IP externa a través de este parámetro. No tiene ningún valor asignado por defecto, esta vacío (`empty`).
Ejemplo:

```
proxy_interfaces = 31.48.213.167
```

- **mail_spool_directory**

En el caso de que los correos se consideren locales, este parámetro especifica el directorio donde se localizan los buzones locales de los diferentes usuarios del sistema. El valor por defecto es `/var/mail`.
Ejemplo:

```
mail_spool_directory = /var/spool/mail
```

.

Parámetros relacionados con restricciones del servicio

- **smtp_connect_timeout**

Determina el tiempo límite permitido para que un cliente complete una conexión TCP al puerto SMTP (25) del servidor. Por defecto su valor es de 30 segundos (30s). Este tiempo se puede especificar en diversas unidades: segundos (s), minutos (m), horas (h), días (d) o semanas (w).
Ejemplo:

```
smtp_connect_timeout = 2m
```

- **smtpd_client_restrictions**

Nos permite de manera opcional establecer restricciones a los clientes SMTP. Puede establecerse una lista de restricciones separadas por comas o espacios en blanco. No tiene ningún valor asignado por defecto (`empty`) y entre las posibles restricciones existentes cabe destacar:

permit_mynetworks → Sólo aceptará peticiones SMTP de aquellos clientes cuyas IP aparezcan en el parámetro `$mynetworks`.

reject_unknown_client → Serán rechazadas aquellas peticiones procedentes de equipos desconocidos. Para decidir si se trata de un equipo desconocido o no, el servidor de correo solicitará una resolución inversa de la dirección IP del cliente. Si el cliente no tiene asignado un nombre cualificado, su correo no será aceptado.

Por ejemplo, si el equipo `192.168.199.142` no estuviera registrado en el fichero de resolución inversa del DNS, e intentara hacer uso del servicio de correo desde un cliente (por ejemplo kmail) le aparecería el mensaje de error mostrado en la figura 5.5.a)

check_client_access → Haciendo uso de la tabla `/etc/postfix/access` se nos permite establecer las direcciones IP, nombres de dominio, direcciones de red o direcciones de correo de los clientes a los que se les restringirá el acceso al servidor. En el caso de hacer uso de un fichero indexado, la sintaxis del mismo es:

```
'direccion IP'|'nombre dominio'|'dirección correo' OK|REJECT
```

Ejemplo 1:

```
smtpd_client_restrictions = permit_mynetworks reject_unknown_client
```

Ejemplo 2:

```
smtpd_client_restrictions = check_client_access hash:/etc/postfix/access
```

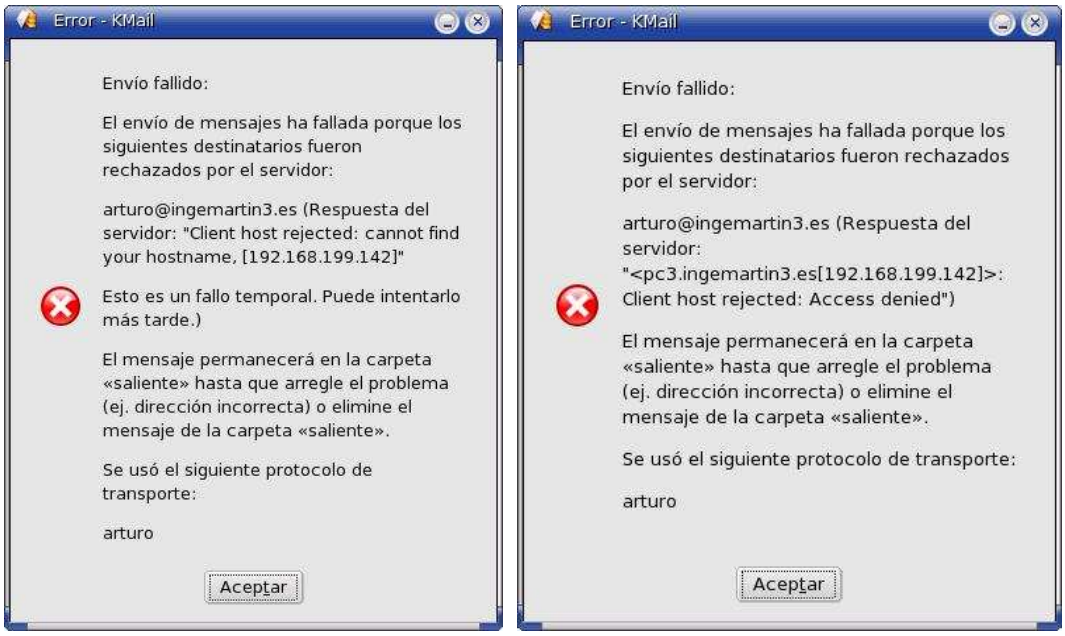
Si en el fichero `/etc/postfix/access` existiese la siguiente línea:

```
192.168.199.142 REJECT
```

Al intentar el equipo `192.168.199.142` hacer uso del servicio de correo desde un cliente le aparecería el correo de error mostrado en la figura 5.5.b)

- **smtpd_recipient_restrictions**

El valor por defecto de este parámetro es: `permit_mynetworks`, `reject_unauth_destination`. Esto establece que nuestro servidor sólo aceptará correos de equipos clientes cuya dirección IP esté incluida en el parámetro `mynetworks`, o figure como destinatario de correo alguno



a) b)

Figura 5.5: a) Mensaje de error producido en caso de que el cliente no tenga un nombre cualificado. b) Mensaje de error producido en caso de que el equipo cliente no sea fiable.

de los contemplados en los parámetros `relay_domains`, `inet_interfaces`, `mydestination`, `virtual_alias_domains` o `virtual_mailbox_domains`.

En el caso de que queramos añadir restricciones se aconseja mantener las opciones por defecto y añadir a estas las nuevas. Entre las restricciones posibles destacaríamos:

check_sender_access → Esta restricción permite establecer a qué cuentas de correo se les permite enviar correo a través de nuestro servidor.

reject_unknown_sender_domain → Rechaza aquellos correos asociados a nombres de dominio desconocidos (no tienen asignado un registro PTR en el servidor DNS).

Ejemplo:

```
smtpd_recipient_restrictions  = permit_mynetworks reject_unauth_destination
                                check_sender_access hash:/etc/postfix/usuarios
```

.

- **smtpd_sender_restrictions**

No tinen ningún valor asignado por defecto, lo que significa que en principio no hay restricciones. Es el que determina desde qué equipos clientes y a través de qué direcciones de correo pueden enviarse correos a nuestro servidor como destino. Entre la lista de restricciones posibles a especificar se encuentran,

permit_mynetworks → Acepta los correos generados desde los equipos especificados en el parámetro `mynetworks`.

reject_unknown_sender_domain → Rechaza todos aquellos correos con un nombre de

dominio no cualificado.

Puede especificarse directamente una lista (/etc/postfix/access) con los usuarios a restringir.

Ejemplo:

```
smtpd_sender_restrictions = hash:/etc/postfix/access, permit_mynetworks
```

- **smtpd_client_connection_count_limit**

Limita el número de conexiones SMTP que pueden establecerse de manera simultánea en nuestro servidor. Su valor por defecto es 50. Para deshabilitar este límite deberemos asignarle el valor 0.
Ejemplo:

```
smtpd_client_connection_count_limit = 0
```

.

- **authorized_submit_users**

Establece los usuarios que están autorizados para enviar correos. Por defecto, no existen restricciones pudiendo hacer uso del servicio cualquiera (`anyone`). En el caso de querer restringir el servicio a una lista de usuarios, es posible especificar sus nombres separados por espacios o comas.
Ejemplo:

```
authorized_submit_users = arturo, juanjo
```

Si la lista de usuarios resulta ser excesivamente grande es recomendable indicar como valor el nombre de un fichero que contenga dicha lista (tabla).

Otros parámetros

- **alias_database** y **alias_maps**

Por defecto no tiene asignado ningún valor. En principio, todo usuario del sistema GNU/Linux que esta dado de alta en el equipo servidor de correo tiene habilitada una cuenta de correo de la forma:

```
'nombre usuario'@$myorigin
```

En el supuesto de que nos interese crear nuevas cuentas de correo asociadas a los usuarios del sistema deberemos crear alias de su cuenta de correo nativa.

Para poder crear esos alias existe el fichero (tabla) `/etc/postfix/aliases` cuya sintaxis es: *'nombre alias': 'nombre usuario'*. Según esto, si en nuestro sistema (por ejemplo,

`ingemartin.es`) está dado de alta un usuario llamado `francisco` (`francisco@ingemartin.es`) y deseamos configurar una cuenta de correo para él llamada `paco77@ingemartin.es` deberíamos editar el fichero de alias y añadir la siguiente línea: `paco77: francisco`.

Por último, para que estos alias sean tenidos en cuenta deberemos ejecutar el comando `newaliases`; este comando da forma al contenido del fichero para poder indexar los alias. Tras su ejecución se deben incluir en el fichero de configuración `/etc/postfix/main.cf` los parámetros `alias_database` y `alias_maps` informando de la ruta donde se localiza el fichero de alias.
Ejemplo:

```
alias_database = /etc/postfix/aliases
alias_maps = /etc/postfix/aliases
```

- **transport_maps**

Este parámetro especifica la ruta donde se localiza la base de datos o tabla que será utilizada para tomar decisiones relacionadas con el envío de los correos salientes. Por defecto no tiene ningún valor asignado, esto significa que en el caso de no especificar nada los correos se emitirán como si se tratasen de un paquete TCP/IP más (se recomienda ver la sección 5.2 como recordatorio). Si por el contrario deseamos desviar determinados correos hacia un servidor específico será necesario hacer uso de este parámetro.

Para crear la base de datos será necesario editar en primer lugar un fichero plano, y luego transformarlo mediante el comando `postmap`. Ese fichero plano se suele corresponder con `/etc/postfix/transport` (ver explicación en la página 207) y su sintaxis es:

```
'nombre de dominio' 'protocolo':'servidor'
```

Ejemplo:

```
transport_maps = /etc/postfix/transport
```

.

- **smtpd_banner**

El valor por defecto que toma este parámetro es `$myhostname ESMTP $mail_name`. Establece la cabecerá que acompañara al código de estado 220. Puede ser cualquier texto, aunque el correo deberá comenzar obligatoriamente por `$myhostname`, al ser esta, una especificación del protocolo SMTP.
Ejemplo:

```
smtpd_banner = $myhostname Servidor WebMail
```

.

Una vez que se hayan realizado cambios en el archivo de configuración de `postfix` tendremos dos opciones, recargar `postfix` (postfix reload) o reiniciar[8] el servicio. De las dos opciones,

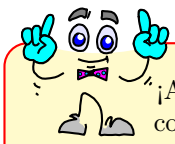
[8] Para reiniciar escribiríamos:

recargar la configuración de `postfix` es lo más aconsejable debido a que el servicio no es parado en ningún momento:

```
[root@linux]# postfix reload
```

5.7. Instalación del servidor de correo entrante (POP/IMAP)

Al instalar `postfix` convertimos nuestro equipo en un servidor de correo (MTA) con capacidad de gestionar los correos salientes. Una vez que el servidor MTA advierte que el destinatario se corresponde con una dirección de correo perteneciente al dominio al cual da servicio, lo almacena a la espera de que el usuario correspondiente lo solicite a través de su aplicación cliente (MUA). Como ya se ha visto, el protocolo utilizado para enviar los correos al servidor es el SMTP, pero como ya fue comentado en la sección 5.2 (página 203), el protocolo utilizado por nuestra MUA para recoger los correos almacenados por `postfix` en el servidor no es ese, sino que debe ser POP o IMAP. Por este motivo, el equipo que haga de servidor de correo deberá tener habilitados o bien un servicio POP o un IMAP para permitir que los usuarios puedan consultar los correos que tengan almacenados. En ambos casos será necesario instalar el programa `xinetd`, que se encarga de gestionar ambos servicios.



¡Aclaración! Las aplicaciones MUA nos permiten tanto editar y enviar correos, como recogerlos del servidor cuando nosotros seamos los que figuramos como destinatarios. Una vez que ya hemos editado el correo, para poder enviarlo a nuestro servidor, y que este lo haga llegar a su destinatario, necesitamos hacer uso de SMTP (protocolo de correo saliente). Por el contrario, en el caso de que solicitemos recoger nuestros correos almacenados en el servidor necesitaremos hacer uso del protocolo POP o IMAP (protocolos de correo entrante). Esto significa que nuestra MUA deberá hacer uso de diferentes protocolos (SMTP/POP/IMAP) dependiendo de si envía o recibe correos.

Cuando el servicio elegido sea IMAP únicamente se deberá instalar el paquete `imap` y será el propio sistema quien detectará su dependencia con `xinetd` instalándolo igualmente (ver figura 5.6):

```
[root@linux]# urpmi imap
```

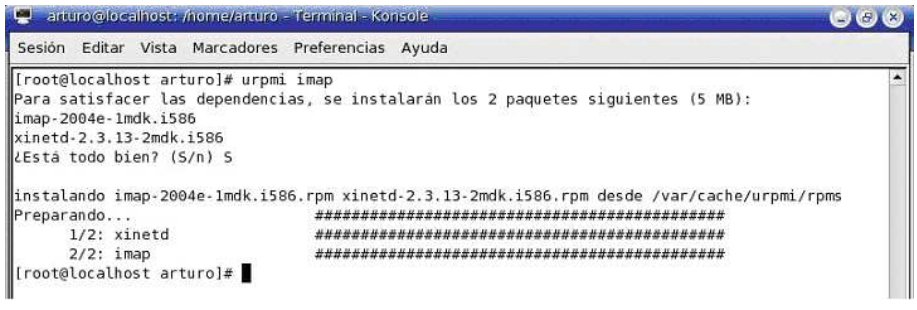


Figura 5.6: Instalación del servicio imap desde la línea de comandos.

Tanto el protocolo POP como el IMAP están pensados para leer los correos electrónicos almacenados en el servidor utilizando una aplicación cliente (MUA) o vía Web (WebMail) , pero presentan algunas diferencias:

```
[root@linux]# service postfix restart
```

- El protocolo IMAP es posterior al protocolo POP. Esto nos sugirie que el protocolo IMAP recoge todas las ventajas del protocolo POP y trata de corregir algunos de sus inconvenientes[9].
- El protocolo POP extrae los correos del servidor y los almacena en el equipo cliente, obligando al usuario a tener que consultarlos desde un único equipo. En cambio el protocolo IMAP sólo los consulta, permaneciendo estos en el servidor hasta que sean eliminados. Esto permitirá que los correos puedan ser consultados desde diferentes ubicaciones, ya que se encuentran centralizados en el servidor.
- El protocolo IMAP permite compartir un buzón (mailbox), al encontrarse este almacenado, entre diferentes usuarios (diferentes equipos clientes). Así, deja realizar consultas simultaneas e interactivas sobre el correo (directorios públicos). Mediante POP esto es imposible, al recoger el correo en el equipo local.
- El protocolo IMAP proporciona algoritmos de búsqueda usando criterios previamente establecidos. Con esto, el usuario puede leer solamente aquellos correos en los que esta interesado. En cambio a través de POP la única opción es descargar todo el correo, y después hacer la selección en el equipo local (cliente).
- El protocolo IMAP, por las características que acaban de citarse, es el utilizado en la implementación del servicio de correo vía Web (Webmail).

Sopesando las características de ambos protocolos, en este libro se ha optado por elegir IMAP para efectuar la mayoría de las explicaciones y configuraciones. A pesar de esta elección, posteriormente se dará una breve explicación sobre la instalación de POP.

5.7.1. Configuración de xinetd/IMAP

Una vez instalado el paquete IMAP (ver figura 5.6) debe hacerse una sencilla configuración para habilitar el servicio, que por defecto se encuentra deshabilitado. Para lo cual, debe editarse el fichero de configuración del servicio IMAP, `/etc/xinetd.d/imap` y modificar el parámetro `disable` cambiando el `yes` por un `no`.

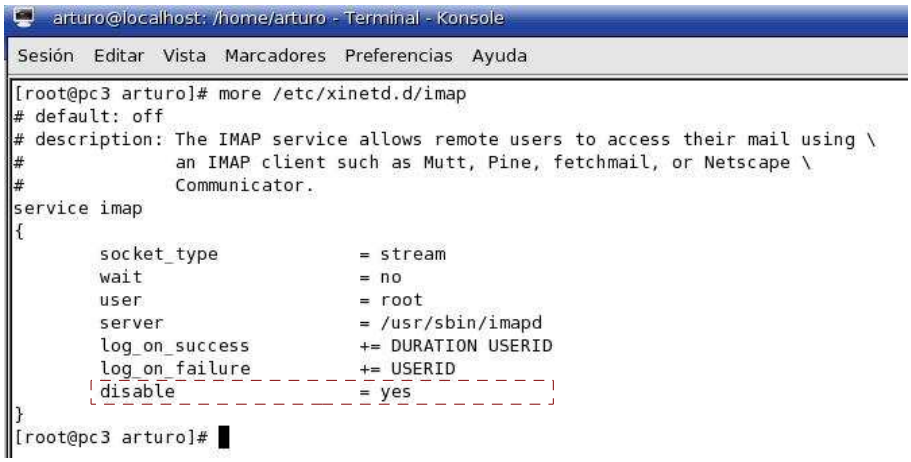


Figura 5.7: Post-configuración del servicio IMAP. Para habilitar el servicio es necesario modificar la opción `disable` sustituyendo el `"yes"` por un `"no"`.

Para que los cambios surtan efecto es necesario reiniciar el servicio `xinetd`:

```
[root@linux]# service xinetd restart
```

[9]Esta apreciación está hecha desde un punto de vista general, ya que en alguna situación concreta podría suceder que el protocolo POP se adaptase mejor que el IMAP a nuestras necesidades.

Los servicios que están activos pueden chequearse utilizando el comando `chkconfig`. Así para comprobar si IMAP está realmente activo se puede ejecutar ese comando junto con la opción `--list`. Esto, listará el conjunto de servicios[10] ofrecidos por nuestro equipo, indicando su estado en función del modo de arranque (init 1, 2, 3, 4 ó 5):

```
[root@linux]# chkconfig --list
```

5.8. Comprobación del funcionamiento del servicio de correo

Para practicar todo lo visto a lo largo del capítulo realizaremos el montaje en red mostrado en la figura 5.8 mediante la ayuda de VMware (ver el apéndice A para obtener información de cómo hacerlo):

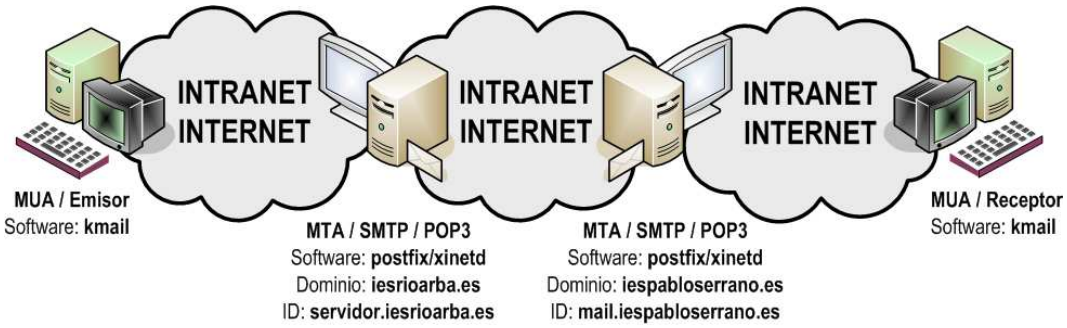


Figura 5.8: Estructura del servicio de correo.

Para facilitar su implementación haremos que en un mismo equipo corran tanto el software cliente (MUA/`kmail`) como el del servidor (MTA/`postfix/xinetd`) reduciendo la configuración en red a dos equipos. Ver la figura 5.9.

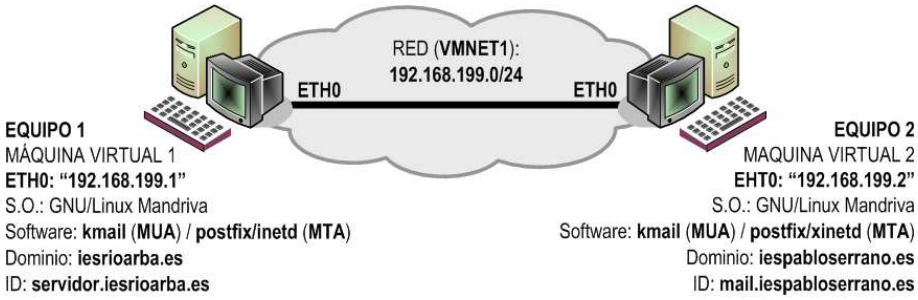


Figura 5.9: Implementación del servicio de correo a través de VMware.

Los pasos que seguiremos para ello son:

1. Configuraremos el servicio `named` (bind) del equipo servidor DNS. Cualquiera de los dos equipos puede desempeñar esas funciones, en este caso se ha elegido que el DNS sea `servidor.iesrioarba.es`.

   Tras editar el fichero `/etc/named.conf`, deberán inscribirse las zonas correspondientes a los dos nombres de dominio mostrados en la figura 5.9 (`iesrioarba.es` e `iespabloserrano.es`) y además la correspondiente zona de resolución inversa:

[10] Para visualizar únicamente el asociado a IMAP escribiríamos:

```
[root@linux]# chkconfig --list | grep imap
```

```
####### CONTENIDO DEL FICHERO /etc/named.conf #######
options{
          directory /var/named";
};
zone "iesrioarba.es"in {
          type master;
          file "maestra.iesrioarba.es";
};
zone "iespabloserrano.es"in {
          type master;
          file "maestra.iespabloserrano.es";
};
zone "199.168.192.in-addr.arpa"in {
          type master;
          file "inversa.199.168.192.in-addr.arpa";
};
```

A continuación editaremos los ficheros de zona *directa* correpondientes a los dos nombres de dominio[11]:

```
iesrioarba.es.   IN SOA    servidor                root.localhost.
                                                   (1309200601;
                                                   21600;
                                                   10800;
                                                   604800;
                                                   21600; )
                 IN NS     servidor
                 IN MX 10  servidor.iesrioarba.es.
servidor         IN A      192.168.199.1
iesrioarba.es.   IN A      192.168.199.1
```

Observa que es fundamental que aparezca el registro `MX` en los ficheros de zona[12], ya que los servidores de correo consultan su valor para determinar el equipo que dentro de un dominio se encarga de recoger los correos electrónicos dirigidos a sus usuarios. Olvidarse del registro `MX` provocaría que el servidor no sabría a qué equipo destino reenviar los correos. Por su parte el contenido del fichero de zona de resolución inversa `/var/named/inversa.199.168.192.in-addr.arpa` sería similar a:

```
@  IN SOA  servidor                 root.localhost.
                                    (1309200601;
                                    21600;
                                    10800;
                                    604800;
                                    21600; )
   IN NS   servidor
1  IN PTR  servidor.iesrioarba.es.
2  IN PTR  mail.iespabloserrano.es.
```

Una vez se haya configurado el servicio `named`, se sugiere chequear[13] el servicio mediante `named-checkconf` y las zonas mediante el uso de `named-checkzone`. Si todo es correcto sólo queda reiniciar el servicio:

[11] Sólo se describe el contenido del fichero de zona `/var/named/maestra.iesrioarba.es` ya que el asociado a `/var/named/maestra.iespabloserrano.es` es similar intercambiando nombres de dominio y direcciones IP.

[12] Ver para más detalles el capítulo 2, apartado 2.7

[13] Ver el capítulo 2, secciones 2.8.2 y 2.8.4

```
[root@linux]   named-checkconf /etc/named.conf
 [root@linux]  named-checkzone iesrioarba.es /var/named/maestra.iesrioarba.es
[root@linux]   named-checkzone iespabloserrano.es
                                    /var/named/maestra.iespabloserrano.es
[root@linux]   named-checkzone 199.168.192.in-addr.arpa
                                    /var/named/inversa.199.168.192.in-addr.arpa
[root@linux]   service named restart
```

Por último se debe comprobar que en los equipos, tanto en los clientes como en los servidores, figura como servidor DNS la dirección IP del equipo que acabáis de configurar el servicio named (bind). Recuerda que esto se indica en el fichero `/etc/resolv.conf`.

2. A continuación configuraremos el equipo servidor de correo encargado de recoger el correo escrito por el usuario emisor (equipo 1). Para ello, instalaremos `postfix` y añadiremos los parámetros correspondientes, respetando el valor asignado a los parámetros relacionados con el sistema `postfix` tras la instalación.

```
[root@linux]# urpmi postfix
```

En el siguiente script, a los parámetros añadidos o modificados en `main.cf` se les ha colocado el comentario # modificado con el fin de hacerlo más legible.

```
####### CONTENIDO DE /ect/postfix/main.cf #######

# These are changed by postfix install script
readme_directory = /usr/share/doc/postfix-2.2.5/README_FILES
html_directory = /usr/share/doc/postfix-2.2.5/html
sendmail_path = /usr/sbin/sendmail.postfix
setgid_group = postdrop
command_directory = /usr/sbin
manpage_directory = /usr/share/man
daemon_directory = /usr/lib/postfix
newaliases_path = /usr/bin/newaliases
mailq_path = /usr/bin/mailq
queue_directory = /var/spool/postfix
mail_owner = postfix
# User configurable parameters
inet_interfaces = all # modificado al valor por defecto.
mynetworks_style = subnet # modificado al valor por defecto.
delay_warning_time = 4h
smtpd_banner = $myhostname ESMTP $mail_name ($mail_version) (Mandriva Linux)
unknown_local_recipient_reject_code = 450
smtp-filter_destination_concurrency_limit = 2
lmtp-filter_destination_concurrency_limit = 2
smtpd_sasl_path = /etc/postfix/sasl:/usr/lib/sasl2
# Parámetros relacionados con la Identificación del equipo
myhostname = servidor.iesrioarba.es # modificado
mydomain = iesrioarba.es # modificado
myorigin = iesrioarba.es # modificado
# Control de los correos salientes y entrantes
mydestination = localhost localhost.$mydomain $myhostname $mydomain # modif.
mynetworks = 127.0.0.0/8, 192.168.199.0/24 # modificado
```

A continuación debe recargarse la nueva configuración del servicio `postfix` (o reiniciaremos el servicio tal como se explicó en la sección 5.6, página 215):

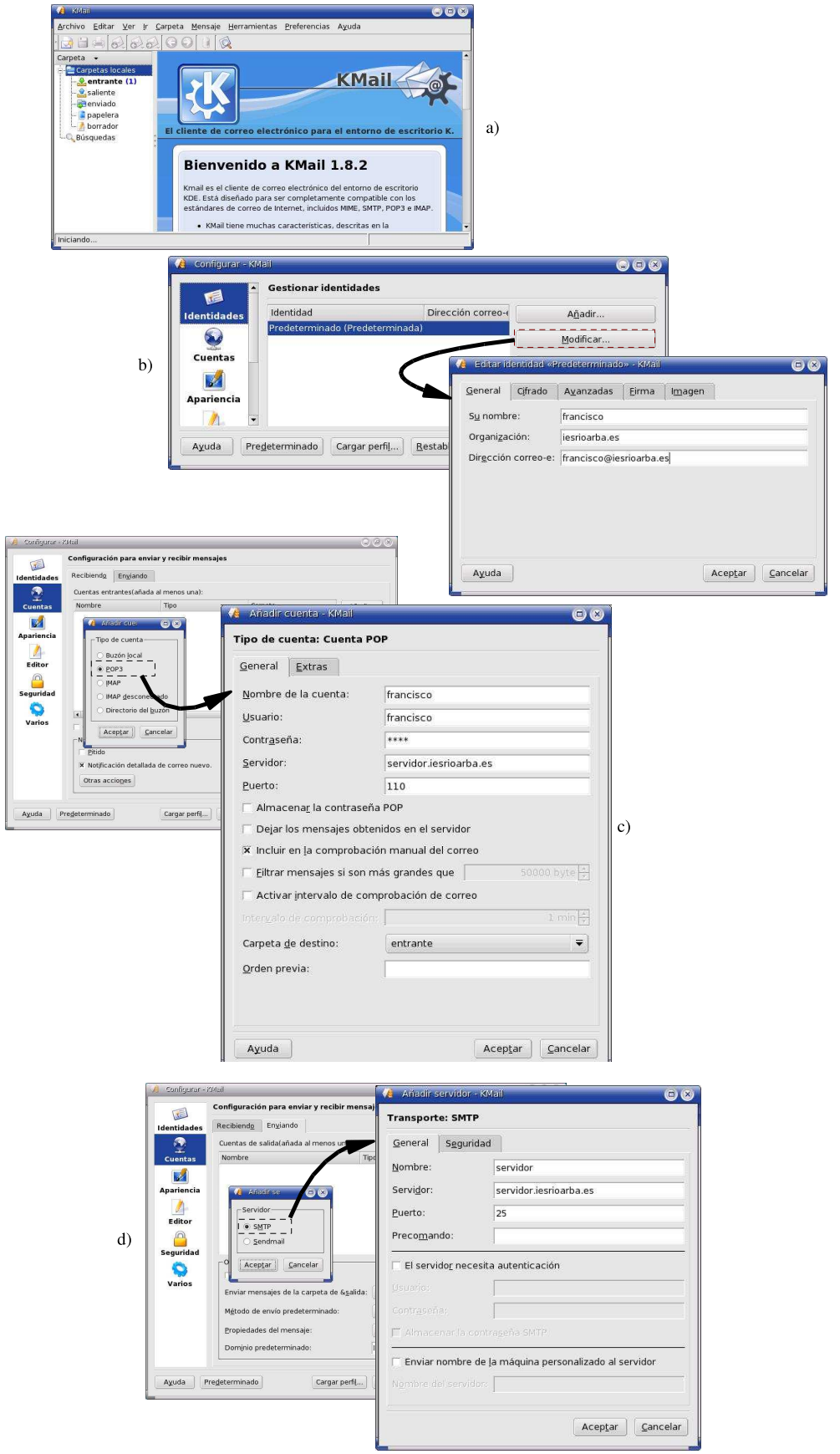


Figura 5.10: a) Aplicación cliente de correo (MUA) para enviar y recibir correos: kmail. b) Configuración de kmail: identificación. c) Configuración de kmail: servidor de correo entrante. d) Configuración de kmail: servidor de correo saliente.

```
[root@linux]# postfix reload
```

3. Se debe configurar, en los equipos servidores de correo que así estipulemos, el servicio POP3 (`xinetd/ipop3`). Bastará con instalar el paquete software `xinetd` y comprobar mediante `chkconfig` que el servicio POP (`ipop3`) está activo:

```
[root@linux] urpmi xinetd
[root@linux] service xinetd start
[root@linux] chkconfig --list | grep pop
```

4. A continuación configuraremos en los propios equipos servidores un software cliente de correo (MUA) que nos permita tanto escribir correos como recibirlos. De entre los existentes, por sencillez en su configuración y manejo, se ha seleccionado `kmail` (ver figura 5.10.a)).

   Crearemos las cuentas de usuario en los dos sistemas que se van a cartear (por ejemplo, `francisco` en el equipo 1, y `elena` en el equipo 2). Tras ello, se iniciará el sistema con las nuevas cuentas de usuario creadas. Para configurar los dos clientes (MUA) ejecutaremos el programa `kmail`[14]. Iremos a su menú `preferencias` y seleccionaremos la opción `configurar kmail`. Desde la ventana que nos aparezca podremos configurar nuestra identidad y nuestros servidores de correo saliente (`SMTP/postfix`) y entrante (`POP3/xinetd/ipop3`).

   En primer lugar, ver la figura 5.10.b), comenzaremos configurando la identidad del usuario. A continuación y como se puede ver en la figura 5.10.c), tras seleccionar la opción `cuentas` y posteriormente la pestaña `Recibiendo`, pasaremos a configurar el servidor de correo entrante (POP3) pinchando sobre el botón `Añadir`.

   Terminaremos la configuración de kmail indicando quién será nuestro servidor de correo saliente (SMTP) desde la pestaña `Enviando`, pinchando sobre el botón `Añadir`; esto se muestra en la figura 5.10.d).

   Los mismos pasos seguidos para la configuración realizada con la cuenta de `francisco` en el equipo 1, deberán darse en el equipo 2 para configurar la cuenta de `elena`.

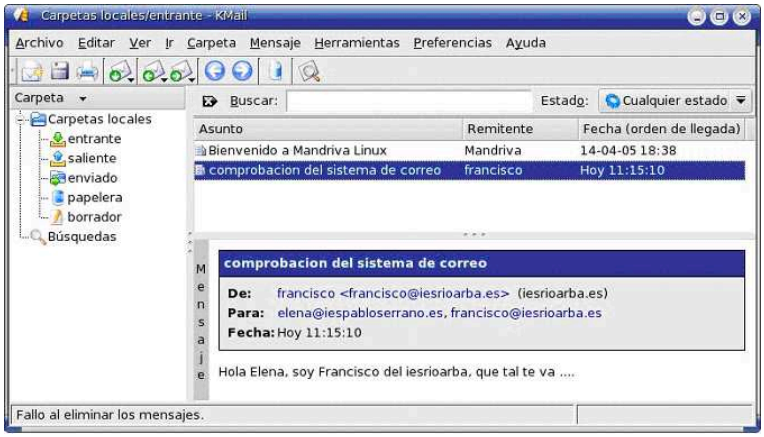


Figura 5.11: Comprobación de envío y recepción de correos mediante kmail.

5. Ahora tan sólo queda probar que el servicio funciona correctamente. Esto puede hacerse mandando un correo desde el equipo 1 (remitente: `francisco@iesrioarba.es`) para recibirlo

[14] Una posibilidad es que `francisco` escriba en la línea de comandos:

```
[francisco@linux] kmail &
```

en el equipo 2 (destinatario: `elena@iespabloserrano.es`). Cómo se escriben y reciben correos desde `kmail` se da por sabido.

5.9. Correo vía web. Instalación y configuración de SquirrelMail

En este último punto del capítulo se va a describir la forma de configurar un servidor WebMail. Para el desarrollo de este apartado es necesario que la configuración realizada en el apartado anterior haya sido exitosa.

El servicio WebMail no es más que una simbiosis entre nuestro servidor de correo (SMTP/-POP3/IMAP) y nuestro servidor web (APACHE). La implementación en red que habrá que realizar en este caso será similar a la del ejercicio anterior y que puede verse en la figura 5.12. La única diferencia es que ahora se va a prescindir del cliente de correo (MUA/kmail) debido a que las acciones (enviar y recibir correos) se efectuarán a través de un simple navegador Web.

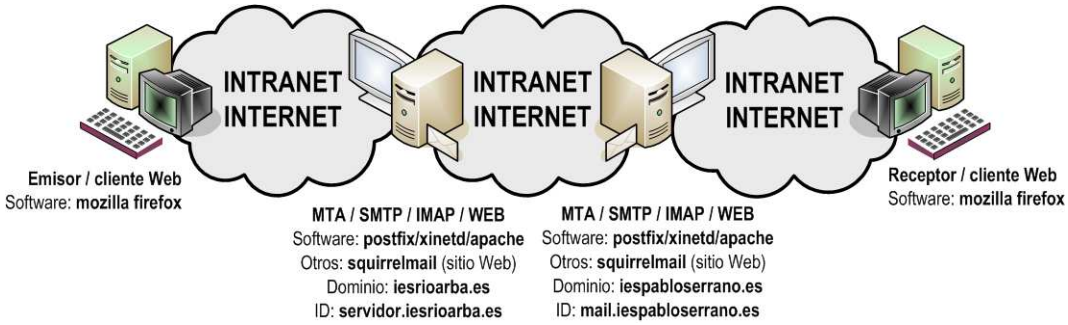


Figura 5.12: Estructura e implementación del servicio Webmail.

Los pasos que deberemos seguir para una correcta configuración son:

1. Asegurarnos de que los servidores de correo tienen habilitados los servicios SMTP (`postfix`) e IMAP (xinetd/imap). En el caso de que la comprobación del apartado anterior haya resultado exitosa, el servicio `postfix` no será necesario modificarlo (a no ser que se desee añadir alguna característica nueva) y tan sólo deberemos asegurarnos de tener habilitado el servicio IMAP[15]:

```
[root@linux]# chkconfig --list | grep imap
```

2. Necesitaremos registrar en el DNS (equipo 1) el nombre de dominio que deseamos asignar al sitio WebMail (por ejemplo, `webmail.iesrioarba.es` y `webmail.iespabloserrrano.es`) modificando los ficheros de zona de cada uno de los dominios:

```
iesrioarba.es.          IN SOA    servidor                root.localhost.
                                                          (1309200601;
                                                          21600;
                                                          10800;
                                                          604800;
                                                          21600; )
                        IN NS     servidor
                        IN MX 10  servidor.iesrioarba.es.
servidor                IN A      192.168.199.1
iesrioarba.es.          IN A      192.168.199.1
webmail.iesrioarba.es.  IN A      192.168.199.1
```

[15]Si es necesario repasa los apartados 5.7 y 5.7.1 (página 216).

Observa que el equipo 1 desarrolla cuatro servicios: DNS, SMTP, IMAP y HTTP.

3. Instalar y configurar apache para alojar el sitio Web que hará de interfaz entre el cliente Web del usuario y el servidor de correo: `SquirrelMail`. Como el sitio web `SquirrelMail` está creado en código PHP, para que pueda ser servido desde apache será necesario agregar el módulo `apache2-mod_php`.

```
[root@linux]  urpmi apache2
[root@linux]  urpmi apache2-mod_php
```

A continuación editaremos el fichero de configuración de apache `/etc/httpd/conf/httpd.conf` y crearemos un nuevo host virtual basado en nombre para alojar a `SquirrelMail`.

```
#Contenido del fichero de configuración del servidor apache:
#/etc/httpd/conf/httpd2.conf (Mandriva 2005)
#1er BLOQUE
ServerRoot          /etc/httpd/2.0
PidFile             /var/run/httpd.pid
ErrorLog            logs/error_log
LogLevel            warn
#Creamos un nuevo host virtual para el sitio Web SquirrelMail bajo la
# IP del servidor
NameVirtualHost     192.168.199.1          #IP del servidor de correo
<VirtualHost 192.168.199.1>
     DocumentRoot   /var/www/html/webmail   #Raíz del sitio SquirrelMail
     ServerName     webmail.iesrioarba.es   #Nombre del sitio web a servir
</VirtualHost>
...
#El resto de parámetros se pueden dejar a su valor por defecto
```

Para que los cambios surtan efecto será necesario reiniciar el servicio `httpd`:

```
[root@linux]# service httpd restart
```

4. A continuación deberemos descargarnos `SquirrelMail` (formato `tar` o `tar.gz` preferiblemente) y desempaquetamos su contenido en `/var/www/html`. Si se encuentra en `tar.gz` la secuencia de comandos[16] sería:

```
[root@linux]  cp squirrelmail.tar.gz /var/www/html
[root@linux]  cd /var/www/html
[root@linux]  tar -xvzf squirrelmail.tar.gz
```

Al desempaquetar se habrá creado una carpeta con el nombre y versión del sitio Web `SquirrelMail` descargado, por ejemplo `squirrelmail-1.4.6`. Para ser consecuentes con el valor dado al parámetro `DocumentRoot` especificado en `/etc/httpd/conf/httpd.conf` deberemos renombrar dicha carpeta por `webmail`:

```
[root@linux]# mv squirrelmail-1.4.6 webmail
```

[16] Si se encuentra en `.tar` el último de los comandos debería cambiarse por:

```
[root@linux]# tar -xvf squirrelmail.tar
```

Para albergar los datos temporales que manipule el servicio WebMail `SquirrelMail` crearemos una última carpeta que quede únicamente al alcance de apache:

```
[root@linux]  mkdir -p /var/squirrelmail/attach
[root@linux]  chmod -R 700 /var/squirrelmail
```

Por último, si tenemos en cuenta que el usuario encargado de ejecutar y llevar a cabo todas las acciones del servicio Webmail es apache, deberemos cambiar de propietario en todos los directorios anteriores:

```
[root@linux]  chown -R apache.apache /var/www/html
[root@linux]  chown -R apache.apache /var/squirrelmail
```

5. Una vez que ya hemos terminado de configurar todo lo referente a apache, pasaremos a configurar `SquirrelMail`. Para ello ejecutaremos el script (lenguaje `Perl`) `/var/www/html/webmail/configure`:

```
[root@linux]# /var/www/html/webmail/configure
```

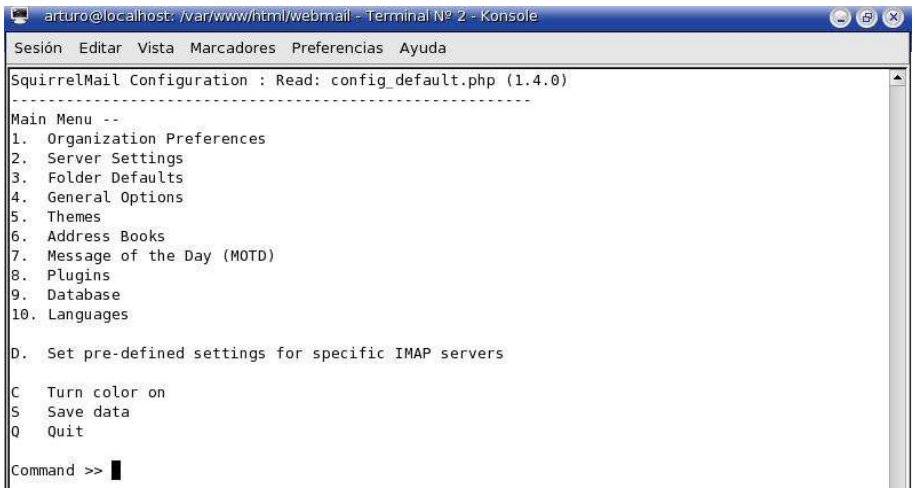


Figura 5.13: Configuración de `SquirrelMail`.

Entre todas las opciones de configuración que nos ofrece el menú cabría destacar las siguientes:

- `Organization Preferences`: Nos permite configurar la apariencia de la página de inicio del servicio WebMail: logotipo (qué imagen y con qué dimensiones), nombre de nuestra organización, ... (ver la figura 5.14.a)).
- `Server Settings`: Nos permite especificar el nombre del dominio al que da servicio el servidor de correo, y los protocolos utilizados (ver la figura 5.14.b)). Observa que tiene el mismo significado que el parámetro `myorigin` de `postfix`.
- `Folder Defaults`: Entre otras cosas nos permite personalizar los nombres de las carpetas (buzones) utilizadas para organizarnos los correos electrónicos (ver la figura 5.14.c)).

En cuanto al resto de opciones del menú no presentan un interés especial. Una vez hayamos terminado de configurarlo, guardaremos los cambios (opción `s`) y saldremos (opción `q`). Tal como se podrá observar, al salir de la configuración de `SquirrelMail`, se nos sugiere que testeemos la nueva configuración y comprobemos que todo es correcto. Para ello debemos acceder vía Web al directorio virtual `src` dentro del servicio Webmail y ejecutar el script PHP `configtest.php` tal y como se muestra en la figura 5.15.

a)

```
arturo@localhost: /var/www/html/webmail - Terminal Nº 2 - Konsole
Sesión  Editar  Vista  Marcadores  Preferencias  Ayuda
SquirrelMail Configuration : Read: config_default.php (1.4.0)
---------------------------------------------------------
Organization Preferences
1.  Organization Name      : IES Rio Arba
2.  Organization Logo      : ../images/aula4.jpg
3.  Org. Logo Width/Height : (400/400)
4.  Organization Title     : Servicio WebMail IES Rio Arba (TAUSTE)
5.  Signout Page           :
6.  Top Frame              : _top
7.  Provider link          : http://www.squirrelmail.org/
8.  Provider name          : SquirrelMail

R   Return to Main Menu
C   Turn color on
S   Save data
Q   Quit

Command >>
```

b)

```
arturo@localhost: /var/www/html/webmail - Terminal Nº 2 - Konsole
Sesión  Editar  Vista  Marcadores  Preferencias  Ayuda
SquirrelMail Configuration : Read: config_default.php (1.4.0)
---------------------------------------------------------
Server Settings

General
-------
1.  Domain                 : iesrioarba.es
2.  Invert Time            : false
3.  Sendmail or SMTP       : SMTP

A.  Update IMAP Settings   : localhost:143 (other)
B.  Update SMTP Settings   : localhost:25

R   Return to Main Menu
C   Turn color on
S   Save data
Q   Quit

Command >>
```

c)

```
arturo@localhost: /var/www/html/webmail - Terminal Nº 2 - Konsole
Sesión  Editar  Vista  Marcadores  Preferencias  Ayuda
SquirrelMail Configuration : Read: config_default.php (1.4.0)
---------------------------------------------------------
Folder Defaults
1.  Default Folder Prefix         :
2.  Show Folder Prefix Option     : false
3.  Trash Folder                  : BASURA
4.  Sent Folder                   : ENVIADOS
5.  Drafts Folder                 : BORRADORES
6.  By default, move to trash     : true
7.  By default, move to sent      : true
8.  By default, save as draft     : true
9.  List Special Folders First    : true
10. Show Special Folders Color    : true
11. Auto Expunge                  : true
12. Default Sub. of INBOX         : true
13. Show 'Contain Sub.' Option    : false
14. Default Unseen Notify         : 2
15. Default Unseen Type           : 1
16. Auto Create Special Folders   : true
17. Folder Delete Bypasses Trash  : false
18. Enable /NoSelect folder fix   : false

R   Return to Main Menu
C   Turn color on
S   Save data
Q   Quit

Command >>
```

Figura 5.14: Configuración de `SquirrelMail`: a) `Organization Preferences`, b) `Server Settings` y c) `Folder Defaults`.

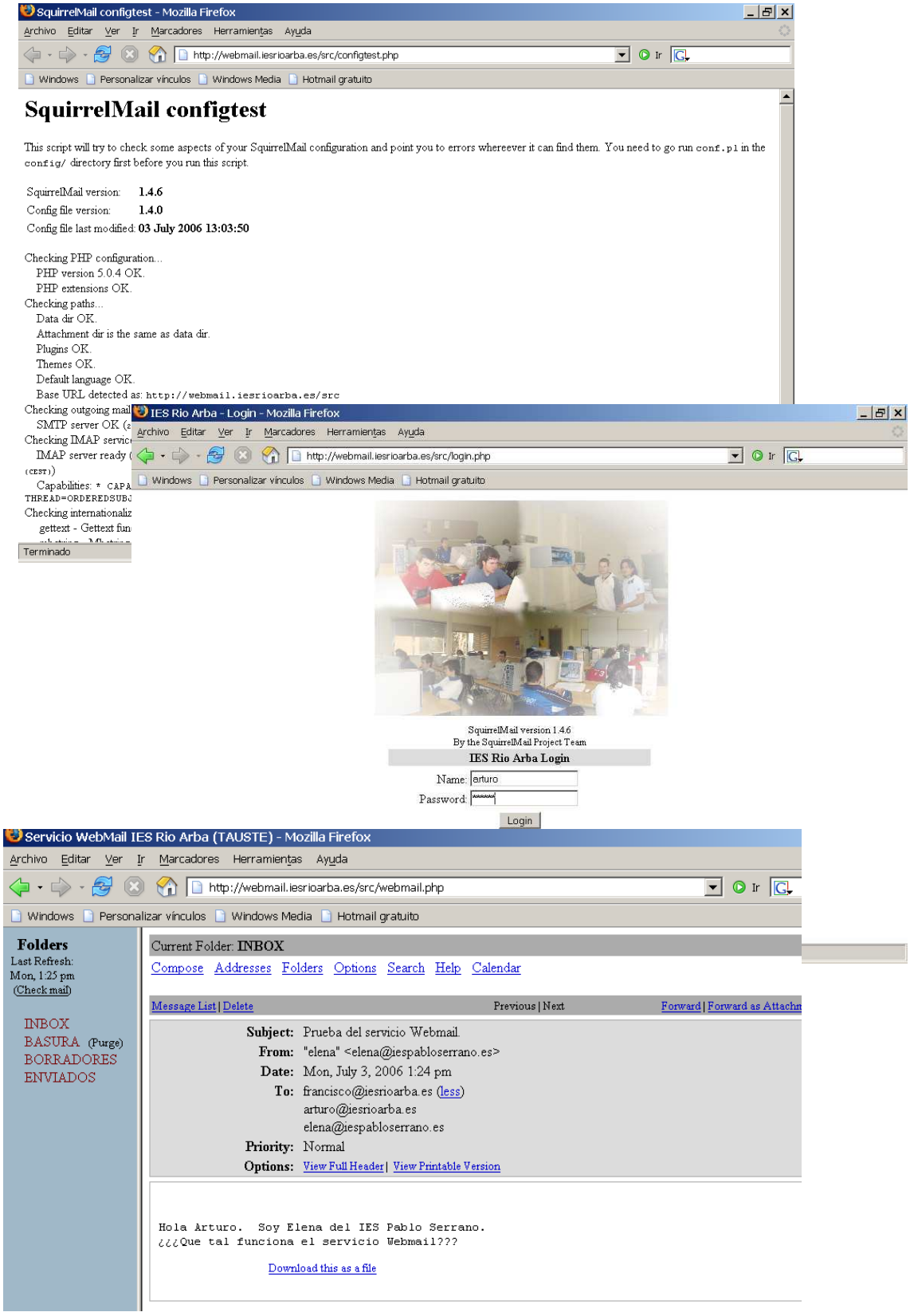


Figura 5.15: (Arriba) Mediante configtest.php es posible testear y comprobar que la configuración realizada es correcta. (Abajo) Acceso vía web a nuestro servidor de correo y comprobación del correcto funcionamiento de `SquirrelMail`: enviar y recibir correo vía web.

6. Una vez que `configtest.php` nos advierta que ya tenemos todo correcto podremos utilizar nuestro servicio WebMail colocando como URL el valor de la directiva `ServerName` especificada en el host virtual creado en apache, en nuestro caso `http://webmail.iesrioarba.es`.

   Para logarnos en el servicio WebMail, deberemos introducir el mismo nombre de usuario y contraseña otorgados al dar de alta la cuenta de usuario dentro del equipo servidor.

Capítulo 6

CONFIGURACIÓN DE UN FIREWALL EN LINUX: IPTABLES

6.1. Introducción a los cortafuegos (Firewall)

La interconexión global de todas las redes informáticas a través de Internet, se ha traducido simultáneamente en una inagotable fuente de servicios (mensajería electrónica, transferencia de archivos, control remoto de equipos, etc.), pero también en un agujero permanente de seguridad. En este capítulo vamos a centrarnos esencialmente en dos aspectos fundamentales:

1. Cómo habilitar el acceso a los servicios y recursos que se ofrecen por las distintas redes, que se encuentran conectadas a Internet, en función de dónde se localice el origen y el destino ⇒ NAT.

   El problema de establecer una comunicación entre dos equipos informáticos, surge en el momento en que uno de los implicados (emisor-receptor, origen-destino) pertenece a una red lógica privada, y trata de identificarse mediante una dirección IP privada únicamente reconocida dentro de la propia intranet a la que pertenece. Es decir, el problema sería equivalente a decir que deseamos poner en comunicación dos usuarios mediante el tradicional correo postal, y alguno de ellos, no tiene asignada una dirección postal pública, y por tanto reconocida por el servicio postal, lo cual derivaría en que no habría forma de que las cartas repartidas por carteros pudiesen alcanzar su correspondiente destino, o remitente en caso de devolución (los paquetes serían descartados).

   Según esto, sólo cuando los dos extremos de la comunicación tengan direcciones reconocidas (direcciones IP públicas) podrán establecer comunicación sin ningún tipo de problemas, y cuando esto no se cumpla, necesitaremos de una estrategia denominada NAT.

2. Cómo filtrar las comunicaciones que se establecen a través de Internet, pudiendo decidir qué servicios son accesibles y cuáles no, con la única pretensión de proteger nuestra red (integridad y confidencialidad). A esto se lo denomina filtrado (FILTER).

   Toda red informática conectada a Internet esta expuesta a posibles `ataques` que pongan en jaque tanto los datos e información confidencial que podamos almacenar (troyanos, sniffing, spoofing, hijacking, etc.), como el correcto funcionamiento (virus, gusanos, applets java, etc.) de los equipos que la forman. Con la finalidad de salvaguardar nuestra red, una posible solución es filtrar toda comunicación que se establece tanto desde nuestra red, como hacia ella.

6.2. Firewall GNU/Linux: IPTABLES

Para convertir nuestro equipo GNU/Linux en un equipo firewall será necesario instalar un paquete software que se integrará con el propio kernel del sistema operativo, el cual nos permi-

tirá decidir que paquetes TCP/IP aceptar y cuales denegar: `iptables`.

En concreto, este software nos permitirá filtrar los paquetes TCP/IP en los niveles de red y transporte (capas 3 y 4) del modelo de referencia TCP/IP. Es decir, esencialmente podremos filtrar en función de quien sea el origen o destino (dirección IP de origen o destino ⇒ capa 3 / nivel red), del protocolo de comunicación (tcp/udp/icmp ⇒ capa 4 / nivel transporte) y el puerto de comunicación (21/ftp, 53/dns, 80/http, 137:139/netbios, 10000/webmin, etc. ⇒ capa 4 / nivel transporte). Todo ello se explicará más detenidamente en la siguiente sección.

Además `iptables` nos va a permitir llevar a cabo `NAT` y mapeo de puertos (`mapping`):

NAT

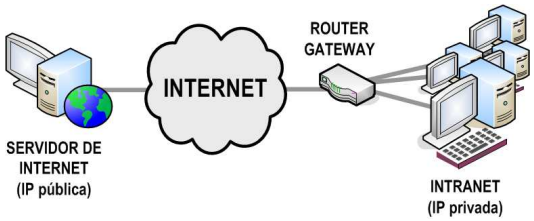
Network Address Translation: Mediante `NAT` se nos permite realizar alteraciones en las direcciones IP de origen (`SNAT`, Source Network Address Translation) y destino (`DNAT`, Destination Network Address Translation) de los paquetes TCP/IP. Es decir, si observaramos la estructura de los paquetes TCP/IP que viajan por la red, advertiríamos que tienen la siguiente estructura: cabecera (header) y datos. En la cabecera se encuentra la información referente a las direcciones y puertos de comunicación de origen y destino. Es decir, su estructura es similar a la carta tradicional, compuesta por un sobre (cabecera), donde se informa de la dirección postal del destinatario y remitente, y un contenido (datos). La finalidad de `NAT` es poder alterar las direcciones de origen y destino con el único objetivo de que la comunicación pueda establecerse.

SNAT (`Source NAT`): La alteración de la dirección IP de origen suele llevarse a cabo por parte de los equipos que hacen de `gateway` entre redes, donde intervienen direcciones IP de tipo privado.

Para comprender `SNAT` supondremos encontrarnos en la siguiente situación: un equipo que se encuentra dentro de una intranet (red privada), que desea conectarse a Internet a través del modem-router suministrado por un proveedor de servicios de Internet. En este supuesto, el equipo que quiere establecer conexión con un servidor cualquiera de la Internet (por ejemplo, `www.google.com`) dispone de una dirección IP privada únicamente reconocida dentro de la intranet donde se encuentra. Esto significa que en el momento en que le llegase la petición emitida por este al equipo servidor de la Internet y observase el remitente no reconocería la dirección IP, y por tanto, no sabría contestarle. Para solucionar este problema, debemos observar que el único equipo dentro de nuestra red que dispone una dirección IP pública, y por tanto, reconocida en Internet es el router/gateway a través del cual nos conectamos, lo que implica que todos los paquetes TCP/IP que atraviesan el router procedentes de la intranet deben ser alterados por este, intercambiando la dirección IP de origen privada por su dirección IP pública (SNAT), de tal forma que al recibir los paquetes el servidor de Internet destinatario, pueda reconocer la dirección IP del remitente y así poderle contestar. A esta modificación de los paquetes TCP/IP se le denomina también `enmascaramiento` (MASQUERADE).

En definitiva, nuestro `modem/router/gateway` es un dispositivo que al menos dispone de dos interfaces de comunicación, una pública y una privada, que le permite hacer de intermediario entre la intranet y la Internet, separando desde el punto de vista lógico ambas redes.

En el caso de implementar SNAT a través de iptables, este nos va a permitir decidir que paquetes alterar en función de,



SNAT:

1. Un equipo de la Intranet (direcciones IP privadas) genera un paquete destinado hacia Internet.
2. Al ser el destinatario del paquete TCP/IP un equipo externo a la red lógica interna es enviado al `gateway/router` para que este decida que hacer con él.
3. Al recibir el `router` el paquete TCP/IP, lo enruta hacia Internet, pero justo antes de expulsar el paquete altera el paquete enmascarando la dirección IP de origen (SNAT) intercambiándola por su dirección IP pública (SNAT POSTROUTING, posterior al enrutamiento, se altera el paquete).
4. El servidor de la Internet responde a la petición contestando con un nuevo paquete TCP/IP dirigido a la dirección IP pública, y por tanto, reconocida del `router`. El `router` al recibir la respuesta desenmascara el paquete y se lo entrega al equipo de la intranet que llevó a cabo la petición.

Figura 6.1: Explicación gráfica de SNAT.

- la interfaz de red por donde vayan a salir los paquetes TCP/IP (opción `-o`, output) tras su enrutamiento (POSTROUTING).
- quien sea el equipo origen (opción `-s`, source, dirección IP de origen).
- a quien vaya destinado (opción `-d`, destination, dirección IP de destino).
- el protocolo de comunicación utilizado (opción `-p`, protocol), o el servicio o puerto de comunicaciones al que se desea acceder (opción `--dport`, destination port).
  permitiéndonos decidir por que dirección IP intercambiar la dirección IP de origen (opción `-j SNAT --to`).

En cuanto a la sintaxis del comando SNAT (opción `-A` ⇒ "Añadir regla a la lista POSTROUTING SNAT"):

```
[root@linux] iptables -t nat -A POSTROUTING -o 'interfaz red' -s 'dirección
             IP origen' -d 'dirección IP destino' -p 'protocolo'
             --dport 'puerto de destino' -j SNAT --to 'nueva IP de origen'
```

Por ejemplo, si quisiéramos únicamente alterar todos aquellos paquetes que saliesen de nuestro equipo `router/gateway` a través de su interfaz `eth0` procedentes de la red privada `192.168.1.0/24` en caso de utilizar el protocolo tcp/80 (http):

```
[root@linux] iptables -t nat -A POSTROUTING -o eth0 -s 192.168.1.0/24
             -p tcp --dport 80 -j SNAT --to 37.98.111.65
```

Donde se ha supuesto que la dirección IP de la interfaz `eth0` del router es `37.98.111.65`.

Si observamos detenidamente el comando anterior podremos advertir la existencia de una redundancia. Estamos indicando que todo aquel paquete TCP/IP que salga por `eth0` y cumpla las condiciones marcadas sea alterado enmascarando la dirección IP de origen por la de dicha interfaz. Es decir, si tenemos en cuenta que todo enmascaramiento conlleva implícitamente un intercambio de la dirección IP de origen del paquete por la de la interfaz del equipo por la que salga, no es necesario indicarlo en el comando explicitamente. Según esto, detallar todo lo anterior tan sólo tiene sentido en caso de que la interfaz de red de salida, en este caso `eth0`, tenga asignada más de una dirección IP y queramos evitar ambigüedades. Por este motivo, en los casos en que nuestra

interfaz de red sólo tenga asignada una dirección IP haremos `SNAT` de manera implicita escribiendo `-j MASQUERADE` indicando exactamente lo mismo. Así la anterior instrucción podría escribirse,

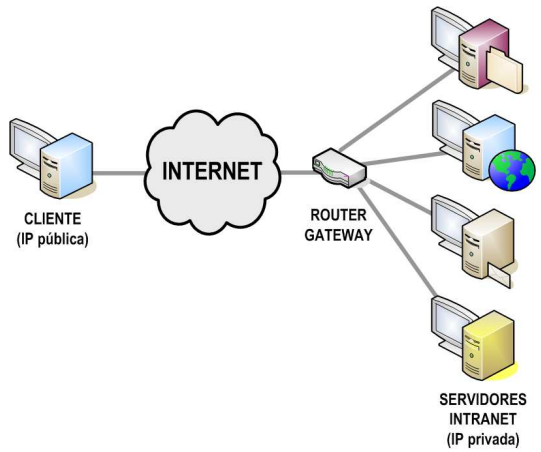
```
[root@linux]  iptables -t nat -A POSTROUTING -o eth0 -s 192.168.1.0/24
              -p tcp --dport 80 -j MASQUERADE
```

Como la situación habitual es filtrar los paquetes mediante la opción `iptables -t filter`, lo normal es que por defecto nuestro `router/gateway` enmascare todo lo que salga por cualquiera de sus interfaces de red, sin poner ningún tipo de restricciones a través de `nat` (`iptables -t nat` *restricciones*), y posteriormente mediante `filter` decidiremos que dejamos pasar y que no. Por ello lo común es que el enmascaramiento sea general, independientemente de la interfaz de salida, de las direcciones de origen y destino, y del protocolo utilizado, quedando finalmente:

```
[root@linux]# iptables -t nat -A POSTROUTING -j MASQUERADE
```

DNAT (`Destination NAT`): La alteración de la dirección IP de destino suele llevarse a cabo por parte de los equipos que hacen de `gateway` en el momento en que este se encarga de encubrir los servicios proporcionados por una red con direcciones IP privadas (intranet).

Para comprender `DNAT` supondremos encontrarnos en la siguiente situación: un equipo conectado a Internet desea acceder a un servicio ofrecido por otro equipo ubicado dentro de una intranet. En este supuesto el equipo servidor al pertenecer a una red interna privada tiene asignada una dirección IP no reconocida en Internet, y por tanto, inaccesible. Esto se traduce en que la única manera de que este servidor pueda recibir solicitudes de conexión es a través de un *intermediario* que se encargue de recoger las solicitudes emitidas desde cualquier punto de la Internet (mediante una dirección IP pública), y que posteriormente se las entregue a este (que posee una dirección IP privada). Este dispositivo *intermediario* no es otro más que el `router/gateway`.



DNAT:

1. Un equipo conectado a Internet realiza una solicitud a un servicio ofrecido por una Intranet (dirección IP privada ⇒ inaccesible), colocando como destinatario del paquete TCP/IP la dirección IP pública del `router/gateway` que hace de intermediario.
2. El dispositivo `router/gateway` recibe la solicitud y reconoce el servicio al que desea conectarse a través del puerto de comunicación de destino.
3. Intercambia la dirección IP pública de destino (DNAT) por la dirección IP privada dentro de la Internet correspondiente al equipo que ofrece el servicio solicitado. Una vez alterado (PREROUTING) el paquete TCP/IP es enrutado por el `router/gateway` hacia la interfaz de red que le permite a éste establecer comunicación con el servidor privado (DNAT PREROUTING, previo al enrutamiento se realiza la ateración del paquete TCP/IP).
4. El servidor de la intranet recibe la solicitud y emite una respuesta entregándosela nuevamente al `router/gateway` para que se la haga llegar al equipo de la Internet que inicio la comunicación.

Figura 6.2: Explicación gráfica de DNAT.

En el caso de implementar DNAT a través de iptables, este nos va a permitir decidir que paquetes alterar en función de,

- la interfaz de red por donde entren los paquetes TCP/IP (opción `-i`, input).
- quien sea el equipo origen (opción `-s`, source, dirección IP de origen).
- el protocolo de comunicación utilizado (opción `-p`, protocol), o el servicio o puerto de comunicaciones al que se desea acceder (opción `--dport`, destination port),

permitiéndonos decidir porque dirección IP intercambiar la dirección IP y puerto de destino (opción `-j DNAT --to`). En cuanto a la sintaxis del comando DNAT[1] sería la siguiente:

```
[root@linux]  iptables -t nat -A PREROUTING -i 'interfaz red' -s 'dirección
              IP origen' -p 'protocolo comunicaciones' --dport 'puerto
              comunicaciones de destino' -j DNAT --to 'nueva dirección IP
              y puerto de destino'
```

Por ejemplo, si quisieramos que aquellos paquetes recibidos por nuestro `router/gateway` a través de su interfaz de red `eth0` con protocolo `tcp/udp` destinados al puerto de comunicaciones[2] 21 fuesen alterados con la finalidad de intercambiar la dirección IP de destino (la de su interfaz de entrada) por la dirección IP (por ejemplo, `192.168.1.5`) del servidor FTP de la intranet (red privada) a la que encubre deberíamos ejecutar los siguientes comandos:

```
[root@linux]  iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 21
              -j DNAT --to 192.168.1.5:21
```

```
[root@linux]  iptables -t nat -A PREROUTING -i eth0 -p udp --dport 21
              -j DNAT --to 192.168.1.5:21
```

De esta forma, los paquetes recibidos que cumplan las condiciones anteriores serán alterados por DNAT, para que posteriormente sean enrutados (reenviados, forwarding) por el `router/gateway` hacia la interfaz de red que le pone en comunicación con el servidor FTP. Por tanto, se hace la alteración como paso previo al enrutamiento (*`PREROUTING`*). Esta es la razón por la cual DNAT siempre esta asociado a PREROUTING, al igual que ocurre con SNAT que se asocia con el POSTROUTING.

mapping

El mapeo de puertos no es más que una forma de `NAT` en la cual se alteran los paquetes TCP/IP con la finalidad de modificar los puertos de comunicaciones origen y destino. Esta situación es habitual en `DNAT`. Por ejemplo, siguiendo la figura 6.3, imaginad que nuestro `router/gateway` encubre una intranet que da servicio web[3] a través de un equipo servidor con dirección IP privada `192.168.1.4` a través del puerto 8080.

Todo cliente Web al realizar una solicitud a un servidor Web la hace por defecto al puerto 80. Al recibir esta solicitud el `router/gateway` que hace de intermediario entre el equipo conectado a Internet y el servidor que se encuentra dentro de la intranet a la que encubre, deberá alterar los paquetes TCP/IP doblemente, ya que debe modificar tanto la dirección IP como el puerto de destino. Para llevar a cabo este DNAT con `mapping` haciendo uso de iptables ejecutaríamos el siguiente comando,

```
[root@linux]  iptables -t nat -A PREROUTING -i eth0 -p tcp
                        --dport 80 -j DNAT --to 192.168.1.4:8080
```

[1] Se empleará la opción `-A` en el caso de querer 'añadir una regla a la lista PREROUTING DNAT'
[2] El puerto 21 corresponde a `ftp`, protocolo de transferencia de ficheros.
[3] El servicio web utiliza http, que es un protocolo tcp. El puerto por defecto para ofrecer este servicio es el 80.

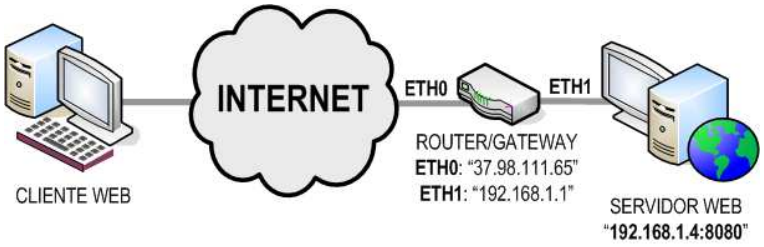


Figura 6.3: Ejemplo de mapping.

6.3. Sintaxis de IPTABLES para el filtrado de paquetes

Un firewall es un componente hardware (dispositivo dedicado) o software (por ejemplo, GNU/-Linux) con capacidad de filtrar los paquetes TCP/IP que maneja en base a una política (reglas de filtrado) previamente establecida. Según esto, un firewall inspecciona el contenido de los paquetes TCP/IP (su cabecera) que recibe por sus distintas interfaces de red (eth0, eth1, etc.), y en base a las "reglas de filtrado" programadas establece una decisión entre tres posibles:

1. Aceptar el paquete TCP/IP (ACCEPT).

2. Denegar el paquete TCP/IP (DROP).

3. Rechazar el paquete TCP/IP (REJECT). La diferencia entre rechazar el paquete y denegarlo, es que cuando se deniega un paquete no hay que dar ninguna explicación, y en cambio, si se opta por rechazarlo, es necesario devolver al emisor (origen) del paquete TCP/IP un acuse de recibo en el que se indica el motivo del rechazo.

Un firewall implementado mediante `iptables` dispone de tres listas de reglas de filtardo (`tabla FILTER`), `INPUT`, `FORWARD`, y `OUTPUT`. A estas listas de reglas también se les denomina `cadenas` o `chains`, y sus características son las siguientes:

Lista/Chain	Función de las reglas incluidas en la lista
INPUT	Sus reglas son consultadas en el momento en que entra un paquete por una de las interfaces de red del firewall y figura él en la cabecera TCP/IP como destinatario (dirección IP de destino). La decisión a adoptar, como ya se ha explicado, puede ser: aceptarlo (`ACCEPT`), denegarlo (`DROP`) ó rechazarlo (`REJECT`).
FORWARD	Sus reglas son consultadas en el momento en que el firewall recibe un paquete por una de sus interfaces de red y no figura el como destinatario, siendo necesario un reenvió o enrutamiento de este hacia una interfaz de salida con la finalidad de que pueda alcanzar su destino. Las reglas pueden determinar si se reenvia (`ACCEPT`) o no (`DROP/REJECT`).
OUTPUT	Sus reglas son consultadas en el momento en que va a salir un paquete TCP/IP por alguna de las interfaces de red del firewall y figura él como origen (dirección IP de origen), pudiendo decidir entre dejarlo salir (`ACCEPT`) o no (`DROP/REJECT`).

El comando parametrizado `iptables` posee numerosas opciones. Algunas de las más importantes se muestran en la siguiente lista. Todas ellas serán utilizadas posteriormente a lo largo del capítulo mediante diferentes ejemplos prácticos.

Opciones del comando `iptables`

- **-t 'tipo'**

La opción `-t` (-t, table) nos permite especificar a iptables si queremos llevar a cabo una operación de filtrado (-t filter) o una alteración de los paquetes TCP/IP (-t nat) (ver sección 6.2). En caso de no indicarse ninguna de las opciones se da por hecho que se trata de una operación de filtrado. Por tanto, las dos instrucciones siguientes son equivalentes,

```
[root@Linux] iptables -t filter -L ⇒ [root@Linux] iptables -L
```

Además de las dos tablas anteriores relacionadas con el filtrado (-t filter) y la nat (-t nat) existe una última tabla relacionada con la alteración especializada de paquetes, mangle (-t mangle), que no será utilizada en este capítulo.
Ejemplos.-

```
[root@linux]#  iptables -t filter -L
```

```
[root@linux]#  iptables -t nat -L
```

- **-L 'lista'**

Esta opción nos lista (-L, list) todas las reglas de la lista de la tabla especificada. Las listas de mayor interés son `INPUT`, `FORWARD` y `OUTPUT` asociadas a la tabla de filtrado (-t filter), y `POSTROUTING` y `PREROUTING` asociadas a la tabla de nat (-t nat). Es importante listar las reglas para conocer el orden en el que se van a ejecutar; recordar que se recorren secuencialmente hasta que se encuentra una regla que afecta al paquete que se este analizando.
Ejemplos.-

```
[root@linux]#  iptables -t filter -L INPUT
```

```
[root@linux]# iptables -t nat -L POSTROUTING
```

En caso de no indicar el nombre de ninguna lista, por defecto se mostrarán todas las listas de reglas asociadas a la tabla especificada, `-t filter` o `-t nat`. En ocasiones puede resultar interesante que acompañe a la opción listar (-L, list) algunos parámetros que faciliten la interpretación de la información mostrada. Entre ellos cabría señalar tres:

-v ⇒ Modo verboso. Esta opción resulta útil cuando deseamos listar las reglas que contiene una de las listas suministrando la mayor información posible.

--line-numbers ⇒ Numera las reglas que contiene cada una de las listas. Esto puede resultar útil en aquellos casos en que estemos interesados en conocer la ubicación concreta de una regla dentro de una lista de una tabla: operaciones de borrado (-D, delete) e inserción (-I, insert).

-n ⇒ Mediante esta opción indicamos a `iptables` que no resuelva ni los nombres de dominio, ni los nombres de los servicios asociados a las reglas indicando de manera numérica tanto las

direcciones IP como los números de puertos de comunicaciones. Si no se indica, por defecto `iptables` tratará de hacer la asociación de direcciones IP a los correspondientes nombres de dominio (o alias), y de números de puerto a su correspondiente nombre de servicio.

Ejemplos.-

```
[root@linux]# iptables -t filter -L INPUT --line-numbers
```

```
[root@linux]# iptables -t filter -L FORWARD -n -v --line-numbers
```

- **-A 'lista' 'regla'**

Añade (-A, append) una nueva regla al final de la lista especificada. Esta es la forma común de programar un `firewall`: una vez que se ha realizado un estudio de que se desea filtrar, se establecen las reglas para ello, y se van añadiendo a la lista correspondiente una a una.
Ejemplos.-

```
[root@linux]# iptables -t filter -A FORWARD -i eth1 -o eth0 -j DROP
```

```
[root@linux]# iptables -t nat -A POSTROUTING -j MASQUERADE
```

- **-I 'lista' 'posicion' 'regla'**

Inserta (-I, insert) una nueva regla en la lista indicada, en la posición especificada. Haremos uso de esta opción en caso de que una vez configurado el `firewall`, queramos hacer unos últimos retoques ajustando el filtrado resultante.
Ejemplo.-

```
[root@linux]# iptables -t filter -I FORWARD 6 -i eth1 -o eth0 -j DROP
```

- **-F 'lista'**

Borra (-F, flush) todas las reglas de lista especificada. Este comando resulta útil cuando deseamos reconfigurar (inicializar) un `firewall` de nuevo. En caso de no especificar una lista concreta, borra todas las listas de reglas de la tabla correspondiente:

```
[root@linux]# iptables -t filter -F
```

Ejemplo.-

```
[root@linux]# iptables -t nat -F PREROUTING
```

- **-D 'lista' 'posicion'**

Permite eliminar (-D, delete) una regla concreta dentro de una lista. Teniendo en cuenta que el orden de las reglas dentro de una lista es importante de cara al filtrado posterior, puede interesarnos eliminar una de las reglas, para insertarla (-I, insert) en una nueva posición, o simplemente porque no nos interesa.
Ejemplo.-

```
[root@linux]# iptables -t filter -D INPUT 3
```

- **-Z 'lista'**

Pone a cero (-Z, zero) el contador del número de reglas que posee una lista. Resulta útil hacer uso de esta opción tras haber optado por borrar todas las reglas de una lista o tabla. Si no se especifica que lista, pone a cero los contadores de todas las listas asociadas a la tabla indicada.
Ejemplo.-

```
[root@linux]# iptables -t filter -Z OUTPUT
```

En relación a la sintaxis a cumplir a la hora de especificar una regla concreta, existen multitud de parámetros que nos permiten ajustar la semántica de la regla para que haga exactamente lo que deseamos.

Parámetros de `iptables`

- **-j 'accion'**

Específica cual es la acción a llevar a cabo sobre el paquete TCP/IP que cumple las condiciones de la regla.
Ejemplo.-

```
[root@linux]# iptables -t filter -A INPUT -i lo -j ACCEPT
```

En relación a las reglas de filtrado (-t filter), tal como se ha señalado a lo largo del capítulo, una vez que el `firewall` analiza el paquete TCP/IP, tenemos tres posibles acciones a llevar a cabo: aceptarlo (`ACCEPT`), denegarlo (`DROP`), o rechazarlo (`REJECT`). En el caso de optar por rechazarlo (`REJECT`) habrá que especificar la razón de su rechazo (icmp-net-unreacheable, icmp-host-unreachable, icmp-port-unreachable, icmp-proto-unreachable, icmp-net-prohibited ó icmp-host-prohibited):

Ejemplo.-

```
[root@Linux]  iptables -t filter -A INPUT -p udp -j REJECT
                         --reject-with icmp-port-unreachable
```

En relación a las reglas de nat (-t nat), tal como se ha indicado en la sección 6.2, existen básicamente dos acciones a poder llevar a cabo: SNAT (alteración de la dirección IP de origen) o DNAT (alteración de la dirección IP de destino).

Ejemplo.-

```
[root@Linux]  iptables -t nat -A POSTROUTING -s 192.168.121.0/24
                          -o eth0 -j SNAT --to 21.34.197.8
```

- **-i 'interfaz input' y -o 'interfaz output'**

Mediante estos parámetros podemos especificar que interfaces o tarjetas de red de nuestro equipo `firewall` GNU/Linux se van a ver afectadas por la regla de filtrado (`-t filter`) o nat (`-t nat`). Por ejemplo, en el caso en que nuestro `firewall` desempeñe las funciones de `gateway/router` (FORWARD) necesitará al menos de dos tarjetas de red (eth0 y eth1), lo que obligará a especificar la interfaz por la que entran los paquetes que se verán afectados por la regla de filtrado, y de igual forma, la interfaz por donde saldrían (eth0, eth1, eth2, etc.). En caso de no indicar nada se asumirá que la regla por defecto afectara a todas las interfaces de red.
Ejemplo.-

```
[root@linux]# iptables -t filter -A FORWARD -i eth1 -o eth0 -j DROP
```

- **-s 'origen' y -d 'destino'**

Ejemplo.-

```
[root@linux]# iptables -t filter -A INPUT -s 192.168.121.0/24 -j ACCEPT
```

Muchas veces puede resultar interesante especificar direcciones IP de origen o de destino a las que deseamos les afecte la regla con la finalidad de restringir el acceso a determinados equipos. En caso de no indicar nada por defecto se asumirá que la regla afectará a cualquier origen o destino.

- **-p 'protocolo' ,--dport 'nº puerto' y --sport 'nº puerto'**

La opción `-p` (protocol) nos permite filtrar en función del protocolo utilizado y el puerto de origen o destino.
Ejemplo.-

```
[root@Linux]  iptables -t filter -A INPUT -s 192.168.121.0/24
                         -p tcp --dport 80 -j ACCEPT
```

Para su correcta utilización deberemos conocer que protocolo utiliza cada uno de los servicios y el puerto asociado (puerto de destino, `--dport`, o puerto de origen, `--sport`). Entre los más conocidos cabría destacar los siguientes:

`TCP: 80 (WWW/HTTP), 21 (FTP), 25 (SMTP), 23 (Telnet/Control remoto), 22 (SSH/Remote Login Protocol Security), 109-110 (POP ver.2 y ver.3), 53 (DNS/Domain Name Server).`
`UDP: 53 (DNS/Domain Name Server), 123 (NTP/Network Time Protocol).`

La diferencia entre los servicios que hacen uso del protocolo TCP y los que hacen uso de UDP, es que los primeros están orientados a conexión, requiriendo de un establecimiento previo de la conexión antes de llevarse a cabo la transferencia de la información, y en los segundos (UDP), esta no es necesaria. Esto supone que las comunicaciones que hacen uso de TCP son más fiables (sin duplicados, sin pérdidas, sin errores y con una secuencia ordenada de paquetes) que las UDP (menos fiabilidad a costa de mayor sencillez en la comunicación).

Además de los dos protocolos anteriores también puede establecerse restricciones de uso sobre el protocolo de nivel de red `ICMP`. Este es utilizado implícitamente en las comunicaciones TCP/IP para informar a las equipos involucrados (origen, destino y enrutadores) del estado de la comunicación: mensajes de control y error. Un uso explicito de él se hace al ejecutar el comando ping usado habitualmente para comprobar que el enlace esta activo.

A modo de resumen, en la figura 6.5 se describen las opciones más importantes de las que se hará uso en `iptables` tanto para el filtrado de paquetes TCP/IP como para la `nat`.

6.3.1. Opciones de filtrado: Entrada, Salida y Reenvío

Cuando se desea configurar un cortafuegos es necesario analizar que tipo de filtrado (opción `-t filter`) queremos llevar a cabo. Pueden distinguirse tres tipos:

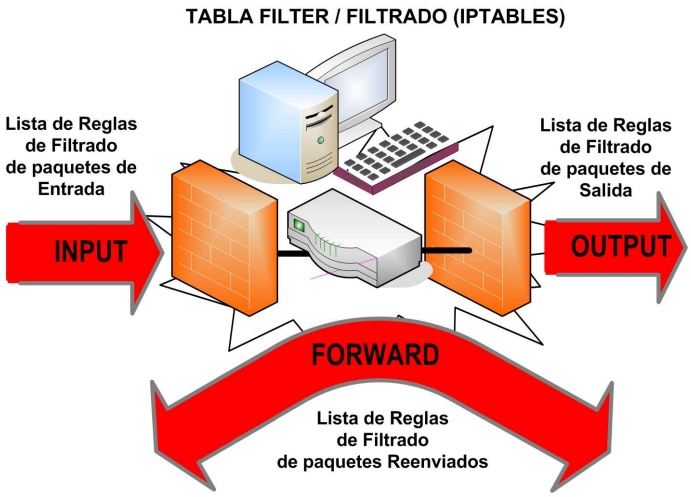


Figura 6.4: Tipos de filtrado de paquetes TCP/IP que pueden realizarse con iptables.

1. Filtrado de entrada (INPUT) . Nos permite decidir que paquetes dejaremos entrar a nuestro equipo, figurando este como destinatario[4]. Se establece que paquetes se aceptan o deniegan en función de,

   - la interfaz de red por la que entren los paquetes TCP/IP al equipo. Opción `-i` (input).
   - quien sea el remitente de los paquetes. Opción `-s` (source), dirección IP de origen.
   - el tipo de protocolo, opción `-p` (protocol[5]) y puerto a través del cual se quiere llevar a cabo la comunicación, opción `--dport` (destination port), puerto de destino.

[4]Recordar que no todos los paquetes que entrar a nuestro equipo van destinados a él. En el caso de que desempeñe funciones de `gateway` (intermediario) el destinatario de los paquetes será otro equipo.

[5]Puede distinguirse entre los protocolos tcp, udp e icmp.

$$
iptables\ -t \left\{
\begin{array}{l}
filter \left\langle \begin{array}{l} -A \\ -I \\ -D \end{array} \right\rangle \left\{
\begin{array}{l}
INPUT\ [-i\ \ interfaz\ red][-s\ \ dirección\ IP] \\
OUTPUT\ [-o\ \ interfaz\ red][-d\ \ dirección\ IP] \\
FORWARD\ [-i\ \ interfaz\ red][-o\ \ interfaz\ red] \left[\begin{array}{l} -s\ dirección\ IP \\ -d\ dirección\ IP \end{array} \right]
\end{array}
\right\} -p \left\{ \begin{array}{l} tcp \\ udp \\ icmp \end{array} \right\} \left[\begin{array}{l} --sport\ \ puerto\ origen \\ --dport\ \ puerto\ destino \end{array} \right] -j \left\{ \begin{array}{l} ACCEPT \\ DROP \\ REJECT \end{array} \right\} \\
\\
nat \left\langle \begin{array}{l} -A \\ -I \\ -D \end{array} \right\rangle \left\{
\begin{array}{l}
PREROUTING\ [-i\ \ interfaz\ red] \left[-p \left\{ \begin{array}{l} tcp \\ udp \end{array} \right\} \right] [--dport\ \ puerto\ destino] -j\ DNAT\ --to\ \ dirección\ IP \\
POSTROUTING\ [-o\ \ interfaz\ red]\ -j\ \ MASQUERADE
\end{array}
\right.
\end{array}
\right.
$$

Figura 6.5: Esquema explicativo del uso del comando `iptables` con sus opciones básicas.

En el caso de querer añadir una nueva regla a la lista de filtrado de entrada (`-A INPUT`) la sintaxis del comando sería la siguiente:

```
[root@linux]  iptables -t filter -A INPUT -i 'interfaz red'
              -s 'dirección IP origen' -p 'protocolo comunicaciones'
              --dport 'puerto de destino' -j 'ACCEPT/DROP/REJECT'
```

A modo de ejemplo, supongamos que nos encontramos con un servidor de archivos samba (protocolo tcp/upd, puertos 137, 138 y 139) con dos interfaces de red (eth0 y eth1) que le permiten establecer comunicación con la red `192.168.2.0/24` e Internet tal como se muestra en la figura 6.6. Si quisiéramos proteger nuestro servidor de tal forma que tan sólo pudiesen

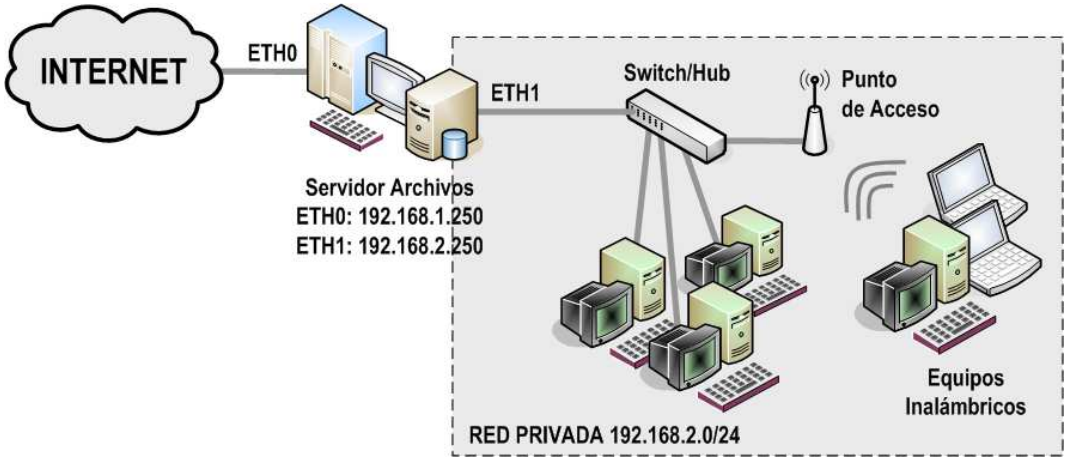


Figura 6.6: Ejemplo de uso de filtrado de paquetes de entrada.

conectarse a él los equipos de la red privada, y en ningún caso desde Internet, evitando al mismo tiempo la entrada por cualquier otro puerto, una posible solución sería ejecutar el siguiente conjunto de comandos `iptables` (shellscript):

```
#Script de Configuración del FIREWALL: REGLAS DE FILTRADO DE ENTRADA (INPUT)
# 1°.- Rechazamos (-j DROP) cualquier paquete TCP/IP que trate de establecer
#conexión con el equipo servidor a través de la interfaz eth0 (-i eth0):
iptables -t filter -A INPUT -i eth0 -j DROP
# 2°.- Aceptamos (-j ACCEPT) las conexiones al servidor de archivos samba (-p
#tcp/udp --dport 137:139) a través de la interfaz de red ETH1 (-i eth1):
iptables -t filter -A INPUT -i eth1 -p tcp --dport 137:139 -j ACCEPT
iptables -t filter -A INPUT -i eth1 -p udp --dport 137:139 -j ACCEPT
# 3°.- Rechazamos cualquier otro tipo de conexión al servidor a través de eth1:
iptables -t filter -A INPUT -i eth1 -j DROP
```

Si quisiéramos modificar la configuración anterior permitiendo que el `jefe` pudiera conectarse desde su casa (Internet) al servidor de archivos de la empresa, deberíamos permitir su entrada por la interfaz `eth0`. Suponiendo que la dirección IP pública[6] del `jefe` es la `111.21.8.45` sería necesario insertar[7] las siguientes reglas:

```
[root@linux]  iptables -t filter -I INPUT 1 -i eth0 -s 111.21.8.45
              -p tcp --dport 137:139 -j ACCEPT
[root@linux]  iptables -t filter -I INPUT 2 -i eth0 -s 111.21.8.45
              -p udp --dport 137:139 -j ACCEPT
```

[6] Dirección a través de la cual sale a Internet.

[7] Observa que en este caso es necesaria la opción `I` ⇒ 'Insertar regla en la posición indicada'.

Las reglas de filtrado se van leyendo de manera secuencial hasta que se encuentra una regla que afecta al paquete que se analiza en ese momento. Por esta razón debería advertirse que en los dos comandos anteriores `iptables` se ha hecho uso de la opción `-I` en lugar de `-A` con la finalidad de intercalar las reglas de filtrado al comienzo de la lista de filtrado (posiciones 1 y 2) para anteceder a la regla `iptables -t filter -A INPUT -i eth0 -j DROP` del script anterior y así garantizar el acceso para `111.21.8.45`.

2. Filtrado de salida (OUTPUT). El administrador del equipo (o cualquier otro usuario al que se le hayan concedido los suficientes privilegios) también tiene la opción de filtrar los paquetes que salen de él[8]. Es importante tener en cuenta, que si ese equipo hace reenvío de paquetes procedentes de otros equipos[9] a los que interconecta, estos podrán salir por la interfaz de red correspondiente, sin tener en cuenta estas reglas de filtrado, ya que para ello se dispone de otras reglas de filtrado denominadas de reenvió (FORWARD). Concretando, podemos decidir que paquetes permitimos salir en función de,

   - la interfaz de red por la que salgan estos. Opción `-o` (output).
   - quien sea el destinatario. Opción `-d` (destination), dirección IP de destino.
   - el tipo de protocolo, opción `-p` (protocol) y puerto a través del cual se quiere llevar a cabo la comunicación, opción `--dport` (destination port), puerto de destino.

   En el caso de querer añadir una nueva regla a la lista de filtrado de salida (`-A OUTPUT`) la sintaxis del comando sería la siguiente:

```
[root@linux]  iptables -t filter -A OUTPUT -o 'interfaz red'
              -d 'dirección IP destino' -p 'protocolo comunicaciones'
              --dport 'puerto de destino' -j 'ACCEPT/DROP/REJECT'
```

   Veamos un ejemplo. Suponed un equipo personal con una única tarjeta de red. Deseamos restringir las conexiones a Internet permitiéndole únicamente enviar (protocolo tcp, puerto 25 `smtp`) y recibir correo (protocolo tcp, puerto 110 `pop3`) electrónico. La secuencia de comandos sería,

```
#Script de Configuración del FIREWALL: REGLAS DE FILTRADO DE SALIDA (OUTPUT)
# 1°.- Aceptamos (-j ACCEPT) las conexiones a servidores de correo saliente
# (-p tcp --dport 25) y entrante (-p tcp --dport 110) a través de la interfaz
# de red eth0 (-o eth0):
iptables -t filter -A OUTPUT -o eth0 -p tcp --dport 25 -j ACCEPT
iptables -t filter -A OUTPUT -o eth0 -p tcp --dport 110 -j ACCEPT
# 2°.- Teniendo en cuenta que para hacer uso de direcciones de correo
# electrónicas asociadas a un nombre de dominio es necesario una resolución
# previa, aceptaremos las conexiones al servicio DNS (-p udp --dport 53):
iptables -t filter -A OUTPUT -o eth0 -p udp --dport 53 -j ACCEPT
# 3°.- Rechazamos cualquier otro tipo de conexión a otro servicio:
iptables -t filter -A OUTPUT -o eth0 -j DROP
```

   Como puede advertirse en el shellcript de configuración del firewall anterior, además de permitir conexiones hacia servidores `smtp` y `pop3` es necesario permitir conexiones al servicio `DNS` (Servidor Nombres de Dominio) ya que cuando enviamos un correo electrónico, por ejemplo a `amartinr@educa.aragon.es`, nuestro equipo necesita conocer la dirección IP del equipo servidor `educa.aragon.es` encargado de recoger todos los mensajes enviados hacia los usuarios registrados en ese dominio.

[8] Esto significa que nuestro equipo será origen (dirección IP de origen) de esos paquetes.
[9] Evidentemente en esos paquetes no figuraría ese equipo como origen.

3.- Filtrado de reenvio (FORWARD). Si deseamos programar un firewall en un equipo que hace las funciones de `gateway` o `router` será necesario filtrar aquellos paquetes que entran por una interfaz de red y son reenviados hacia otra interfaz[10]. En este caso haremos uso de reglas de filtrado de tipo `FORWARD`. En concreto, se nos permite decidir que paquetes reenviar en función de

- la interfaz de red por la que entren. Opción `-i` (input).
- la interfaz de red por la que salgan. Opción `-o` (output).
- quien sea el origen. Opción `-s` (source), dirección IP de origen.
- quien sea el destinatario del paquete. Opción `-d` (destination), dirección IP de destino.
- el protocolo utilizado, opción `-p` (protocol) y de los puertos de comunicaciones utilizados, opción `--dport` para el puerto de destino y `--sport` para el puerto de origen.

En el caso de querer añadir una nueva regla a la lista de filtrado de reenvío (`-A FORWARD`) la sintaxis del comando sería la siguiente:

```
[root@linux]  iptables -t filter -A FORWARD -i 'interfaz red'
              -o 'interfaz red' -s 'dirección IP origen' -d 'dirección IP destino'
              -p 'protocolo' --dport 'puerto de destino' -j 'ACCEPT|DROP|REJECT'
```

En este momento puede que la cantidad de información dada parezca elevada e incluso confusa. A lo largo del capítulo se mostrarán ejemplos resueltos de distintas configuraciones de `gateway/firewall` para que sean implementadas y probadas.

6.4. Post-Configuración del Firewall GNU/Linux con IPTABLES

Una característica que presenta la configuración de un firewall mediante el uso de iptables, es que sólo es efectiva hasta el siguiente apagado del equipo. Es decir, en caso de tener que reiniciar el equipo GNU/Linux que desempeña la función de firewall, todas las reglas (`filter` y `nat`) se habrán perdido, y con ello toda la configuración.

En el caso de desear que nuestro firewall se reconfigure en cada reinicio del equipo, una posible opción es incluir todas las instrucciones iptables de programación del firewall en un script, y configurar este como script de inicio. De esta forma, cada vez que iniciemos nuestro equipo GNU/Linux será ejecutado el correspondiente script reconfigurando el firewall. Los pasos a seguir para ello son:

1. Crearemos el script dentro del directorio `/etc/init.d`. Mediante el uso de nuestro editor de textos preferido (vi, vim, kate, kwrite, gedit, etc.) escribiremos una a una las instrucciones iptables encargadas de configurar el firewall. Por ejemplo, en el caso de que nuestro script de inicio encargado de configurar nuestro GNU/Linux se llame `conf_firewall`[11]:

```
[root@linux]# vi /etc/init.d/conf_firewall
```

[10] Los paquetes atraviesan el equipo. En ellos no figura el gateway como origen ni como destinatario.
[11] Observad que el script se corresponde con la solución al ejercicio práctico N°3 de la práctica

```
#!/bin/sh
#Script de Configuración del GATEWAY/FIREWALL: /etc/init.d/conf_firewall
#
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -F
iptables -t filter -F
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -A FORWARD -i eth0 -o eth1 -s 192.168.199.0/24 -p tcp
                                                --dport 137:139 -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -s 192.168.199.0/24 -p udp
                                                 --dport 137:139 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -p ICMP -j ACCEPT
iptables -t filter -P FORWARD DROP
```

Donde la primera línea del `script` indica quien será el intérprete de comandos de `GNU/Linux` encargado de ejecutar el script de inicio, para lo cual habrá que darle permisos de ejecución al script:

```
[root@linux]# chmod u+x /etc/init.d/conf_firewall
```

En el directorio `/etc/init.d` se encuentran todos los scripts que serán ejecutados al iniciar el equipo, pero será necesario indicárselo explícitamente a GNU/Linux. La foma de hacer esta indicación se explicará en los puntos siguientes.

2. Observaremos el contenido del fichero de inicio del sistema `/etc/inittab`. En concreto la línea referente al modo de inicio, cuyo aspecto es similar a,

   ```
   id:3:initdefault:
   ```

   de la cual, lo que realmente nos interesa es el número que aparece en el segundo campo (en el ejemplo, 3). En concreto, este número (3) informa de que nuestro equipo GNU/Linux se inicia en modo multiusuario, con soporte de red, pero sin interfaz gráfica[12].

   En GNU/Linux existen cinco modalidades de inicio (1,2,3,4,5), de las cuales las más habituales[13] son la 3 y 5. Además, cada modo de inicio ejecuta sus propios scripts de inicio. En concreto, dentro del directorio `/etc` existen tantos subdirectorios del tipo `/etc/rc`x`.d` como modalidades de inicio existen (x =1,2,3,4,5). A su vez, cada uno de esos subdirectorios contiene enlaces simbólicos a los scripts localizados en `/etc/init.d` que deseamos ejecutar con cada uno de los modos de inicio. Es decir, si nuestro equipo inicia en modo 3, todos los scripts localizados en `/etc/init.d` que estén asociados a un enlace simbólico dentro de `/etc/rc3.d` serán ejecutados al iniciarse la máquina.

3. En función de cual sea nuestro modo de inicio, supongamos modo 3, nos situaremos en el subdirectorio `/etc/rc`x`.d` correspondiente, que contiene los enlaces simbólicos a los scritps de inicio que serán lanzados al iniciarse:

   ```
   [root@linux]# cd /etc/rc3.d
   ```

   A continuación, situados en `/etc/rc`x`.d`, crearemos un enlace simbólico al script `conf_firewall` que hemos creado en el primer paso, para que sea lanzado al iniciarse el equipo. Para crear el enlace simbólico haremos uso del comando `ln -s` cuya sintaxis es:

[12] Una vez arrancado es posible lanzar la interfaz gráfica, desde una sesión iniciada mediante el comando `startx`
[13] El modo 5 inicia automáticamente la interfaz gráfica al usuario que inicia sesión

```
[root@linux]# ln -s 'fichero a enlazar' 'nombre del enlace simbólico'
```

En relación al nombre del enlace simbólico a crear no puede ser cualquiera, sino que debe seguir una sintaxis concreta para que GNU/Linux entienda que deseamos hacer con él. Esto es debido a que lo normal es que los scripts de inicio estén asociados a la puesta en marcha de determinados servicios. Por ese motivo, el nombre del enlace simbólico deberá comenzar por `S` (start) o `K` (kill) seguido de un numero y un nombre. El numero indica el orden en que será ejecutado en relación al resto de scripts de inicio, y el nombre identifica al script de inicio localizado en `/etc/init.d` que deseamos ejecutar, así la estructura del nombre del enlace simbólico sería,

```
'S|K''numero''nombre script'
```

.

En nuestro caso, como no se trata de un servicio, nos da igual poner `S` o `K`, como número podremos una cantidad alta para que no interfiera con el resto de servicios, y como nombre el del `script`:

```
[root@linux]# ln -s ../init.d/conf_firewall S98conf_firewall
```

4. Por último, reiniciaremos el equipo y comprobaremos que se ha autoconfigurado el firewall.

6.5. Objetivos de la Práctica

La finalidad de la presente práctica es conocer en cierta profundidad las herramientas de las que disponemos en GNU/Linux para llevar a cabo tanto la alteración de los paquetes TCP/IP (nat) como su filtrado (filter) a través de iptables. Aunque las opciones y combinaciones que nos ofrece iptables son innumerables se tratara de mostrar mediante una serie de supuestos prácticos reales sus características más importantes.

Al igual que en otros capítulos del libro, para la implementación y comprobación de los diferentes ejercicios que se proponen se recomienda la utilización de máquinas virtuales VMware evitando de esta forma el coste, tanto monetario como temporal, que conllevaría hacer uso de equipación física.

PRÁCTICA: CONFIGURACIÓN DE UN GATEWAY/FIREWALL

6.6. Configuración de GNU/Linux como firewall

Como ya se ha comentado anteriormente, será necesario instalar el paquete software `iptables`. Su instalación puede llevarse a cabo a través de la interfaz de línea de comandos mediante `urpmi`, o mediante `rpmdrake`[14] en el caso de hacer uso de interfaz gráfica.

1ª Forma: Desde la interfaz de línea de comandos ⇒ `urpmi`.

```
[root@linux]# urpmi iptables
```

2ª Forma: Desde la interfaz grafica ⇒ `rpmdrake`.

```
[root@linux]# rpmdrake &
```

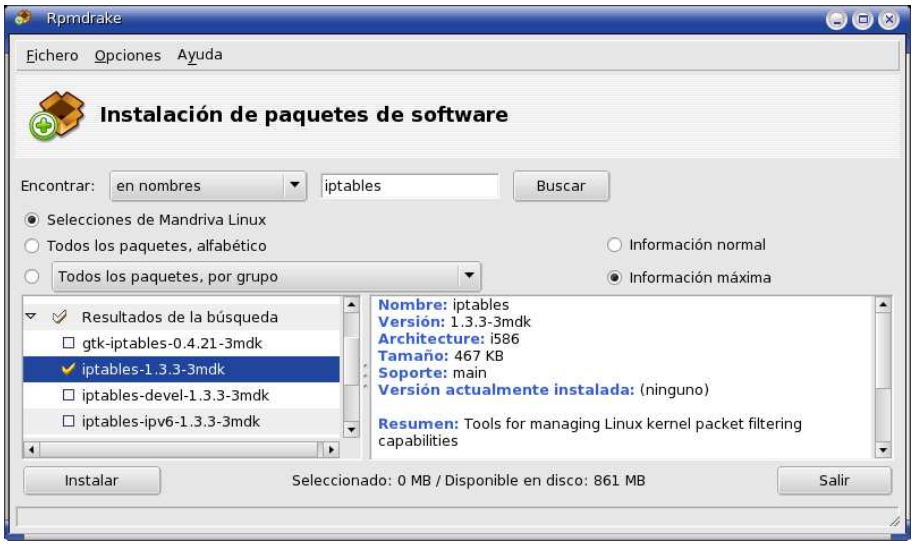


Figura 6.7: Vista de la instalación desde el GUI de Mandriva.

6.6.1. Implementación y comprobación de configuraciones del firewall

A lo largo de la práctica se van a mostrar múltiples situaciones prácticas reales que deberás implementar y comprobar su correcto funcionamiento. Una posibilidad sería implementar la situación en red que se plantee en cada caso mediante los equipos físicos: dispositivos de interconexión y medios de transmisión implicados. El principal inconveniente de esta opción esta en la disponibilidad de equipación. Una segunda opción igualmente válida y más aconsejable sería hacer uso de equipación virtual[15]. Esta opción permitirá disponer de tantos puestos de prácticas como equipos informáticos tengan instalado el software encargado de crear máquinas virtuales. Esto conlleva un ahorro tanto en coste de equipación como en tiempo de montaje. En el apéndice A encontrarás una breve descripción sobre como crear redes virtuales con VMware.

[14] Observa que `rpmdrake` es un programa propio de Mandriva.

[15] La creación de máquinas virtuales puede realizarse con VMware, XEN, ... En este texto hemos optado por VMware.

Ejercicio 6.1

Configura el equipo GNU/Linux `gateway/firewall` para que permita la comunicación entre los equipos que forman parte de las dos redes que interconecta (por ejemplo, 192.168.199.0/24 y 192.168.233.0/24) sin ningún tipo de restricciones o filtrado. Una vez llevada a cabo la configuración anterior, comprueba su correcto funcionamiento, realizando un `ping` desde un equipo de una red a la otra advirtiendo que la comunicación entre ambos es posible.

Solución del ejercicio 6.1

Por motivos de seguridad, por defecto, todo equipo GNU/Linux tiene desactivada la opción de reenvío de paquetes TCP/IP entre las distintas interfaces de red de que disponga. Esto significa que si la función de un `gateway` o `puertadeenlace` es recoger los paquetes que recibe por una de sus interfaz de red, observar la dirección IP del destinatario de estos paquetes, y encaminarlos hacia la interfaz de red adecuada con la finalidad de que puedan alcanzar el destinatario, será necesario activar el reenvío de paquetes TCP/IP.

Para configurar está función de reenvío se dispone de una variable del sistema de red, `net.ipv4.ip_forward`, donde dependiendo de su valor `1` ó `0`, permitirá o no el reenvío de paquetes (su valor por defecto 0, no reenviar). Para asignar un valor a dicha variable será necesario editar el fichero `/etc/sysctl.conf`, para modificarlo. Tras ello será necesario reiniciar los servicios de red mediante alguno de comandos siguientes (son equivalentes):

```
[root@linux]# service network restart
```

```
[root@linux]# /etc/init.d/network restart
```

Otra opción consiste en redireccionar el valor a asignar a la variable `net.ipv4.ip_forward` sobre el fichero virtual `/proc/sys/net/ipv4/ip_forward`. En este caso no será necesario reiniciar los servicios de red,

```
[root@linux]# echo "1" > /proc/sys/net/ipv4/ip_forward
```

.

Además, si tenemos en cuenta que nuestro equipo `gateway` va a hacer de intermediario entre dos redes privadas, tal como se vio al comienzo del cápitulo (sección 6.2), será necesario enmascarar los paquetes que lo atraviesan (`SNAT POSTROUTING`).

```
#Script de Configuración del GATEWAY/FIREWALL: Ejercicio 6.1
#
# 1°.- Permitimos que se puedan reenviar (forward) los paquetes
# entre las distintas interfaces de red del router/gateway GNU/Linux
echo 1 > /proc/sys/net/ipv4/ip_forward
# 2°.- Hacemos NAT postrouting para que los paquetes
# generados en una de las redes que interconecta el Gateway
# al reenviarlos hacia otra red salgan del router/gateway
# (postrouting) con dirección IP de origen la del Gateway
iptables -t nat -A POSTROUTING -j MASQUERADE
```

De esta forma, sin necesidad de configurar nada más, nuestro equipo GNU/Linux ya tendría la capacidad de comunicar las distintas redes que interconecta pasando a comportarse como un auténtico `gateway`. Al mismo tiempo, esto puede traducirse en una amenaza para la seguridad de las redes que interconecta al comportarse nuestro equipo GNU/Linux como un `gateway` totalmente transparente. Es decir, independientemente de quien es el equipo origen y destino de la comunicación y la finalidad de esta[16], el `gateway` permite el estableciendo de la conexión, y el tráfico que conlleva sin ningún tipo de restricción.

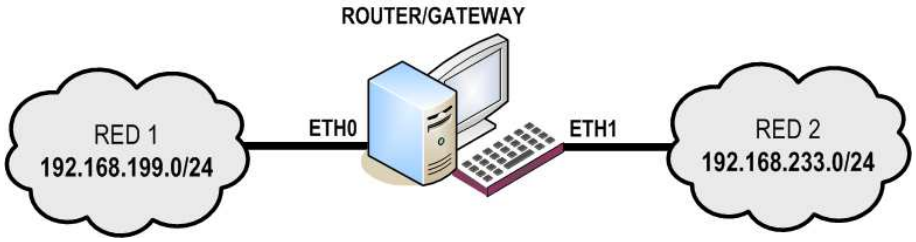


Figura 6.8: Esquema de red del ejercicio 6.1.

Comprobación de la configuración hecha en el ejercicio 6.1:

Para implementar dos redes lógicas comunicadas a través de un equipo que haga de `gateway` y así comprobar el correcto funcionamiento del script de configuración anterior tenemos dos opciones:

1. Montar físicamente la red.
2. Hacerlo virtualmente mediante la ayuda de VMware obteniendo resultados equivalentes (ver apéndice A).

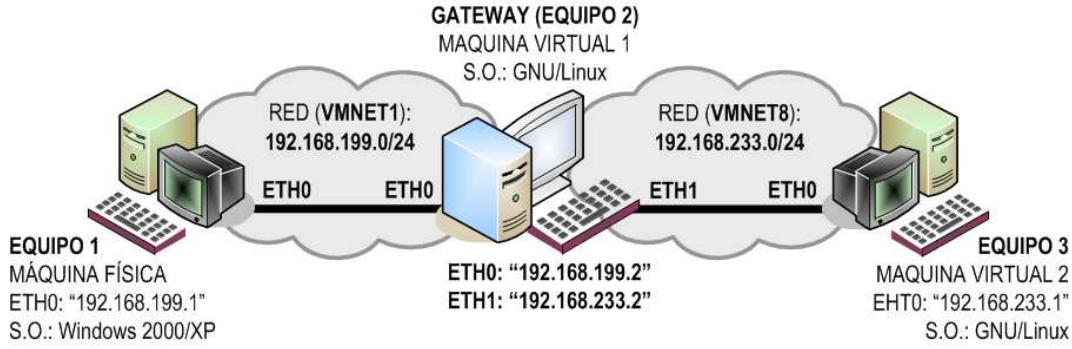


Figura 6.9: Implementación y comprobación del ejercicio mediante VMware.

En el caso de hacer uso de VMware será necesario tener en cuenta los siguientes aspectos:

a) En el equipo 1, la máquina física, asumiendo que corre bajo Windows, deberemos tener habilitada únicamente una única interfaz de red virtual (por ejemplo `VMnet1`), asegurándonos que el resto de interfaces de red virtuales están desactivas. En relación a la dirección IP asignada, tenemos la opción de elegirla manualmente dentro del rango de la red lógica `VMnet` a la que pertenezca (por ejemplo, `192.168.199.0/24`) o en caso de hacer uso de DHCP conocerla mediante la ejecución del comando[17] `ipconfig`.

[16] Por ejemplo, la comunicación puede ir destinada a establecer conexión con un servicio `FTP`, `HTTP`, `DNS`, `IRC`, `SMB`, etc. Con la finalidad de restringir a quienes se les permite establecer la conexión, controlando a la vez la finalidad de esta, posteriormente configuraremos mediante `iptables` el `gateway` para que se convierta al mismo tiempo en un `firewall`.

[17] Para ejecutar comandos en Windows deberás abril la consola MS-Dos llamada también símbolo del sistema.

Además deberemos indicar al equipo 1 que en el caso en que desee comunicarse con un equipo fuera de su red lógica el equipo `gateway` que hará de puerta de enlace será el equipo 2, `192.168.199.2`, ver la figura 6.10.

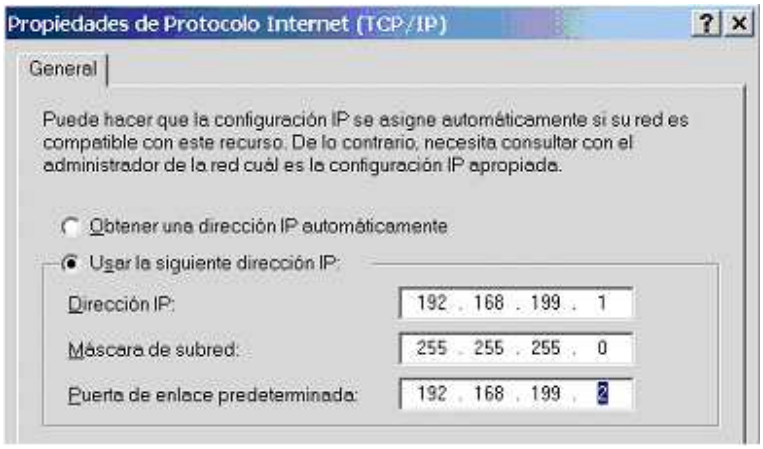


Figura 6.10: Configuración manual de la interfaz de red virtual en Windows.

b) En el equipo 3, la segunda máquina virtual (asumiremos GNU/Linux), deberemos habilitar una interfaz de red virtual dentro de una `VMnet` diferente a la que pertenece el equipo 1 (por ejemplo, `VMnet8`), y al igual que en el equipo 1, deberemos asignar manualmente una dirección IP (mediante `ifconfig` o desde `/etc/sysconfig/network-scripts/`) o conocer mediante `ifconfig` la que se le ha asignado automáticamente vía DHCP. De igual forma, mediante el comando `route` o modificando `/etc/sysconfig/network` le informaremos al equipo 3 que su `gateway` es el equipo 2, `192.168.233.2` (ver capítulo 1).

c) En el equipo 2, la primera máquina virtual (GNU/Linux), deberemos habilitar dos interfaces de red correspondientes a las dos redes virtuales a las pertenecen los equipos 1 y 3 (`VMnet1` y `VMnet8`). Al igual que el equipo 3, deberemos asignar manualmente direcciones IP dentro de las redes `192.168.199.0/24` y `192.168.233.0/24` (por ejemplo, `192.168.199.2` y `192.168.233.2`), o conocer las que se les ha asignado automáticamente. Después deberemos ejecutar el script de configuración del `gateway` mostrado en la solución anterior.

d) Por último, comprobaremos mediante el uso del comando `ping` la comunicación entre los equipos de las redes VMnet1 y VMnet8. También es interesante comprobar que si la variable `ip_forward` en el `gateway` pasa a valer 0 dejan de comunicarse.

Ejercicio 6.2

Mediante el uso de `iptables` configura un equipo servidor HTTP/FTP GNU/Linux de tal forma que se garanticen las siguientes restricciones de acceso:

- Debe admitir las solicitudes de conexión al servicio `http`, protocolo `tcp` puerto 80, independiente de la interfaz por la que sean recibidas y del equipo que las realiza.
- Debe restringirse el acceso al servicio `ftp`, protocolo `tcp/udp` puerto 21, de tal forma que solamente sean aceptadas las solicitudes de conexión recibidas a través de las interfaces de red `eth1`, `eth2` y `eth3`, garantizando que sólo tengan acceso a este servicio los equipos pertenecientes a las redes privadas/internas `192.168.0.0/16`.

- El resto de puertos del equipo deberán permanecer cerrados con la finalidad de evitar accesos indeseados.

Solución del ejercicio 6.2

Tras leer el enunciado del supuesto práctico debemos advertir que lo que se pretende es filtrar la entrada al equipo servidor http/ftp decidiendo a quien debemos dejar acceder y a quien no. Para ello haremos uso del software `iptables` en modo `filter` a través de las reglas de filtrado de la lista de `INPUT` (`iptables -t filter -I/A INPUT`).

```
# Script de Configuración del FIREWALL en el servidor (INPUT): EJERCICIO 6.2
#
# 1º.- Permitimos la entrada de paquetes dirigidos al servicio
# Web (http), protocolo tcp:puerto 80, independientemente de la
# interfaz por la que sean recibidas las solicitudes, y quien sea
# el solicitante (independientemente de la dirección IP de origen):
iptables -t filter -A INPUT -p tcp --dport 80 -j ACCEPT
# 2º.- Evitamos explicitamente el acceso a través de la interfaz
# eth0 que esta en contacto con la red pública (Internet) al
# servicio FTP: protocolo tcp/upd:puerto 21.
iptables -t filter -A INPUT -i eth0 -p tcp --dport 21 -j DROP
iptables -t filter -A INPUT -i eth0 -p udp --dport 21 -j DROP
# 3º.- Permitimos el acceso al servicio FTP desde cualquier
# otra interfaz de red (Intranet - servicio privado) para las
# redes privadas 192.168.0.0/16:
iptables -t filter -A INPUT -s 192.168.0.0/16 -p tcp --dport 21 -j ACCEPT
iptables -t filter -A INPUT -s 192.168.0.0/16 -p udp --dport 21 -j ACCEPT
# 4º.- Por último, establecemos como política por defecto
# denegar todo intento de conexión al resto de puertos/servicios
# del equipo servidor GNU/Linux :
iptables -t filter -P INPUT DROP #Resto de puertos cerrados
```

Comprobación de la configuración hecha en el ejercicio 6.2:

Como ya se ha comentado, se recomienda hacer las comprobaciones utilizando VMware. En la figura 6.12 se muestra la configuración propuesta; advierte que la red virtual `VMnet1` emula la red pública Internet.

Al igual que en el ejercicio anterior, será necesario asignar a cada equipo su dirección IP manualmente o tomar las que se les asigna automáticamente vía DHCP. Será necesario habilitar tres redes virtuales[18] (por ejemplo, `VMnet1`, `VMnet3` y `VMnet8`). Nota que con esta configuración son necesarios elevados requerimientos de hardware para la máquina física. Si no cumpliéramos los requerimientos exigidos por `VMware` sería necesario introducir algún equipo físico más.

Una vez implementado, para comprobar el funcionamiento deberemos configurar la máquina virtual 1 (equipo 4) para que haga de servidor `Web` y `FTP` mediante la instalación y posterior configuración básica de `apache` y `proftpd`. Posteriormente tan sólo será necesario ejecutar en los equipos clientes (equipos 1, 2 y 3) el correspondiente software cliente Web y FTP (por ejemplo `Mozilla Firefox`) y lanzar peticiones al equipo que hace de servidor comprobando si este acepta las solicitudes de conexión o no.

[18] En el apéndice A puede verse una estructura similar a la propuesta.

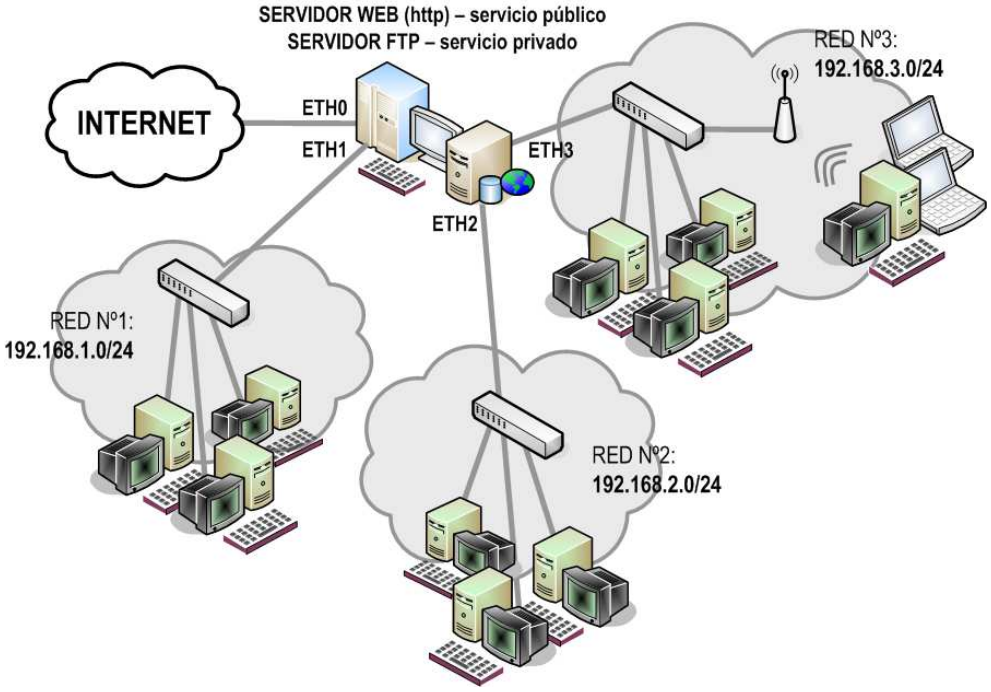


Figura 6.11: Esquema de red del ejercicio 6.2

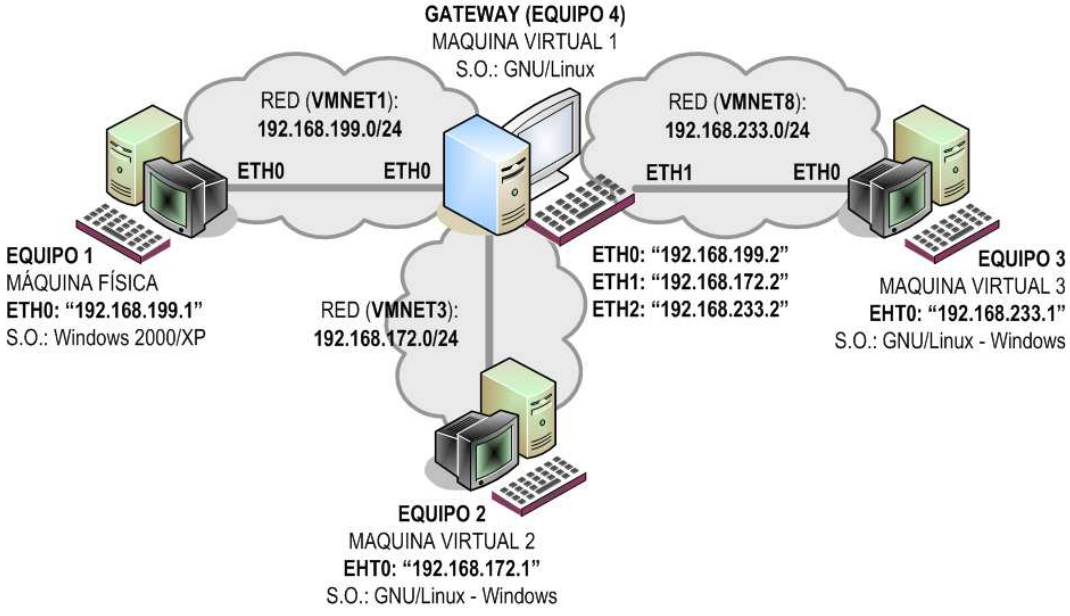


Figura 6.12: Implementación y comprobación del ejercicio 6.2 mediante VMware.

Ejercicio 6.3

Configura un equipo GNU/Linux como `gateway/firewall` para que establezca un filtrado en las comunicaciones que se establecen entre los equipos pertenecientes a las redes que interconecta (por ejemplo, `129.168.199.0/24` y `129.168.233.0/24`), permitiendo únicamente establecer conexiones desde la red `129.168.199.0/24` a los recursos compartidos por los servidores samba (protocolo `NetBIOS/SMB` de Microsoft Windows) que se encuentran en la red `192.168.233.0/24`, pero no al revés. Además, habrá que evitar el reenvío de paquetes dirigidos a otros servicios/-puertos.

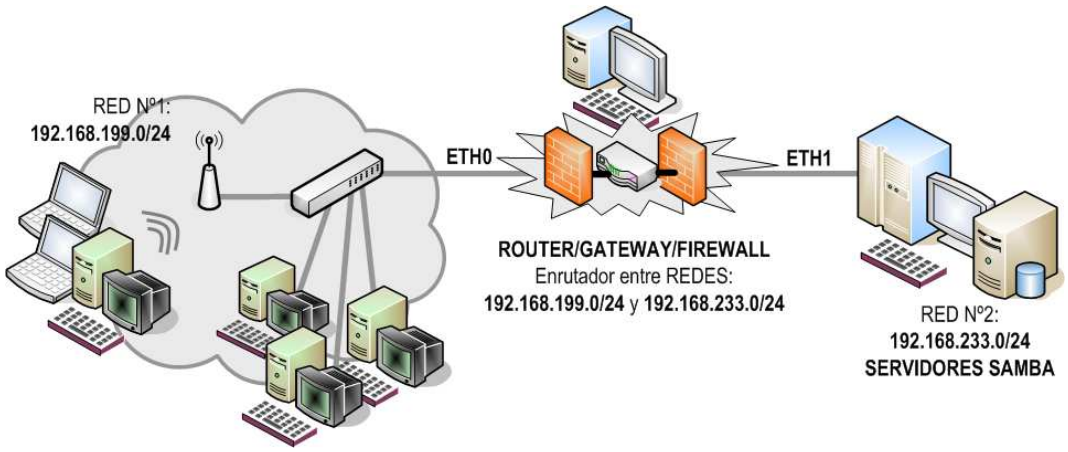


Figura 6.13: Esquema de red del ejercicio 6.3

Solución del ejercicio 6.3

La situación planteada es similar al ejercicio práctico N°1 con la diferencia de que en este caso no reenviamos todo lo que le llega al `gateway`. En este caso vamos a establecer un filtrado (iptables -t filter) reenviando (FORWARD) únicamente aquellos paquetes TCP/IP que estén relacionados con el servicio NetBIOS/SMB usado en redes Microsoft para compartir recursos. Para poder discriminar que paquetes van asociados a dicho protocolo NetBIOS/SMB deberemos conocer antes sus puertos de comunicaciones asociados.

¡¡Ayuda!! Los puertos `tcp/udp` utilizados por las redes Microsoft Windows y GNU/Linux/Samba con la finalidad de compartir recursos (carpetas/unidades de disco/impresoras) en red son:

- Puerto 137: `netbios-ns`. `NetBios Name Services`, es el puerto a través del cual se realiza la resolución de nombres NetBios en redes Windows de Microsoft y GNU/Linux/Samba (no confundir con los nombres de dominio).
- Puerto 138: `netbios-dgm`. `NetBios Datagram Services`, es el puerto a través del cual se realiza la transferencia de datos/información.
- Puerto 139: `netbios-ssn`. `NetBios Session Services`, puerto encargado de establecer, mantener activa, y liberar la sesión NetBios entre dos equipos.

Además, para cumplir las exigencias del enunciado del supuesto práctico deberemos discriminar entre paquetes relacionados con el establecimiento de la comunicación, y paquetes asociados a una conexión ya establecida, ya que debemos permitir el tráfico de datos en ambos sentidos, pero sólo admitir conexiones en un sentido. Es decir, antes de transferirse datos entre los equipos de la

red `192.168.199.0/24` y los servidores de la red `192.168.233.0/24`, los equipos cliente deben hacer una solicitud de conexión al servicio, y posteriormente los servidores aceptarla. Es entonces cuando puede comenzar el trafico de paquetes asociados a los datos que se intercambien. Según el enunciado, aunque hubiese servidores de archivos SAMBA en ambas redes sólo debería permitirse a la red 1 conectarse a los de la red 2, pero en ningún caso al reves. Esto se produce en situaciones reales por cuestiones de confidencialidad. Para tener en cuenta esta discriminación mencionaremos dos posibilidades:

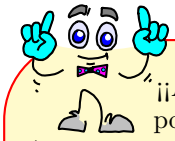
1ª forma.- Puede diferenciarse entre aquellos paquetes que van dirigidos a establecer una conexión de los que son simplemente datos a través de unos `flag` que existen en la cabecera de los paquetes TCP/IP: `ACK`, `SYN` y `FIN`. En concreto, se reconocen los paquetes que intentan establecer una conexión por tener el flag `SYN` activado y los flags `ACK` y `FIN` desactivados ⇒ opción `--syn` del protocolo tcp (-p tcp). Según esto podemos establecer una regla de filtrado mediante iptables haciendo uso de la opción `--syn` que evite que paquetes `tcp` orientados a conexión recibidos por la interfaz `eth1` (procedentes de la red `192.168.233.0/24`) no sean reenviados (`-jDROP/REJECT`):

```
[root@Linux]  iptables -t filter -A FORWARD -i eth1 -p tcp --syn
                                    -j REJECT --reject-with icmp-port-unreacheable
```

2ª forma.- En la *1ª forma* se ha descrito como reconocer los paquetes TCP/IP encargados de establecer una conexión. Esta *2ª forma* consistirá en detectar los paquetes que están relacionados con una conexión ya establecida. Puede observase el estado de los paquetes TCP/IP mediante la opción `-m state` y comprobar si son paquetes asociados a una conexión previamente establecida a través de los atributos `RELATED` y `ESTABLISHED`:

```
[root@Linux]  iptables -t filter -A FORWARD -i eth0 -o eth1
                                     -p tcp --dport 137:139 -j ACCEPT
[root@Linux]  iptables -t filter -A FORWARD -i eth0 -o eth1
                                     -p udp --dport 137:139 -j ACCEPT
[root@Linux]  iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Advierte que con las dos primeras instrucciones (reglas) únicamente aceptamos (`-j ACCEPT`) reenviar (`FORWARD`) a través del `gateway` aquellos paquetes TCP/IP relacionados con el servicio NetBIOS (`-p tcp --dport 137:139`) entrantes por la interfaz `eth0` (`-i eth0`) y salientes por `eth1` (`-o eth0`). Para evitar conexiones en sentido opuesto (`-i eth1 -o eth0`) no se permite el reenvío de paquetes TCP/IP en sentido contrario, excepto aquellos que sean generados como respuesta (del servidor hacia los clientes) a las conexiones establecedas (`-m state --state ESTABLISHED,RELATED -j ACCEPT`). Por esta razón, la última instrucción acepta el reenvío (-j ACCEPT) de todos esos paquetes independientemente de la interfaz de red de entrada y salida.



¡¡Aclaración!! En comunicaciones informáticas dentro de la capa de transporte del modelo TCP/IP existen básicamente dos tipos de protocolos: `TCP` (Transport Control Protocol) y `UDP` (User Datagram Protocol). Ambos se diferencian básicamente en que el protocolo `TCP` esta orientado a conexión y el `UDP` no.
Esto se traduce, en que para establecer una comunicación vía `TCP` es necesario en primer lugar realizar una solicitud y obtener la correspondiente aceptación de conexión al servicio correspondiente, para que posteriormente pueda existir un tráfico de datos. En concreto, el equipo que desea iniciar la sesión sobre un servicio haciendo uso del protocolo `TCP` se caracteriza porque debe enviar un paquete TCP/IP con el `flag` (indicador) de sincronización (SYN) activado. Otra característica importante de los servicios que hacen uso del protocolo `TCP` es que el receptor de los datos informa al emisor mediante `ACK` (ACKnowledgment) del estado de recepción de los datos (acuse de recibo), lo que ayuda a que la comunicación que se establezca entre emisor y receptor sea más fiable garantizando una secuencia correcta de entrega de los paquetes. Los servicios más destacados que hacen uso del protocolo `TCP` son:

FTP (puerto tcp 21) → Protocolo utilizado en la transferencia de ficheros.

TELNET (puerto tcp 22) → Utilizado en el control remoto de equipos.

HTTP (puerto tcp 80) → Protocolo utilizado en la navegación Web.

NETBIOS-SSN (puerto tcp 139) → Servicio de sesión NetBIOS.

NTP (puerto tcp 123) → Utilizado por los servidores de horas.

Por el contrario, los servicios que hacen uso del protocolo `UDP` no requieren ni un establecimiento previo de conexión ni emitir acuses de recibo (ACK), lo cual se traduce en que la comunicación establecida sea más agil, pero sin presentar una garantía de entrega. Por este motivo, los servicios que hacen uso de este tipo de protocolo se caracterizan por transmitir pequeñas cantidades de datos, de tal forma que en caso de fallo, el coste de un reenvió no sea alto. Los servicios más destacados que hacen uso del protocolo `UDP` son:

NETSTAT (puerto udp 15) → Utilizado para comprobar el estado de la red.

DNS (puerto udp 53) → Utilizado para establecer conversiones de nombres de dominio a direcciones IP, e inversa (DNS, Domain Name Server).

TFTP (puerto udp 69) → Trivial File Transfer Protocol.

NETBIOS-NS (puerto udp 137) → Servicio de nombres NetBIOS.

NETBIOS-DGM (puerto udp 138) → Servicio de datagramas NetBIOS.

SNMP (puerto udp 161) → Utilizado para gestión del estado de las redes.

NTP (puerto udp 123) → Utilizado por los servidores de horas. Observa que el Network Time Protocol hace uso tanto por `udp` como por `tcp`, característica que se suele dar en muchos servicios.

Según todo lo anterior, un posible script que podríamos ejecutar en el `gateway` que cumpla con las especificaciones del supuesto práctico sería el siguiente:

```
#Script de Configuración del GATEWAY/FIREWALL: EJERCICIO 6.3
#
# 1º.- Configuramos el GNU/Linux como Gateway:
echo 1 > /proc/sys/net/ipv4/ip_forward
# 2º.- Hacemos NAT postrouting para que el gateway
# enmascare los paquetes:
iptables -t nat -A POSTROUTING -j MASQUERADE
```

Continúa en página siguiente

```
# 3º.- Borramos las reglas de filtrado eliminando cualquier
# configuración previa del FIREWALL:
iptables -t filter -F
# 4º.- Permitimos el reenvió de paquetes asociados al servicio
# NetBIOS/Samba (puertos tcp y udp 137 netbios-ns,
# 138 netbios-dgm, 139 netbios-ssn) procedentes de la red 192.168.199.0/24:
iptables -A FORWARD -i eth0 -o eth1 -s 192.168.199.0/24 -p tcp
                                              --dport 137:139 -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -s 192.168.199.0/24 -p udp
                                              --dport 137:139 -j ACCEPT
# 5º.- Permitimos el trasiego a través del enlace de paquetes
# asociados a conexiones previamente establecidas:
iptables -A FORWARD -m state -state ESTABLISHED,RELATED -j ACCEPT
# 6º.- Si queremos que puedan hacerse ping (protocolo ICMP,
# Internet Control Message Protocol) entre los equipos de las distintas
# redes que interconecta el gateway para comprobar la comunicación:
iptables -A FORWARD -p ICMP -j ACCEPT
# 7º.- Establecemos como política por defecto no reenviar
# (FORWARD DROP) ningún paquete a excepción de aquellos paquetes
# permitidos por las reglas del FIREWALL anteriores. De esta forma nos
# aseguramos que el resto de puertos/servicios son inaccesibles:
iptables -t filter -P FORWARD DROP
```

a

Comprobación de la configuración hecha en el ejercicio 6.3:

Para implementar y comprobar el correcto funcionamiento del firewall puede hacerse uso del montaje utilizado en el primer ejercicio.

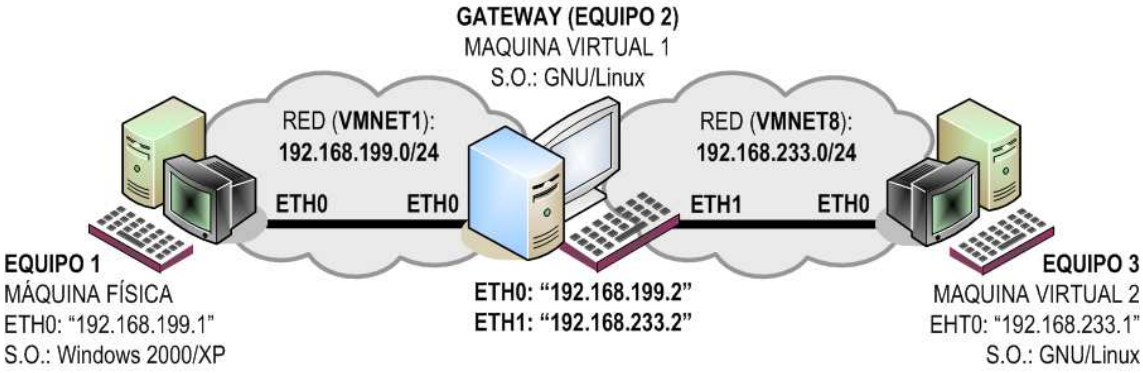


Figura 6.14: Implementación y comprobación del ejercicio 6.3mediante VMware.

En este caso será necesario configurar el equipo 3 (máquina virtual 2) para que haga de servidor de archivos usando el protocolo de Microsoft Windows. Es decir, en el caso de que este equipo funcione sobre GNU/Linux será necesario instalar y configurar el paquete software `samba`[19].

[19] ver el libro de **Instalación y Mantenimiento de servicios de redes de área local** de los mismos autores.

Ejercicio 6.4

Configura un equipo GNU/Linux como `gateway/firewall` de un segmento de red dentro una Intranet protegida (por ejemplo, `192.168.2.0/24`), de tal forma que permita a los equipos que lo forman salir a exterior (extranet) pudiendo establecer conexiones únicamente sobre servidores `HTTP` (puerto tcp 80) y `FTP` (puertos tcp/udp 21).

Esta situación suele corresponderse con todas aquellas organizaciones que poseen un segmento de red que desean proteger permitiéndo a sus equipos hacer conexiones desde la zona protegida hacia los servicios ofrecidos en el exterior (extranet).

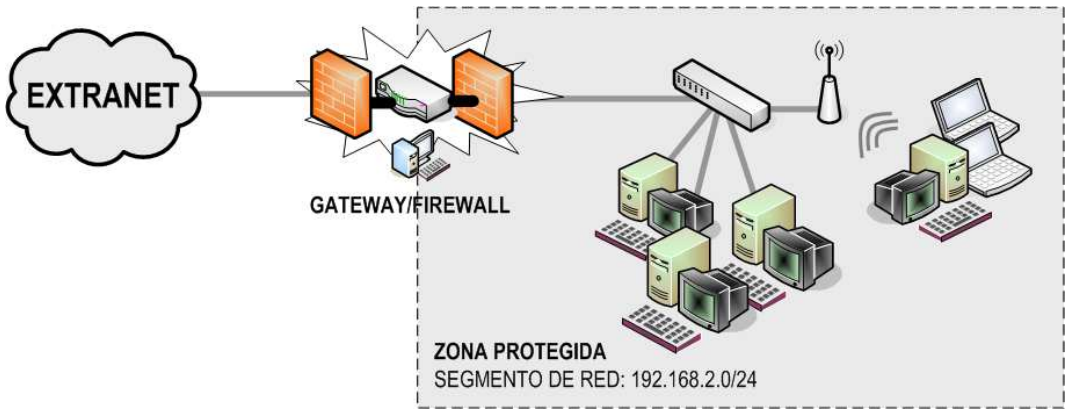


Figura 6.15: Esquema de red del ejercicio 6.4.

Solución del ejercicio 6.4

Es fácil advertir que el problema planteado en este ejercicio es similar al anterior diferenciándose únicamente en el tipo de servicios que se desean controlar. La único que habría que tener en cuenta es que si el `gateway/firewall` debe permitir a los equipos de la intranet navegar por la Web haciendo uso de nombres de dominio (por ejemplo, `www.iesbajoaragon.com`, `www.iesrioarba.com`, `www.cossio.net`, etc.) será necesario abrir, además de los puertos 21/`ftp` y 80/`http`, el puerto que da servicio a la resolución de nombres de dominio (DNS) 53/dns (protocolo udp).

Por tanto, el script de configuración del equipo `gateway/firewall` va a ser muy parecido al del ejercicio anterior. Es importante destacar el papel que desarrolla el equipo `gateway/firewall` enmascarando los paquetes que se generan en la intranet. Intercambia la dirección IP de origen por la suya, `SNAT/MASQUERADE` de tal forma que pueda ser reconocida la dirección de origen de estos paquetes fuera del segmento de red de la intranet:

```
# Script de Configuración del GATEWAY/FIREWALL: EJERCICIO 6.4
#
# 1º.- Configuramos GNU/Linux como Gateway,
# permitiéndole reenviar paquetes
echo 1 > /proc/sys/net/ipv4/ip_forward
# 2º.- Borramos las reglas eliminando cualquier
# configuración previa del FIREWALL:
iptables -t filter -F
iptables -t nat -F
# 3º.- Hacemos NAT postrouting: Garantizamos que los paquetes
# que salen del gateway lo hagan con la dirección IP de origen
```

Continúa en página siguiente

```
# correspondiente a la interfaz de salida de este.
iptables -t nat -A POSTROUTING -j MASQUERADE
# 4º.- Permitimos el reenvio de todos aquellos paquetes relacionados con
# los servicios HTTP, FTP y DNS:
iptables -t filter -A FORWARD -i eth1 -p tcp --dport 80 -j ACCEPT
iptables -t filter -A FORWARD -i eth1 -p tcp --dport 21 -j ACCEPT
iptables -t filter -A FORWARD -i eth1 -p udp --dport 21 -j ACCEPT
iptables -t filter -A FORWARD -i eth1 -p udp --dport 53 -j ACCEPT
# 5º.- Reenviamos igualmente todos aquellos paquetes de respuesta
# procedentes de la extranet que estén relacionados con las conexiones
# establecidas por los equipos clientes de la intranet:
iptables -t filter -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
# 6º.- Establecemos como política por defecto no reenviar
# (FORWARD DROP) ningún paquete, a excepción de aquellos que se le
# indiquen mediante las reglas anteriores al FIREWALL:
iptables -t filter -P FORWARD DROP
```

Observa que el `script` anterior contiene reglas que son muy similares. Tienen la forma,

```
iptables -t filter -A FORWARD -i eth1 ...-j ACCEPT
```

Con la finalidad de simplificar estos scripts de configuración, `iptables` dispone de la opción `-m multiport`, la cual nos permite hacer referencia a una lista de puertos en la misma instrucción:

```
#Script de Configuración del GATEWAY/FIREWALL: EJERCICIO 6.4
#
# 1º.-Configuramos el GNU/Linux como Gateway
echo 1 > /proc/sys/net/ipv4/ip_forward
# 2º.-Borramos las reglas (la posible configuración del FIREWALL)
iptables -t filter -F
iptables -t nat -F
# 3º.-Reglas de filtrado FORWARD y de nat POSTROUTING (SNAT/MASQUERADE):
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -t filter -A FORWARD -i eth1 -p tcp -m multiport --dport 80,21 -j ACCEPT
iptables -t filter -A FORWARD -i eth1 -p udp -m multiport --dport 21,53 -j ACCEPT
iptables -t filter -A FORWARD -m state -state ESTABLISHED,RELATED -j ACCEPT
iptables -t filter -P FORWARD DROP
```

Comprobación de la configuración hecha en el ejercicio 6.4:

Un posible montaje que permite la comprobación del correcto funcionamiento de la configuración propuesta sería el mostrado en la figura 6.16

El equipo 1 (máquina física) hace las funciones de servidor de externo. En el caso de que sea un equipo que funciona bajo Windows, una posibilidad es hacer uso de los componentes de Microsoft Windows `IIS` (Internet Information Server) y `servicios de red` (DNS), los cuales permiten configurar de manera sencilla un equipo Windows como servidor DNS, Web y FTP[20].

El equipo 2 (máquina virtual 1) bajo GNU/Linux hará de `gateway/firewall` para lo cual necesitará tener habilitadas dos interfaces de red, cada una de ellas en contacto con una de las redes virtuales que se forman (por ejemplo, `12.0.0.0/8` y `192.168.2.0/24`).

El equipo 3 (máquina virtual 2) bajo GNU/Linux hará la función de cliente DNS, Web y FTP del segmento de red. Mediante la asignación de diferentes direcciones IP para el mismo equipo podemos engañar al equipo `gateway/firewall`, haciéndole creer que el segmento de red está compuesto por

[20] Ver el libro de **Instalación y mantenimiento de servicios de redes de área local** de los mismos autores.

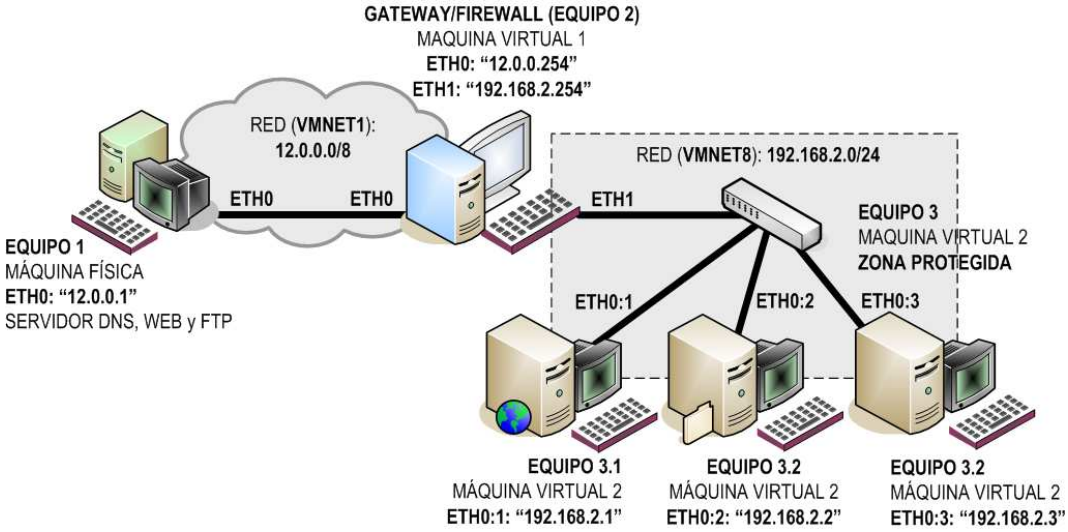


Figura 6.16: Implementación y comprobación del ejercicio 6.4mediante VMware

varios clientes.

Una vez configurado el esquema de red anterior, la comprobación de su correcto funcionamiento puede limitarse a navegar desde un equipo cliente de la intranet por los sitios Web o directorios que se encuentren colgados del servidor Web/FTP (equipo 1).

¡¡IMPORTANTE!! Aclararemos algunos de los aspectos ya comentados a lo largo del ejercicio práctico:

- A la hora de configurar el `gateway/firewall` deberemos tener en cuenta que además de `abrir` los puertos `http` (puerto tcp 80) y `ftp` (puerto tcp/udp 21), para poder establecer conexiones sobre servidores externos a través de su correspondiente nombre de dominio (por ejemplo `ftp.iestiemposmodernos.com`, `www.iesrioarba.com`, etc.) será necesario abrir el `puerto udp 53` correspondiente al servicio DNS para que el servidor de nombres de dominio correspondiente (servidores definidos en el fichero `/etc/resolv.conf` del equipo cliente) pueda informar a los equipos clientes de la dirección IP asociada a los equipos servidores a los que pretenden acceder.
- Por otro lado, deberemos tener en cuenta que si los paquetes TCP/IP procedentes de la intranet son reenviados sin más por el equipo `gateway` hacia el correspondiente equipo servidor, la comunicación no podrá establecerse al desconocer el servidor la ubicación física del equipo cliente, al tratarse de una dirección IP de origen privada (red privada). Para solucionar el problema, el equipo `gateway` tendrá que hacer de intermediario entre ambos sustituyendo la dirección IP de origen del paquete que realiza la solicitud de conexión al servicio por la suya (dirección IP de la interfaz de salida del router), la cual si que será reconocida por el equipo servidor al tratarse de una dirección IP dentro de su misma red lógica. Esta traslación de direcciones IP realizada por el equipo `router/gateway/firewall` es lo que se denomina `nat POSTROUTING` o `SNAT` (`Source Network Address Translation`, Translación de Dirección de Red de Origen). Es decir, una vez que el paquete IP emitido desde el equipo cliente ya ha sido reenviado o enrutado por el `gateway` hacía la interfaz de red de este que le permitirá alcanzar el equipo destino, justo antes de sacarlo, cambia la dirección IP de origen del equipo cliente por la suya.

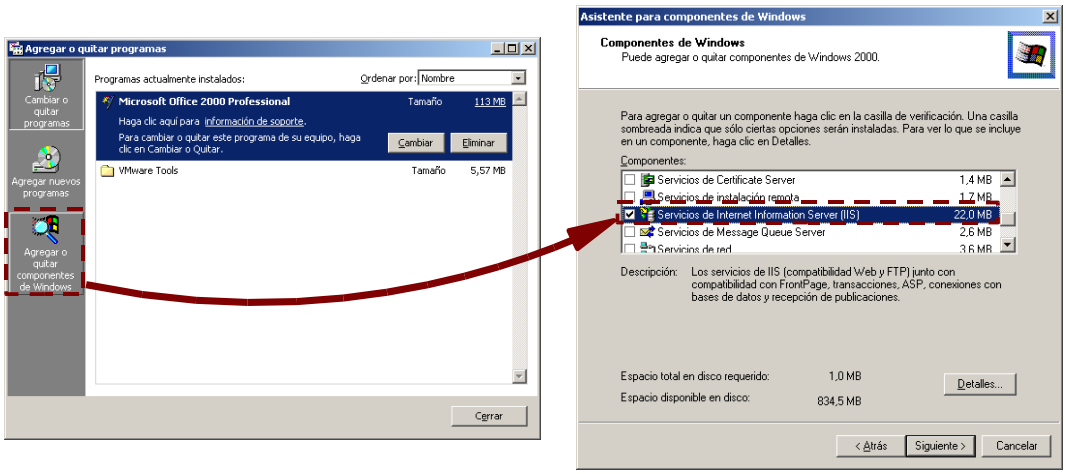


Figura 6.17: IIS nos permite configurar Windows como servidor Web y FTP.

Ejercicio 6.5

Configura un equipo GNU/Linux como `gateway/firewall` de una intranet (segmento de red `192.168.100.0/24`) para que el servidor de archivos (equipo GNU/Linux con `samba` con dirección IP `192.168.100.1`) de esta intranet sea accesible desde la exterior (extranet). Desde el exterior la intranet es transparente siendo el único equipo accesible el propio `gateway`.

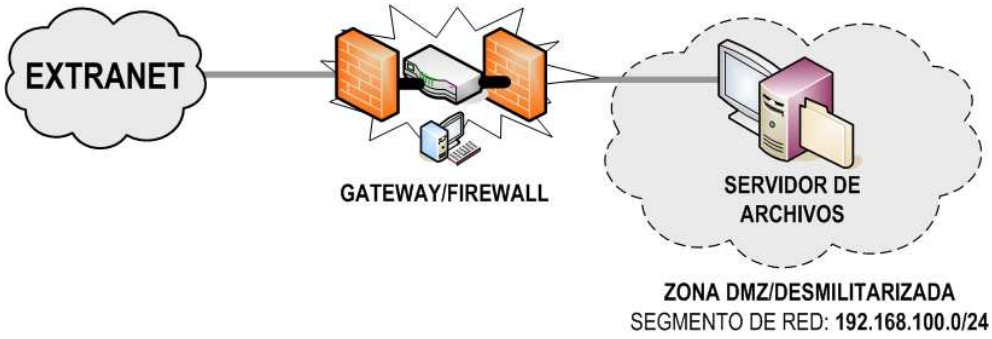


Figura 6.18: Esquema de red del ejercicio 6.5.

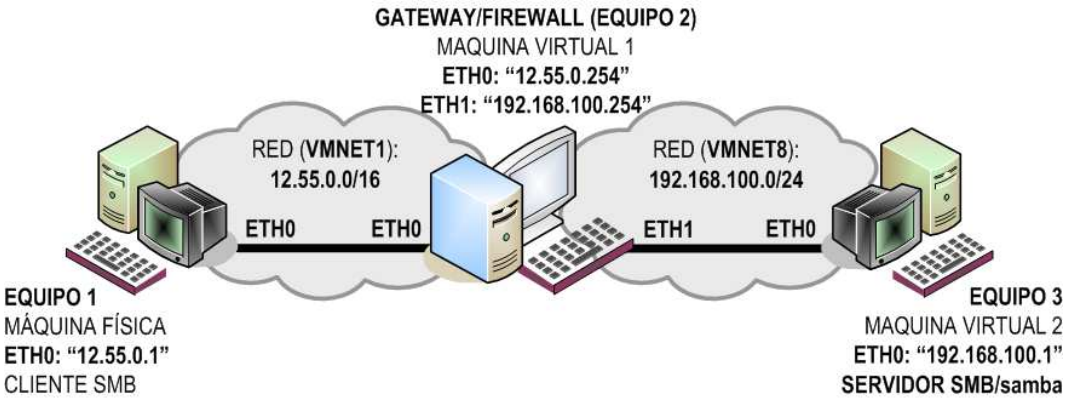
Solución del ejercicio 6.5

Esta situación suele corresponderse en la mayoría de los casos con aquellas organizaciones que ofrecen sus servicios a Internet (equipos con direcciones IP privadas) a través de la dirección IP pública asignada al `router/firewall` suministrado por el proveedor de servicios de Internet. También puede corresponderse con aquellos segmentos de red interna de una empresa a los cuales puede accederse pero de una manera restringida.

Para la implementación y posterior comprobación podría llevarse a cabo la configuración mostrada en la siguiente figura mediante el uso de máquinas virtuales.

El servidor de archivos `192.168.100.1` pertenece a un segmento de red privado (`192.168.100.0/24`). Esto significa que para un equipo exterior (por ejemplo, con dirección IP `12.55.0.1`), tal equipo servidor pasa inadvertido. Por este motivo, para garantizar el acceso a ese servidor, el `gateway` tendrá que desempeñar el papel de intermediario entre los clientes que desean acceder al servicio y el propio servidor. Es decir, las peticiones de conexión al servidor serán enviadas a la

dirección IP del equipo `gateway` (`12.55.0.254`), y este será quien traslade la petición al equipo servidor.



En concreto, si un equipo cliente (equipo 1, `12.55.0.1`), desea conectarse con el servidor samba (equipo 3, `192.168.100.1`), deberá hacer una solicitud de conexión por los puertos NetBios/smb (puertos tcp/udp 137, 138 y 139) al equipo `gateway` (equipo 2, `12.55.0.254`). Este, al detectar tal solicitud advertirá que el equipo que da ese servicio no es él, sino el equipo 3, por lo que alterará el paquete TCP/IP intercambiando la dirección IP de destino suya por la del equipo servidor (`12.55.0.254` ⇒ `192.168.100.1`). A continuación, enrutará dicho paquete hacia la interfaz de red de salida adecuada (eth1) y se lo entregará a su destinatario legítimo. Es decir, hará `nat PREROUTING` o `DNAT` (`Destination Network Address Translation`, Translación de Dirección de Red de Destino). El servidor a su vez contestará al `gateway`, y este a su vez devolverá la respuesta al equipo cliente exterior. El `gateway` no es más que un intermediario entre ambos, haciendo creer a cada uno de los extremos de la comunicación que es el servidor del equipo 1 y el cliente del equipo 3. Todo ello gracias a `nat` (Traslaciones de direcciones IP) `postrouting` y `prerouting`.

Según esto, en el script de configuración del `gateway/firewall` deberán aparecer las siguientes tipos de reglas (iptables):

Reglas de `nat` POSTROUTING (SNAT) → con la finalidad de garantizar que los paquetes de respuesta puedan llegar a su destino, el `gateway` deberá intercambiar la dirección IP de origen de los paquetes TCP/IP que reenvíe colocando en su lugar la dirección IP de su interfaz de salida. Es importante señalar que esta alteración de los paquetes TCP/IP sólo será necesaria en entornos de redes privadas, ya que en el momento que el equipo `router/gateway` se encargue de comunicar redes públicas, todas las direcciones IP que se manejan son públicas y por tanto reconocidas y contempladas en todas las tablas de enrutamiento de los `router` garantizando que estos paquetes puedan llegar a su destino.

Reglas de `nat` PREROUTING (DNAT) → en aquellas situaciones en que los servicios ofrecidos se encuentren inaccesibles (direcciones IP privadas) será necesario un `gateway` con una dirección IP reconocida por el equipo cliente que encubra dichos servicios y asuma la recepción de toda petición de conexión a los mismos. Cada una de estas peticiones será reenviada al correspondiente equipo servidor.

Reglas de `filter` FORWARD → Teniendo en cuenta que un servidor público esta expuesto a todo tipo de amenazas y ataques es necesario intentar disminuir su vulnerabilidad. Para ello el `router/gateway/firewall` que hace de intermediario se encargará de filtrar las solicitudes de conexión aceptando únicamente el reenvío de aquellos paquetes dirigidos a los servicios permitidos.

```
# Script de Configuración del GATEWAY/FIREWALL: EJERCICIO 5
#
# 1º.- Configuramos el GNU/Linux como Gateway
echo 1 > /proc/sys/net/ipv4/ip_forward
# 2º.- Borramos las reglas eliminando cualquier configuración previa del FIREWALL
iptables -t filter -F
iptables -t nat -F
# 3º.- Hacemos NAT POSTROUTING:
iptables -t nat -A POSTROUTING -j MASQUERADE
# 4º.- Hacemos NAT de tipo PREROUTING al recibir el gateway una
# petición smb (puertos 137, 138 y 139), intercambiando su
# dirección IP por la del servidor interno que ofrece el servicio:
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 137:139
                                              -j DNAT --to 192.168.100.1
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 137:139
                                              -j DNAT --to 192.168.100.1
# 5º.- Permitimos el reenvió de todos aquellos paquetes dirigidos al
# servidor de archivos.
iptables -t filter -A FORWARD -i eth0 -p tcp --dport 137:139 -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -p udp --dport 137:139 -j ACCEPT
# 6º.- Permitimos el reenvío de todos aquellos paquetes emitidos
# por el servidor samba como respuesta a las conexiones
# previamente ya establecidas:
iptables -t filter -A FORWARD -m state -state ESTABLISHED,RELATED -j ACCEPT
# 7º.- Establecemos como política por defecto no reenviar (FORWARD DROP)
# ningún paquete, a excepción de aquellos paquetes que se le indiquen
# mediante reglas del FIREWALL, cerrando el resto de puertos/servicios
# para disminuir la vulnerabilidad del servidor:
iptables -t filter -P FORWARD DROP
```

Ejercicio 6.6

Configura un equipo GNU/Linux como `gateway/firewall` de tal forma que haga de `gateway` entre Internet y una intranet. La intranet a su vez se divide en dos zonas, una totalmente protegida contra posibles conexiones procedentes del exterior, y otra encargada de ofrecer sus servicios (DNS, HTTP y FTP) de una forma transparente a los equipos clientes de Internet haciéndoles creer que quien ofrece los servicios es el propio equipo GNU/Linux `gateway/firewall`.

Solución del ejercicio 6.6

Esta situación suele darse en muchas empresas con la finalidad de enmascarar los servicios ofrecidos por diferentes equipos servidores internos (DMZ, zona DesMilitariZada) a través de una única dirección IP pública, la del `router/gateway/firewall`, protegiendo al mismo tiempo a una intranet contra cualquier intruso externo.

De nuevo, la función principal del equipo `gateway` va a ser hacer de intermediario entre dos redes, en este caso, la Internet e intranet. En concreto, deberá cumplir las siguientes exigencias:

1. Debe permitir la salida hacia Internet de todos los equipos pertenecientes a la red privada interna (intranet) sin ningún tipo de restricciones para lo cual será necesario hacer NAT (POSTROUTING/SNAT) sobre su interfaz de salida `eth0`.

2. Debe permitir que desde el exterior (Internet) pueda accederse a los servicios ofrecidos por la

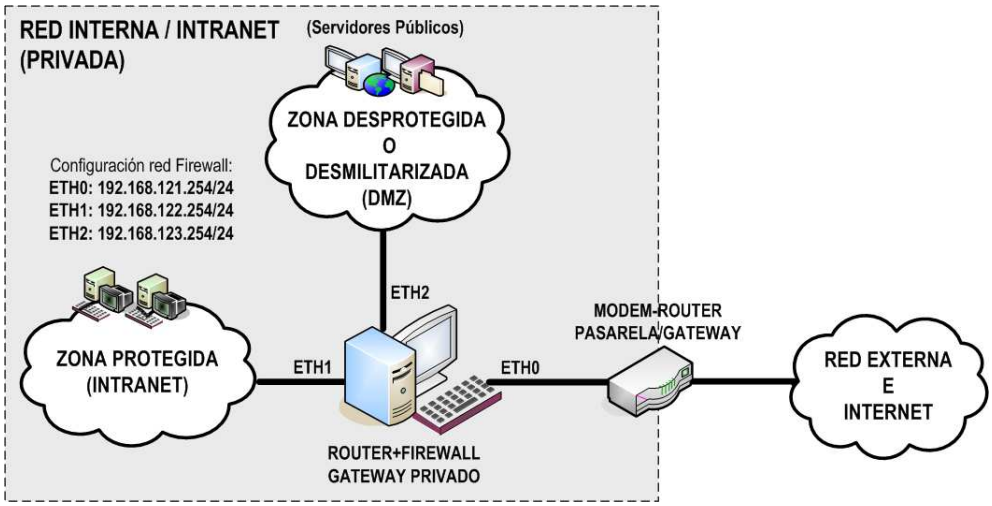
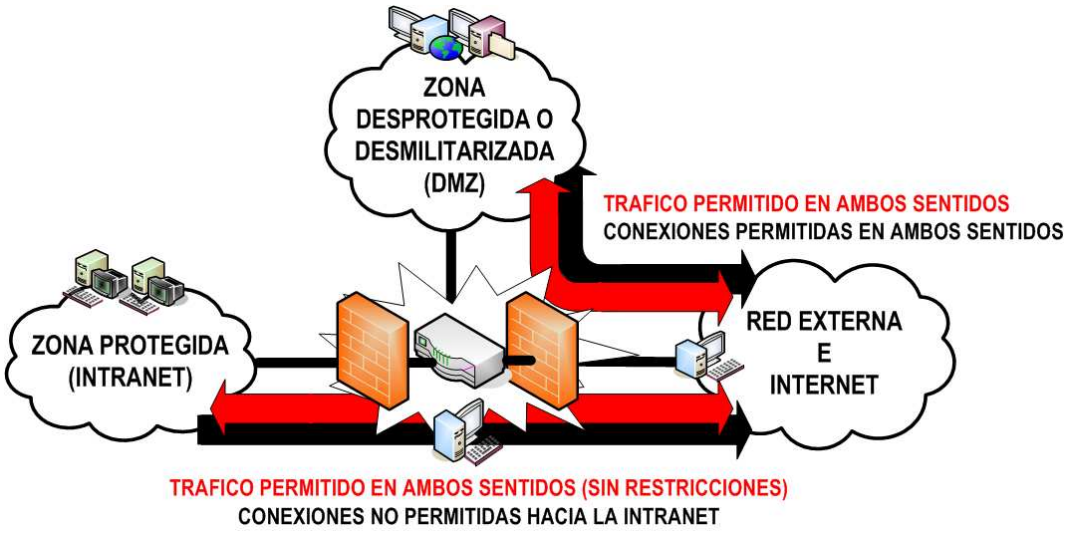


Figura 6.19: Esquema de red del ejercicio 6.6

zona desmilitarizada (`192.168.123.0/24`), pero en ningún caso aceptar peticiones de conexión a puertos asociados a los equipos de la zona protegida (`192.168.122.0/24`). Teniendo en cuenta que los servicios son ofrecidos por equipos que tienen asignadas direcciones IP privadas, el `gateway` deberá encargarse de recibir todas las peticiones de conexión y redirigirlas (PREROUTING/DNAT) al servidor correspondiente.

¡¡Recuerda!! `NAT` es el proceso a través del cual se alteran los paquetes TCP/IP que atraviesan al equipo `gateway/firewall` intercambiando las direcciones IP de origen (`POSTROUTING/SNAT`) o destino (`PREROUTING/DNAT`) con la finalidad de enmascarar y encubrir direcciones IP no reconocidas (direcciones privadas - Intranet) dentro de Internet.

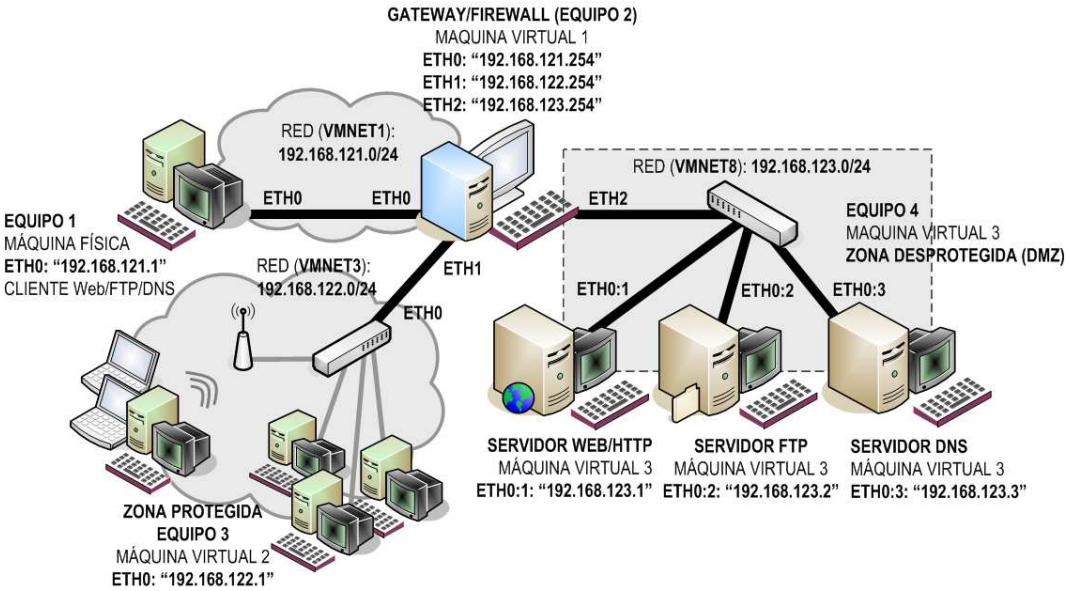


De una manera gráfica, la situación planteada se puede resumir a través de la figura anterior. Puede advertirse que son aceptados todos aquellos paquetes TCP/IP que forman parte del tráfico de datos relacionados con una conexión previamente establecida, independientemente de la interfaz por la que vayan a entrar o salir, pero restringiendo el paso de paquetes dirigidos a establecer una conexión, permitiendo únicamente aquellos que van dirigidos a la DMZ.

Para cumplir todas las especificaciones, podría implementarse y comprobarse la configuración de red propuesta, haciendo uso de VMware tal como se muestra en la última figura.

El script a ejecutar en el equipo que hace `gateway/firewall` podría ser el siguiente:

```
# Script de Configuración del GATEWAY/FIREWALL: EJERCICIO 6
#
# 1º.- Configuramos GNU/Linux como Gateway permitiendo
# el reenvió de paquetes a través de sus interfaces de red:
echo 1 > /proc/sys/net/ipv4/ip_forward
# 2º.- Reseteamos las reglas del gateway/firewall: Inicialicación.
iptables -t filter -F
iptables -t nat -F
# 3º.- Hacemos NAT postrouting
iptables -t nat -A POSTROUTING -j MASQUERADE
# 4º.- Hacemos NAT PREROUTING sobre los servicios de la DMZ al
# recibir los paquetes de conexión el gateway,
# permitiendo al mismo tiempo su reenvío (FORWARD):
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to 192.168.123.1:80
iptables -t filter -A FORWARD -i eth0 -s 0.0.0.0/0 -p tcp --dport 80 -j ACCEPT
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 21 -j DNAT --to 192.168.123.2:21
iptables -t filter -A FORWARD -i eth0 -s 0.0.0.0/0 -p tcp --dport 21 -j ACCEPT
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 21 -j DNAT --to 192.168.123.2:21
iptables -t filter -A FORWARD -i eth0 -s 0.0.0.0/0 -p udp --dport 21 -j ACCEPT
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 53 -j DNAT --to 192.168.123.3:53
iptables -t filter -A FORWARD -i eth0 -s 0.0.0.0/0 -p udp --dport 53 -j ACCEPT
# 5º.- Reenviamos sin restricciones los paquetes originados en
# la zona protegida hacia Internet:
iptables -t filter -A FORWARD -i eth1 -o eth0 -j ACCEPT
# 6º.- Permitimos la transferencia de paquetes relacionados con conexiones
# previamente ya establecidas:
iptables -t filter -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
# 7º.- Por último establecemos como política por defecto no reenviar:
iptables -t filter -P FORWARD DROP
```



Ejercicio 6.7

Dada la figura 6.20 mostrada tras el enunciado, que se corresponde con el esquema de la red informática de una supuesta empresa, resolver las siguientes cuestiones:

1. Realizar un `shellscript` que configure el `ROUTER/GATEWAY/FIREWALL(1)` de tal forma que permita el tráfico de información en ambos sentidos correspondiente a las conexiones que puedan establecerse desde la `intranet` hacia `Internet`, así como desde `Internet` hacia la `DMZ` (únicamente a los servicios públicos). Se deben evitar tanto los accesos a los servicios privados de la `DMZ` como el reenvió de peticiones de conexión dirigidas hacia equipos dentro de la zona protegida de la `intranet`. Además, con el objeto de administrar/gestionar vía Web el firewall(1) deberá dejarse accesible el puerto 80 desde la interfaz `eth1`, denegando su acceso a cualquier otro puerto. Así, el `script` deberá tener en cuenta tanto las reglas de filtrado, `-t filter`, como las de `NAT`, `-t nat`. Dichas tablas, `filter` y `nat`, deberán inicializarse al comienzo del `script`. Al principio del `script` deberá hacerse referencia igualmente a la configuración `TCP/IP` del equipo, es decir, dirección IP y tablas de enrutamiento.

2. Al igual que en el apartado anterior, realizar un nuevo `shellscript` que configure de manera adecuada el equipo GNU/Linux `ROUTER/GATEWAY/FIREWALL(2)` teniendo en cuenta las siguientes premisas:

   a) Las primeras instrucciones del `script` se encargarán tanto de configurar las interfaces de red (`eth`x), como de las tablas de enrutamiento e inicializar las `iptables` (ifconfig, route e iptables). Las direcciones IP que deberán asignarse a cada una de las interfaces de red del segundo firewall se corresponderán con las direcciones IP más altas posibles dentro de cada una de las redes de clase C a la que pertenecen.

   b) En ningún caso se reenviarán peticiones de conexión hacia servicios de la zona protegida de la red interna procedentes del exterior. Tan sólo se permitirá el reenvío de aquellos paquetes que estén asociados con conexiones ya establecidas por equipos de la zona protegida con servicios de la `DMZ` o `Internet` (`iptables -t filter -A FORWARD`).

   c) Todos los equipos de la zona protegida por el segundo firewall de la `Intranet` tienen permisos para conectarse a la `Internet`, pero evitando (excepto para el departamento de desarrollo) que puedan usar clientes `P2P` del tipo `emule/amule/lmule` (puertos 4090:4100) para evitar el consumo del ancho de banda de la línea ADSL contratada (`iptables -t filter -A FORWARD`).

   d) Internamente se permitirá el uso de clientes `P2P` del tipo `DirectConnect` (puerto 1090) con la finalidad de agilizar tanto las transferencias de datos entre los equipos, como para mensajería instantánea mediante el servicio `IRC` que estos ofrecen. El servidor `P2P` se encuentra localizado en la red interna `192.168.2.0/24` (`iptables -t filter -A FORWARD`).

   e) El secretario/director será el único equipo de la `Intranet` con permisos de acceso a la base de datos (BD) de la empresa situada en la red interna `192.168.2.0/24` (`iptables -t filter -A FORWARD`).

   f) Los equipos del departamento de desarrollo son los únicos equipos de la red interna con permisos para acceder vía `Webmin` (puerto 10000) a los servidores de la `DMZ` con la finalidad de poder configurar sus servicios (`iptables -t filter -A FORWARD`).

   g) Se dará la posibilidad de controlar remotamente vía `ssh` (puerto 22) o `webmin` (puerto 10000) el `ROUTER/GATEWAY/FIREWALL(2)` desde un equipo del departamento de desarrollo, `192.168.4.10`, con la finalidad de poder configurarlo y en su caso, solucionar

cualquier incidencia que pudiese solventarse de esta forma (`iptables -t filter -A INPUT`). Para cualquier otro equipo que no sea este (`192.168.4.10`) el acceso remoto al firewall deberá ser denegado. Además el acceso al resto de puertos del equipo GNU/Linux que desempeña las funciones de firewall(2) deberá denegarse.

h) Todos los equipos de la red interna (zona protegida), sin excepción, tienen acceso vía protocolo `SMB/NetBios` al servidor `samba` de la red `192.168.2.0/24` (`iptables -t filter -A FORWARD`).

i) Por último, haremos `NAT` sobre las direcciones de la red interna, de tal forma que en todo momento el equipo firewall(2) se haga pasar por el origen de la información que allí se genera.

3. Realizar dos últimos `shellscript` que al ejecutarlos en los servidores de la zona desprotegida (servidores FTP y HTTP), `DMZ`, garantice que tan sólo sea posible el acceso a los servicios que ofrecen. Se evitará por tanto el acceso a cualquier otro puerto (`iptables -t filter -A INPUT`). Par ello habrá que tener en cuenta expresamente lo expuesto en el apartado 2*f*), y que el firewall(2) hace `NAT POSTROUTING`.

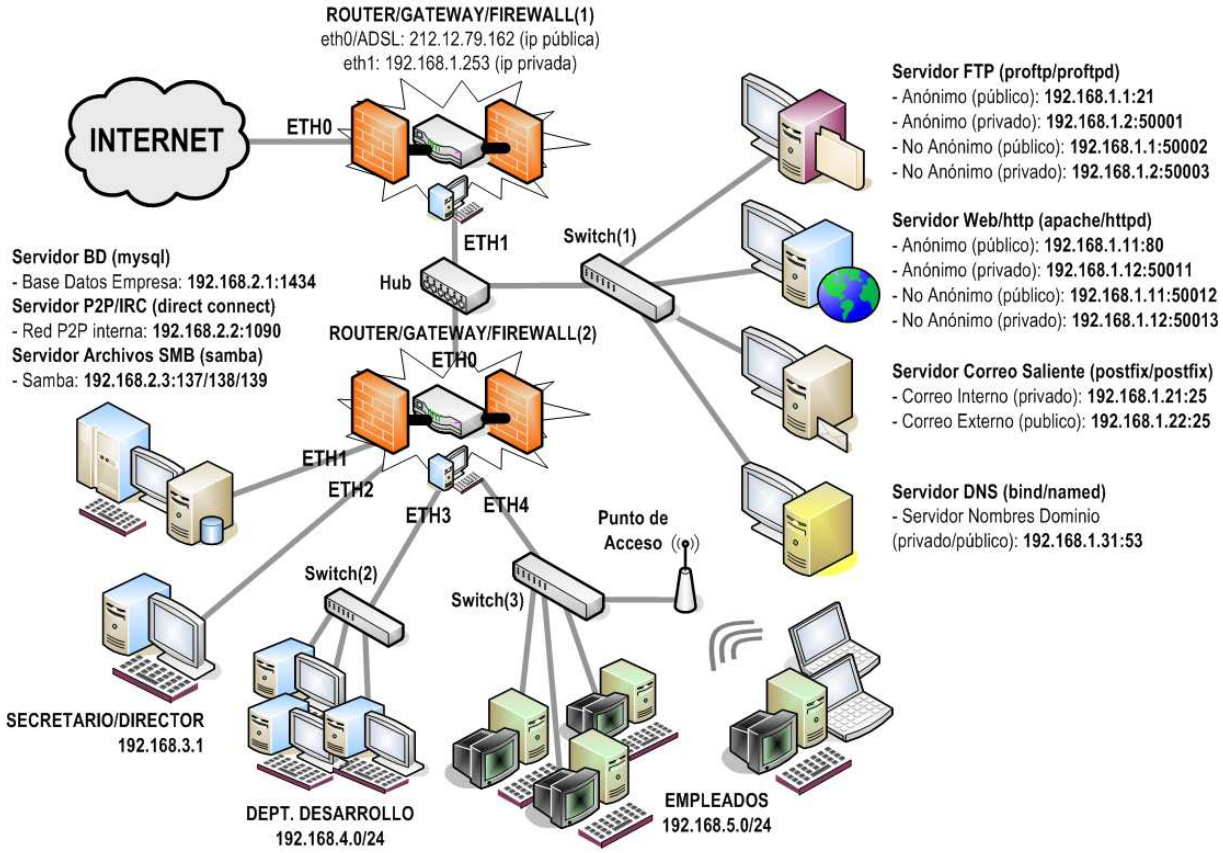


Figura 6.20: Esquema de red del ejercicio 6.7.

Solución del ejercicio 6.7

La configuración del `router/gateway/firewall(1)` se podría hacer por medio de un script similar al mostrado a continuación:

```
# Script de Configuración del ROUTER/GATEWAY/FIREWALL(1)
#
# 1º.- Configuramos las interfaces de red del equipo GNU/Linux
#                                          ROUTER/GATEWAY/FIREWALL(1)
ifconfig eth0 212.12.79.162
ifconfig eth1 192.168.1.253
# 2º.- Le indicamos a nuestro gateway quien será a su vez su
# gateway/router dentro de Internet (dirección IP del primer router
# público dentro Internet encargado de encaminar los paquetes hacia su
# destino, p.e. 212.12.79.15):
route add default gw 212.12.79.15
# 3º.- Configuramos el equipo GNU/Linux como Gateway,
# permitiendo el reenvío de paquetes:
echo 1 > /proc/sys/net/ipv4/ip_forward
# 4º.- Reseteamos las tablas de filtrado (-t filter) y
# las de NAT (-t nat)
iptables -t filter -F
iptables -t nat -F
# 5º.- Permitimos únicamente el acceso al firewall(1) por el puerto 80
# para su gestión (definimos como política por defecto denegar la
# entrada de cualquier paquete por otro puerto e interfaz):
iptables -t filter -A INPUT -i eth1 -p tcp --dport 80 -j ACCEPT
iptables -t filter -P INPUT DROP
# 6º.- Hacemos DNAT sobre las solicitudes recibidas por el
# gateway/firewall(1) procedentes de Internet dirigidas a los
# servicios ofrecidos por la DMZ (servidores públicos):
# Reenviamos los paquetes recibidos por el puerto 21 al servidor FTP
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 21 -j DNAT
                                          --to 192.168.1.1:21
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 50002 -j DNAT
                                           --to 192.168.1.1:50002
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 21 -j DNAT
                                          --to 192.168.1.1:21
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 50002 -j DNAT
                                           --to 192.168.1.1:50002
iptables -t filter -A FORWARD -i eth0 -o eth1 -d 192.168.1.1 -p tcp
                                     -m multiport --dport 21,50002 -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -o eth1 -d 192.168.1.1 -p udp
                                     -m multiport --dport 21,50002 -j ACCEPT
# Reenviamos los paquetes recibidos por el puerto 80 al servidor Web
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT
                                          --to 192.168.1.11:80
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 50012 -j DNAT
                                           --to 192.168.1.11:50012
iptables -t filter -A FORWARD -i eth0 -o eth1 -d 192.168.1.11 -p tcp
                                      -m multiport --dport 80,50012 -j ACCEPT
# Reenviamos los paquetes recibidos por el puerto 25 al servidor SMTP
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 25 -j DNAT
                                          --to 192.168.1.22:25
iptables -t filter -A FORWARD -i eth0 -o eth1 -d 192.168.1.22 -p tcp
                                           --dport 25 -j ACCEPT
```

Continúa en página siguiente

```
# Reenviamos los paquetes recibidos por el puerto 53 al servidor DNS
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 53 -j DNAT
                                              --to 192.168.1.31:53
iptables -t filter -A FORWARD -i eth0 -o eth1 -d 192.168.1.31 -p udp
                                               --dport 53 -j ACCEPT
# 7º.- Hacemos SNAT sobre las solicitudes de conexión a
# Internet procedentes de la Intranet
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# La línea anterior es igual a: ''-j SNAT --to 212.12.79.162''
# Para evitar confusiones en las comunicaciones sería
# más correcto enmascarar todo:
## iptables -t nat -A POSTROUTING -j MASQUERADE
# 8º.- Permitimos el reenvío de peticiones de conexión hacia
# servicios de Internet
iptables -t filter -A FORWARD -i eth1 -o eth0 -j ACCEPT
# 9º.- Por último, permitimos el trafico de paquetes pertenecientes
# a conexiones ya establecidas
iptables -t filter -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
# 10º.- Establecemos como política por defecto no reenviar nada que no
# haya sido indicado explícitamente mediante reglas de filtrado de FORWARD:
iptables -t filter -P FORWARD DROP
```

En relación a la configuración del segundo gateway/firewall(2), el script que realiza esta tarea sería:

```
# Script de Configuración del ROUTER/GATEWAY/FIREWALL(2)
#
# Apartado 2a):
# 1º.- Configuramos las interfaces de red del
#                        equipo GNU/Linux ROUTER/GATEWAY/FIREWALL(2):
ifconfig eth0 192.168.1.254
ifconfig eth1 192.168.2.254
ifconfig eth2 192.168.3.254
ifconfig eth3 192.168.4.254
ifconfig eth4 192.168.5.254
# 2º.- Le indicamos quien será la interfaz que hará de gateway
# hacia la Internet, es decir, la eth1 del GATEWAY(1):
route add default gw 192.168.1.253
# 3º.- Permitimos el reenvío de paquetes entre las distintas
#                        interfaces de red del equipo:
echo 1 > /proc/sys/net/ipv4/ip_forward
# 4º.- Reseteamos las tablas de filtrado (-tfilter)
# y las de NAT (-tnat)
iptables -t filter -F
iptables -t nat -F
```

```
# Apartado 2b):
# 5º.- Definimos como política por defecto denegar la entrada
# de cualquier paquete que desee entrar al firewall(2), al igual
# que con todos aquellos paquetes que necesiten reenviarse.
# Todo lo que quiera dejarse entrar o reenviar deberá indicarse
# explícitamente mediante reglas INPUT y FORWARD:
iptables -t filter -P INPUT DROP
iptables -t filter -P FORWARD DROP
## Escoger como política por defecto para FORWARD DROP garantiza
```

Continúa en página siguiente

```
# que no puedan acceder desde Internet a la zona protegida. De
# una manera más explicita hubiera sido:
## iptables -t filter -A FORWARD -i eth0 -j DROP
# 6º.- Permitimos el tráfico de paquetes asociados a conexiones
# previamente establecidas con Internet:
iptables -t filter -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
# Apartado 2c):
# 7º.- Permitimos la conexión hacia servicios de la Internet
# exceptuando emule/amule (supondremos que los puertos relacionados
# con emule se encuentran en el rango 4090:4100):
iptables -t filter -A FORWARD -o eth0 -p tcp --dport 4090:4100 -j DROP
iptables -t filter -A FORWARD -o eth0 -j ACCEPT
```

```
# Apartado 2d):
# 8º.- Permitimos conexiones asociadas a la red P2P (direct connect)
# dentro de la zona protegida de la Intranet (si quisiéramos
# evitar conexiones a servidores direct connect de Internet deberemos
# incluir las dos siguientes instrucciones):
## iptables -t filter -A FORWARD -p tcp --dport 1090 -i eth0 -j DROP
## iptables -t filter -A FORWARD -p tcp --dport 1090 -o eth0 -j DROP
iptables -t filter -A FORWARD -p tcp --dport 1090 -j ACCEPT
```

```
# Apartado 2e):
# 9º.- Restringimos el acceso a la Base de Datos de la empresa
# al director y secretario
iptables -t filter -A FORWARD -p tcp --dport 1434 -i eth2 -o eth1 -j ACCEPT
## Como la política por defecto es FORWARD DROP el resto
# de equipos no accederán a la BD.
```

```
# Apartado 2f):
# 10º.- Permitimos el acceso a la DMZ vía webmin
# desde el departamento de desarrollo
iptables -t filter -I FORWARD 1 -s 192.168.4.0/24 -p tcp
                                    --dport 10000 -o eth0 -j ACCEPT
iptables -t filter -I FORWARD 2 -p tcp --dport 10000 -o eth0 -j DROP
## Con la finalidad de contrarrestar la regla anterior del
# apartado 2c): ‘‘iptables -A FORWARD -o eth0 -j ACCEPT’’, en
# lugar de añadir (opción -A, append) las reglas al final de la lista de
# FORWARD las insertamos (opción -I, insert) en las primeras
# posiciones de la lista de tal forma que al ser leídas secuencial-
# mente sean tenidas en cuenta antes.
```

```
# Apartado 2g):
# 11º.- Permitimos el control remoto del firewall(2) desde el
```

Continúa en página siguiente

```
# departamento de desarrollo
iptables -t filter -A INPUT -s 192.168.4.10 -p tcp --dport 22 -j ACCEPT
iptables -t filter -A INPUT -s 192.168.4.10 -p tcp --dport 10000 -j ACCEPT
# Como la política por defecto es INPUT DROP el resto de equipos
# no podrán entrar al firewall(2) para gestionarlo/administrarlo.
```

```
# Apartado 2h):
# 12º.- Permitimos conexiones SMB/Netbios dentro de la Zona
# protegida de la Intranet
iptables -t filter -A FORWARD -o eth1 -d 192.168.2.3 -p tcp
                                              --dport 137:139 -j ACCEPT
```

```
# Apartado 2i):
# 13º.- Hacemos SNAT para los paquetes generados en la zona
# protegida de la Intranet
iptables -t nat -A POSTROUTING -j MASQUERADE
```

Por último, se detallarán los scripts que habría que ejecutar en los servidores de la DMZ (ftp y http) para filtrar sus conexiones de entrada:

```
# Script de Configuración del Servidor FTP de la DMZ
#
# 1º.- Configuramos las interfaces de red del equipo GNU/Linux
ifconfig eth0 192.168.1.1
ifconfig eth0:1 192.168.1.2
# 2º.- Le indicamos quien será su gateway hacia la Internet,
# en este caso la eth1 del GATEWAY(1)
route add default gw 192.168.1.253
# 3º.- Reseteamos las tablas de filtrado (-t filter)
iptables -t filter -F
# 4º.- Definimos como política por defecto denegar la
# entrada de cualquier paquete
iptables -t filter -P INPUT DROP
# 5º.- Permitimos únicamente la conexión por los
# puertos 21, 50001, 50002, 50003 (TCP/UDP
iptables -t filter -A INPUT -d 192.168.1.1 -p tcp -m multiport
                                              --dport 21,50002 -j ACCEPT
iptables -t filter -A INPUT -d 192.168.1.2 -p tcp -m multiport
                                              --dport 50001,50003 -j ACCEPT
iptables -t filter -A INPUT -d 192.168.1.1 -p udp -m multiport
                                              --dport 21,50002 -j ACCEPT
iptables -t filter -A INPUT -d 192.168.1.2 -p udp -m multiport
                                              --dport 50001,50003 -j ACCEPT
# 6º.- Permitimos el control remoto vía webmin desde el
# departamento de desarrollo. Al haber hecho SNAT el firewall(2),
# el permiso se le debe conceder a él:
iptables -t filter -A INPUT -s 192.168.1.254 -p tcp --dport 10000 -j ACCEPT
```

```
# Script de Configuración del ServidorHTTP de la DMZ
#
# 1º.- Configuramos las interfaces de red del equipo GNU/Linux
ifconfig eth0 192.168.1.11
ifconfig eth0:1 192.168.1.12
# 2º.- Le indicamos quien será su gateway hacia la
# Internet, la eth1 del GATEWAY(1)
route add default gw 192.168.1.253
```

Continúa en página siguiente

```
# 3º.- Reseteamos las tablas de filtrado (-t filter)
iptables -t filter -F
# 4º.- Definimos como política por defecto denegar la entrada
# de cualquier paquete
iptables -t filter -P INPUT DROP
# 5º.- Permitimos únicamente la conexión por los puertos 80,
# 50011, 50012, 50013
iptables -t filter -A INPUT -d 192.168.1.11 -p tcp -m multiport
                                            --dport 80,50012 -j ACCEPT
iptables -t filter -A INPUT -d 192.168.1.12 -p tcp -m multiport
                                           --dport 50011,50013 -j ACCEPT
# 6º.- Permitimos el control remoto vía webmin desde el
# departamento de desarrollo. Al haber hecho SNAT el firewall(2),
# el permiso se le debe conceder a él:
iptables -t filter -A INPUT -s 192.168.1.254 -p tcp --dport 10000 -j ACCEPT
```

Capítulo 7

UN SERVIDOR DE HORAS POR MEDIO DE GNU/Linux

7.1. Introducción al concepto de tiempo en un ordenador

Los ordenadores contienen, implementado en placa madre, un sistema hardware que les permite conocer la fecha y hora actuales. Es un chip denominado reloj CMOS o reloj hardware que mide el paso del tiempo. Sencillamente es un contador que se va incrementando al pasar una determinada cantidad de tiempo.

La BIOS de nuestro equipo se comunica con este circuito integrado para determinar la hora con la que trabajará la máquina. A su vez los sistemas operativos solicitan a la BIOS la información de fecha y hora. Esta hora y fecha es la indica en los entornos de escritorio con los que normalmente trabajamos. Esto significa que una máquina tiene una sola hora ya que sólo tiene una BIOS[1]. Esto significa que si un mismo ordenador alberga diferentes sistemas operativos el cambio de hora realizado con alguno de ellos afecta a lo que mostrarán el resto. Esto se debe a que el cambio de hora, realizado a través de un sistema operativo, modifica la hora de la BIOS.

En un sistema GNU/Linux el cambio de hora se realiza por medio del comando `date`. Es un comando reservado al usuario `root`, lo que da idea de la importancia que tiene. Por ejemplo el comando:

```
[root@linux]# date 11231426
```

Indicaría al sistema que debe cambiar la fecha del sistema al mes de noviembre (11), día 23 con hora las 14 horas y 26 minutos. Observa que si tu ordenador tiene la posibilidad de arrancar otro sistema operativo, por ejemplo Windows, cuando lo hagas el sistema te indicará la fecha y hora según fue configurada por medio de `date`.

Un último aspecto importante de la configuración horaria de forma manual es que los relojes hardware implementados en la placa madre tienen poca precisión, que además depende del estado de la batería de reloj. Esto provoca que puedan producirse desfases de hasta 20 segundos por día.

7.2. Algo sobre la terminología utilizada en la medida del tiempo

Cuando deseamos configurar en nuestra máquina algún aspecto de los que hacen referencia a la medida del tiempo nos encontramos con acrónimos tales como GMT, UTC, NTP, ... El significado de algunos de estos acrónimos es la siguiente.

GMT
: Greenwich Mean Time. Se corresponde con la hora asociada al meridiano de Greenwich. Su valor tiene una estrecha relación con la posición de determinados cuerpos celestes.

[1] Es necesario indicar que VMware crea una BIOS propia para sus máquinas virtuales.

UTC
: Universal Time Coordenated. En un sucesor de GMT. Es un estándar internacional que pretende corregir dos aspectos de su antecesor. El primero es que basa la hora en función de los relojes atómicos en lugar de los cuerpos celestes, y el segundo es que pretende eliminar la asociación del estándar a un determinado lugar geográfico.

CET
: Central European Time. Hora central europea en relación con la UTC. En este caso CET=UTC+1.

  Existen términos para indicar modificaciones similares a esta, tales como CEST, WET, WEST, ..., pero que a efectos de este capítulo carecen de interés.

Reloj Atómico
: Reloj basado en la frecuencia de una vibración atómica. Para hacernos una idea de la precisión conseguida con estos relojes diremos que el estandar actual permite un error de 1 segundo en un transcurso de 300000 años. Esta exigencia no parece difícil de conseguir con estos relojes, de hecho los relojes actuales se desfasan un segundo cada 52 millones de años.

NTP
: Network Time Protocol. Es un protocolo que permite distribuir el tiempo UTC por Internet.

7.3. El Network Time Protocol (NTP)

NTP es quizás uno de los protocolos más antiguos y usados en Internet. La primera versión de NTP fue lanzada en 1985 y recogida en el RFC958. Se produjeron continuas actualizaciones del protocolo y en 1992 apareció RFC1305 definiendo la versión 3 del mismo, que todavía sige vigente en la actualidad esperando que aparezca el RFC que defina la versión[2] 4 (que es la actual). Actualmente sólo está disponible el RFC4330 (aparecido en enero de 2006) que define el SNTPv4, que a su vez es un subconjunto de NTPv4.

En el mundo actual es necesaria cierta sincronización horaria para poder realizar muchas tareas ligadas a la informática. Por ejemplo el análisis de `logs` de diferentes máquinas exige sincronización horaria para que no se produzcan inconsistencias. Aplicaciones, como por ejemplo BIND[3], necesitan que no exista un desfase elevado entre los relojes de las máquinas que permiten su control. La compartición de bases de datos que procesan transacciones de comercio electrónico o banking podría ser otro ejemplo.

Como ya se dijo antes, NTP permite distribuir la UTC por Internet. Con esto podemos asegurar una adecuada sincronización entre todas nuestras máquinas y el horario oficial. La precisión conseguida es de milisegundos, más que suficiente para las tareas de administración que debemos realizar. De hecho cabría hacerse la pregunta ¿es necesario un sistema de hora tan preciso? Posiblemente no, pero la gran ventaja es que el administrador ya no tendrá que preocuparse de cambiar la hora en los equipos que gestiona.

Así pues, NTP nos va a permitir erigir una máquina de nuestra red como servidor de horas (dará servicio al resto de nuestras máquinas) y ella a su vez será servida (es decir, será cliente) desde otra máquina en internet.

7.3.1. Estructura de los servidores en Internet

Los dispositivos que determinan la hora exacta son los relojes atómicos y satélites GPS. Estos se encuentran a la cabeza de la estructura jerárquica del servicio dado mediante NTP. Dentro de esta estructura, dichos dispositivos se denominan *stratum 0* y son los que dan servicio a los ordenadores catalogados como *stratum 1*, que es el máximo nivel jerárquico en Internet. A su vez los *stratum 1* dan servicio a los *stratum 2* y así sucesivamente hasta el nivel más bajo catalogado como *stratum*

[2]La versión actual de NTP es la 4, pero el RFC que la describe todavía no está disponible.
[3]BIND se describe en el capítulo 2.

16. Si un equipo es *stratum 16* significa que está desincronizado, es decir que no está conectado a ningún servidor.

En la página `www.ntp.org` es posible encontrar un listado de los *stratum 1* y *2*. Muchos de los servidores de este listado son de libre acceso[4], aunque algunos de ellos lo tienen restringido. Lo normal es que nuestra red local se sincronice por medio de su servidor a través de un *stratum 3* con el fin de no sobrecargar niveles superiores.

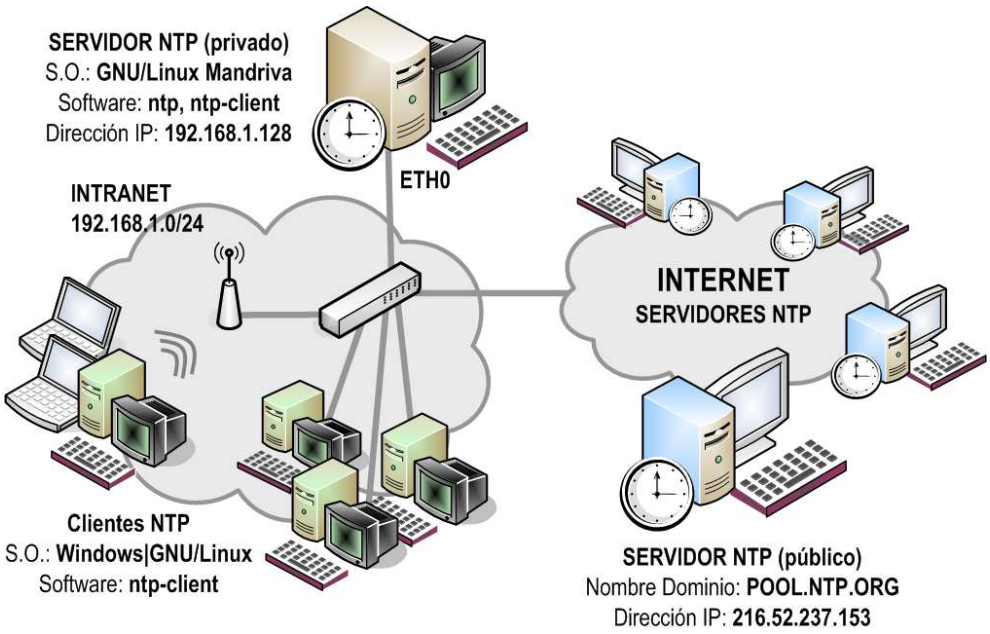


Figura 7.1: Posible esquema de un servicio NTP en una LAN.

Posteriormente veremos a que servidores podemos conectarnos, aunque es importante señalar que dentro del proyecto NTP se está promovieno los NTP Pool Time Servers (Fondo común de servidores de tiempo). Los NTP Pool Time Servers pertenecen a un conjunto de muchos servidores de hora recogidos bajo una misma `url` a los que se les hace la consulta de forma aleatoria. Es decir, en nuestra configuración podemos decir que nuestro servidor de horas es `pool.ntp.org`, esto signifcará que cuando nuestro servidor se actualice pedirá información a alguno de los servidores `pool` de forma aleatoria, lo que evita sobrecargas, además de ser un proceso transparente para el administrador que configura NTP en su LAN.

7.4. Contenidos de la práctica con NTP

En la configuración práctica que se muestra a continuación se explicarán los procedimientos que permiten:

1. Instalar el software necesario para configurar nuestro equipo con un servidor de tiempo.
2. Determinar el contenido del fichero de configuración `/etc/ntp.conf`.
3. Configurar y comprobar el correcto funcionamiento del servidor.
4. Conocer los comandos que permiten controlar el servicio dado por NTP.
5. Afianzar los conocimientos adquiridos por medio de ejercicios prácticos.

[4]En el momento de escribir estas líneas en España sólo existe un servidor *stratum 1* cuyo nombre es `hora.roa.es`, y es de libre acceso.

PRÁCTICA: CONFIGURACIÓN DE UN SERVIDOR DE HORAS

7.5. Instalación del servidor NTP

Para que nuestro equipo haga de servidor de horas es necesario instalar el software `ntp-4.x`. El cuatro indica que el desarrollo de la aplicación está en su cuarta versión.

Por ejemplo, en el caso de Mandriva[5] escribiríamos:

```
[root@linux]# urpmi ntp
```

Observa que la máquina te hace el siguiente aviso:

```
Para satisfacer las dependencias, se instalarán los 2 paquetes siguientes (3 MB):
ntp-4.2.0-18.1.102mdk.i586
ntp-client-4.2.0-18.1.102mdk.i586
¿Está todo bien? (S/n)
```

Si deseas configurar tu máquina como servidor de horas necesitas los dos paquetes, el primero es el que hace realmente de servidor, pero como tu máquina a su vez es cliente de alguno de los servidores de internet (posiblemente *stratum 3*) necesita también el segundo de los paquetes. Observa que sin embargo los ordenadores de la red que tu administras, podrían funcionar únicamente con el paquete `ntp-client-4.`x ya que únicamente tendrán un comportamiento como cliente hacia tu servidor (incluso para un comportamiento como cliente, se recomienda instalar el paquete `ntp-4.`x).

Los requerimientos de hardware son muy bajos y por tanto no deberías tener ningún problema con el correcto funcionamiento de tu servidor.

7.5.1. Funcionamiento del demonio NTPD

Tras instalar NTP en tu máquina dispondrás de una serie de programas que te permitirán su manejo. El más importante de estos programas es el demonio de NTP, llamado `ntpd`. Cuando el ordenador es arrancado el sistema operativo carga la hora proporcionada por el reloj CMOS (el chip de reloj situado en la placa madre). Tras el arranque el demonio `ntpd` trata de sincronizarse con un servidor de hora[6]. Este servidor de hora y `ntpd` irán corrigiendo poco poco la desviación existente en nuestro reloj.

7.5.2. Tipos de clientes y servidores NTP

Los clientes y servidores NTP se pueden relacionar entre sí para trabajar de diferentes formas. A continuación trataremos de dar un pequeño resumen de las diferentes formas de trabajo posibles:

Servidor

Un equipo servidor NTP es el encargado de servir la hora a los equipos clientes. Los clientes hacen una solicitud de actualización de hora al servidor y este les responde enviando información adicional como la precisión con la que trabaja y el *stratum* al que pertenece.

[5] Mandriva está basada en los paquetes `rpm` desarrollados por RedHat. Adapta el comando mostrado arriba a tu distribución. Por ejemplo, en una Debian escribirías:

```
[root@linux]# apt-get install ntp
```

[6] El servidor de hora buscado se encuentra definido en el fichero `/etc/ntp.conf` que posteriormente se comentará de forma detallada.

Cliente
Un equipo cliente es el que consigue las actualizaciones horarias de un servidor (uno o varios), usándolas para calibrar su reloj. El cliente determina la desviación que posee su reloj con respecto a la del servidor y va ajustándola poco a poco intentando conseguir una desviación cero. El máximo error entre el cliente y el servidor viene dado por el tiempo de transmisión de la información por la red.

Peer
Un *peer* es uno de los miembros de un grupo de servidores NTP que están acoplados entre sí. En un grupo de estas características los equipos se sincronizan entre sí consiguiendo aumentar la precisión.

Servidores broadcast/multicast
Los servidores NTP pueden trabajar en modo *broadcast* o *multicast*. Ambos modos funcionan de manera similar. En el modo *broadcast* (difusión) el servidor envía periódicamente las actualizaciones horarias a todos los equipos de su red. En el modo *multicast* se decide a que equipos de la red se mandan las actualizaciones. Usando estos modos se puede reducir enormemente el tráfico de red NTP, sobre todo cuando en ella hay muchos clientes.

Clientes broadcast/multicast
Un cliente *broadcast* o *multicast* escucha las actualizaciones NTP a través de una dirección *broadcast* o *multicast*. Cuando el cliente recibe el primer paquete de actualización trata de cuantificar el retraso, debido a los tiempos de propagación de los datos en la red, con el servidor para mejorar la precisión en posteriores *broadcasts*. Esto se lleva a cabo por medio de una serie de breves intercambios en los que el cliente y el servidor tienen una comunicación entre ellos (no es un proceso de difusión). Transcurridos estos intercambios el cliente tiene una idea del retraso de comunicación en la red y por lo tanto puede estimar la actualización horaria únicamente utilizando paquetes *broadcast*.

7.6. Configuración del servicio NTP

La configuración del servicio queda reflejada en el fichero `/etc/ntp.conf`. Este fichero es leído en el arranque por el demonio `ntpd` con el fin de encontrar las fuentes de sincronización, modos de operación e información relacionada con el servicio.

El formato del archivo es similar a la mayoría de los ficheros de configuración de los sistemas UNIX. Los comentarios son precedidos por # y las líneas en blanco son ignoradas. Los comandos de configuración están constituidos por una palabra clave seguida de parámetros. Algunos de estos comandos de configuración se describen a continuación.

Cláusulas de configuración NTP

- **server**

Sirve para indicar el servidor desde el que se realizarán las actualizaciones. Existen muchos servidores que pueden ser consultados en `http://ntp.isc.org/bin/view/Servers/WebHome`. Aquí recomendamos el uso de los servidores del fondo común[7]. Así, un posible ejemplo de configuración de servidores podría ser:

```
server  pool.ntp.org
server  1.europe.pool.ntp.org
server  2.europe.pool.ntp.org
```

La primera línea tomará un servidor de horas del conjunto global de los servidores recogidos en `pool.ntp.org` y las dos siguientes líneas especifican otros servidores, pertenecientes al fondo común, que además se encuentran en Europa. Es posible especificar incluso servidores de nuestro país introduciendo una línea como la siguiente,

```
server  1.es.pool.ntp.org
```

- **peer**

Esta palabra clave es utilizada en el archivo `/etc/ntp.conf` de nuestro servidor para indicar que trabajará de forma sincronizada con otro servidor y así aumentar la precisión. Normalmente es utilizado entre *stratum* de orden bajo (el orden más bajo es el *16*) para mejorar la precisión, pero también puede ser utilizado para proporcionar redundancia a los *stratum* de orden elevado. *Ejemplo:*

```
peer hora.aula17.cossio.com
peer hora.aula16.cossio.com
```

- **broadcast**

Esta palabra clave permite configurar al servidor de horas en modo *broadcast* o *multicast*. Por ejemplo un funcionamiento del servidor en modo *broadcast* para la red 192.168.19.0 debería ser configurado añadiendo la siguiente línea al archivo `/etc/ntp.conf`,

```
broadcast 192.168.19.255
```

Por su parte la dirección oficial de multicasting con NTP es la 224.0.1.1, por lo tanto para hacer un difusión *multicast* escribiríamos:

```
broadcast 224.0.1.1 ttl 6
```

El `ttl` (Time To Live) es opcional. El número que le acompaña, en nuestro ejemplo el 6, indica el número máximo de saltos a través de diferentes redes que puede dar el paquete multicast.

- **driftfile**

Este parámetro indica el fichero en el que se van a guardar las desviaciones existentes entre nosotros y el servidor de horas que nos está dando la información para el ajuste. El demonio `ntpd` utiliza estas desviaciones para compensar automáticamente las desviaciones que de forma natural sufre el reloj de nuestra máquina. Esto permitirá tener una cierta precisión incluso en el caso de que se pierda la comunicación con los servidores. Debe ubicarse en un lugar donde el usuario `ntp` pueda escribir; `ntp` no escribe directamente sobre este fichero sino que crea uno en el mismo directorio y posteriormente va trasladando los datos a este. Esto significa que tanto el directorio donde se encuentra el archivo, como el propio archivo deben pertenecer a `ntpd`. Ejemplo:

```
driftfile  /var/lib/ntp/drift
```

El proceso de creación del mismo sería:

```
[root@linux]# mkdir /var/lib/ntp/
[root@linux]# touch /var/lib/ntp/drift
[root@linux]# chown -R ntp.ntp /var/lib/ntp
```

- **logfile**

Sirve para indicar el fichero donde se guardará la información que se audita con respecto al servicio del NTP. Al igual que ocurría con el `driftfile` el fichero debe ser accesible por el usuario del sistema `ntp`.

- **restrict**

Por defecto, una vez que tenemos el servidor de horas configurado, puede ser utilizado por otros equipos desde Internet. Si se desea restringir el acceso se puede escribir lo siguiente,

```
restrict default ignore
```

Con esta línea se impide el acceso a cualquier máquina. En este caso el servidor de horas sólo serviría para tener actualizada la hora en la propia máquina. No podría dar servicio a nadie. Desde el punto de vista computacional existen otras soluciones más económicas que la de tener corriendo continuamente el demonio `ntpd`, tal y como se describirá posteriormente.

En el caso de querer permitir el acceso únicamente a las máquinas de nuestra red[8] escribiríamos,

```
restrict 192.168.19.0 mask 255.255.255.0
```

Una opción más adecuada sería,

```
restrict 192.168.19.0 mask 255.255.255.0 notrust nomodify notrap
```

La opción `notrust` indica que nuestro servidor no utilizará nunca a los host de la red como fuentes de sincronización.

La opción `nomodify` rehusa la recepción de paquetes por parte de los equipos de la red de paquetes de configuración que pudieran modificar el estado del servidor.

La opción `notrap` descarta el uso de mensajería de control con los equipos objetivo (los de la red).

Existen muchas más opciones de restricción que pueden ser consultadas en `ntp.conf(5)`.

- **broadcastclient**

El servidor de horas de una red de área local generalmente utiliza procedimientos de difusión para publicar su hora con el fin de que los equipos clientes la actualizen. Este permite la actualización en grandes redes reduciendo considerablemente el tráfico NTP. Para que un cliente trabaje de esta forma basta con colocar en su archivo de configuración `/etc/ntp.conf` la línea,

```
broadcastclient
```

Esto es una buena solución porque cuando un cliente trabaja en este modo no necesita tener especificado un servidor de horas (cláusula `server`) y por tanto los servidores de hora de la red pueden ser cambiados sin causar problemas administrativos en los clientes.

Este tipo de cliente no debe ser usado si las distancias de red son elevadas.

- **multicastclient**

Tiene el mismo funcionamiento que `broadcastclient`, pero el cliente estará ahora configurado para recibir únicamente los paquetes multicasting lanzados por el servidor.

- **broadcastdelay**

Cuando el cliente está funcionando en modo broadcasting o multicasting con los primeros paquetes recibidos se calcula el tiempo de retraso en la transmisión de los datos por la red para corregir errores de sincronización.

Si se desea se puede establecer un tiempo de retraso por medio del comando `broadcastdelay` que será utilizado en el caso de que esas medidas iniciales no puedan realizarse. Por defecto vale 4ms.
Ejemplo:

```
broadcastdelay 0.008
```

Establece ese tiempo en 8ms.

Un ejemplo de fichero de configuración podría ser:

```
server     pool.ntp.org
server     1.europe.pool.ntp.org
server     2.europe.pool.ntp.org
driftfile  /var/lib/ntp/drift
logfile    /var/log/ntpd.log
```

Una vez que tienes el fichero de configuración basta con rearrancar el servicio[9],

```
[root@linux]# service ntpd restart
```

[9]Como ya ha sido comentado varias veces en el resto de capítulos, para iniciar/reiniciar el servicio, puedes hacerlo por medio de:

```
[root@linux]# /etc/init.d/ntpd restart
```

esto es algo que funcionará en todas las distribuciones.

7.6.1. Comprobación del correcto funcionamiento del NTP

Una forma de comprobar que el servicio está corriendo es utilizar el comando `ntptrace`. Nos indicará con quien se ha efectuado la conexión,

```
[root@linux]# ntptrace
```

proporcionando una respuesta similar a esta,

```
localhost: stratum 2, offset 0.000000, root distance 0.121820
Time2.Stupi.SE: stratum 1, offset 0.000000, root distance 0.000000, refid 'PPS'
```

Esto está indicando una conexión con un servidor *stratum 1* y que por tanto nuestro servidor es un *stratum 2*. Los campos devueltos son: nombre de la máquina, tipo de stratum, desviación horaria en segundos, entre el servidor y el equipo que lo sirve a él, distancia de sincronización en segundos, y únicamente para los *stratum 1* el identificador del reloj de referencia (en nuestro ejemplo PPS). Observa que el offset (desviación horaria en segundos) es cero. En realidad nunca es cero, pero cuando el servicio arranca este parámetro se pone a cero hasta que se produce la sincronización debido a que la diferencia entre máquinas puede ser superior a 1000 segundos. En condiciones de funcionamiento normales, si el offset es superior a 1000 segundos `ntpd` manda un mensaje al fichero de `log` y deja de funcionar porque presupone que se ha producido un fallo grave (por ejemplo de hardware) y que por tanto no es posible la sincronización. Si al cabo de un tiempo volvemos a ejecutar el `ntptrace` obtendremos:

```
localhost: stratum 2, offset 0.001193, root distance 0.129666
Time2.Stupi.SE: stratum 1, offset 0.000000, root distance 0.000000, refid 'PPS'
```

Otra forma de conocer si el equipo esta recibiendo datos de sincronización es preguntando al servicio NTP por medio de `ntpq`:

```
[root@linux]# ntpq -pn
```

cuya respuesta sería inicialmente,

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
192.36.143.151  .PPS.            1 u    8   64    1  124.141 10907.2   0.002
80.64.135.105   130.149.17.21    2 u    7   64    1  228.859 10847.2   0.002
80.85.129.25    130.235.20.3     3 u    6   64    1  134.506 10888.6   0.002
```

Esto indicaría que no hay transferencia de datos de sincronización. Cuando esta transferencia existe alguna de las líneas debe estar precedida por `*`. Si transcurridos unos minutos volvemos a realizar la pregunta se obtiene como respuesta,

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*192.36.143.151  .PPS.            1 u  245  512   37  121.820    2.092   4.104
+80.64.135.105   130.149.17.21    2 u  314  512   37  230.160  -55.581   4.263
+80.85.129.25    130.235.20.3     3 u  245  512   37  106.041   -3.558    3.39
```

7.7. Resumen de los comandos más importantes

Al instalar el paquete `ntp`, además del demonio `ntpd` y del archivo de configuración `/etc/ntp.conf` se instalan otras aplicaciones, algunas de ellas ya utilizadas en la sección anterior. A continuación se muestra una breve descripción de las más importantes.

Comandos asociados al servicio NTP

- **ntpd**

Es un demonio[10] que mantiene la hora y la fecha del sistema basándose en los servidores indicados en el fichero `/etc/ntp.conf`. Una característica importante de `ntpd` es que, hasta que se produce la sincronización, va cambiando la hora de forma gradual. Esto tiene como inconveniente el hecho de que el proceso de sincronización puede requerir una cantidad de tiempo elevada y la ventaja de que el usuario no percibe ningún cambio apreciable durante ese proceso. Nota: La comunicación con los servidores que permiten la sincronización/actualización se realiza cada cierta cantidad de tiempo. Esta cantidad de tiempo se encuentra entre 64 y 1024 segundos; y en cada una de estas conexiones se realiza un cambio gradual estipulado por defecto en 0.5ms por cada segundo transcurrido entre dos comunicaciones de actualización (en el caso de que la actualización se hiciese cada 64 segundos se produciría una variación en el reloj de 128ms). Esto significa que un cambio de un segundo en el reloj requiere un transcurso de 2000 segundos.

- **ntp-wait**

Este comando es utilizado durante el proceso de arranque para detener la secuencia de inicio hasta que el demonio `ntpd` corrige y marca la hora del sistema.

- **ntpq**

Este comando permite preguntar al servidor del NTP el estado de las operaciones que realiza. Tiene pocas opciones, alguna de las cuales ya ha sido utiliza en la sección anterior. La principal es `-p`, que muestra en pantalla las conexiones existentes,

```
[root@linux]# ntpq -p
```

```
     remote           refid      st t when poll reach   delay   ...
==============================================================================
*ipx10914.ipxser 192.53.103.103  2 u   49   64   377  113.539  ...
+raptor.limescop 195.113.144.201 2 u   50   64   377  106.613  ...
+nakor.amazing-i 84.207.3.38     3 u   44   64   377  101.249  ...
```

el `*` indica que ese equipo ha sido elegido como servidor y sus variables son las prestadas a nuestro sistema. El `+` indica que el equipo es un candidato adecuado para la transmisión de información en el caso de que el actual servidor cayese. Pueden darse otros prefijos: `x`, `-`, `\#`, ... (ver la ayuda del manual para más información).

Si deseamos ver las direcciones de los servidores se puede colocar la opción `-n`,

```
[root@linux]# ntpq -pn
```

```
     remote           refid      st t when poll reach   delay   ...
==============================================================================
*80.190.252.238  192.53.103.103  2 u  471  512   37    115.334  ...
+80.86.84.26     195.113.144.201 2 u  469  512   37    106.517  ...
+62.3.211.186    84.207.3.38     3 u  470  512   37    104.858  ...
```

`ntpq` admite un comportamiento interactivo con la opción `-i`, ofreciendo un prompt para que el usuario pueda introducir comandos. Un ejemplo podría ser el siguiente:

```
[root@linux]# ntpq -i
```

```
ntpq>version
ntpq 4.2.0@1.1161-r Tue Aug 30 18:01:56 MDT 2005 (1)
ntpq>cooked
Output set to cooked
ntpq>peers
remote refid st t when poll reach delay offset jitter
==============================================================================
pluto.fips.at 130.149.17.21 2 u 19 64 1 117.275 -252.04 0.002
timp.mcti.ro LOCAL(0) 6 u 18 64 1 178.559 -1249.9 0.002
led.electronest 129.132.2.21 3 u 17 64 1 115.599 -255.73 0.002
ntpq>hostnames no
ntpq>peer
remote refid st t when poll reach delay offset jitter
==============================================================================
80.64.135.105 130.149.17.21 2 u 3 128 7 117.275 -252.04 3.838
80.96.196.58 LOCAL(0) 6 u 2 128 7 145.626 -1328.1 61.988
80.74.144.230 129.132.2.21 3 u 3 128 7 108.422 -251.12 3.774
ntpq>quit
```

Los comandos del modo interactivo pueden ser invocados con la opción `-c`, por ejemplo

```
[root@linux]# ntpq -p
```

sería similar a,

```
[root@linux]# ntpq -c peer
```

- **ntpdc**

Este comando se utiliza para preguntar al demonio `ntpd` sobre su estado y solicitar cambios en dicho estado. Al igual que ocurre con el `ntpq` este comando también admite un modo interactivo. De la misma forma, sin invocar al modo interactivo se pueden ejecutar los comandos de este utilizando la opción `-c`. El siguiente ejemplo indica como mostrar por pantalla estadísticas de los contadores existentes en el módulo del protocolo.

```
[root@linux]# ntpdc -c sysstats
```

```
[root@linux]# ntpdc -c memstats
```

```
[root@linux]# ntpdc -c reslist
```

```
[root@linux]# ntpdc -c sysinfo
```

- **nptime**

Este comando lee y muestra las variables del núcleo relacionadas con el tiempo usuando el sistema de llamadas `ntp_gettime()`. Una salida similar a la dada por este comando se puede conseguir utilizando `ntpdc -c kerinfo`.

```
[root@linux]# ntptime
```

```
ntp_gettime() returns code 0 (OK)
time c852c244.80cf3000 Mon, Jul 3 2006 0:08:04.503, (.503161),
maximum error 380890 us, estimated error 6015 us
ntp_adjtime() returns code 0 (OK)
modes 0x0 (),
offset 525.000 us, frequency 5.304 ppm, interval 4 s,
maximum error 380890 us, estimated error 6015 us,
status 0x1 (PLL),
time constant 4, precision 1.000 us, tolerance 512 ppm,
pps frequency 0.000 ppm, stability 512.000 ppm, jitter 200.000 us,
intervals 0, jitter exceeded 0, stability exceeded 0, errors 0.
```

- **ntptrace**

Muestra por pantalla los servidores NTP hasta la fuente primaria. Cuando nos conectamos al servicio nuesta máquina se corresponde con un *stratrum* o capa, por ejemplo *stratum 3*; esto significa que por encima de nuestro servidor se encuentran otros 2 y por encima de ellos un reloj atómico o un satélite GPS. Si en los servidores no se ha cortado el protocolo de mensajes que proporciona la información a este comando en pantalla se mostrará la jerarquía existente desde nuestra máquina hasta la cabeza. Veamos el siguiente ejemplo:

```
[root@linux]# ntptrace
```

```
localhost: stratum 3, offset 0.000848, root distance 0.128296
ipx10914.ipxserver.de: stratum 2, offset -0.000269, root distance 0.011905
192.53.103.103: timed out, nothing received
***Request timed out
```

Observar que el servidor del *stratum 2* tiene anulado el servicio de mensajería que proporcina la información manejada por `ntptrace`.

- **ntpdate**

El programa `ntpd` corrige la hora de forma gradual. El comando `ntpdate` se utiliza para corregir la hora de forma brusca, en lugar de gradualmente. Tradicionalmente ha sido usuado por los clientes NTP. Se suele utilizar junto a `cron`, de forma que el demonio `crond` ejecuta cada cierta cantidad de tiempo este comando para realizar las actualizaciones del reloj. Así se evitaba el uso del demonio `ntpd`.

Actualmente el comando `ntpd` junto con las opciones `-qxg`, es decir `ntpd -qxg`, realiza la misma operación y se prefiere a `ntpdate`. Para ejecutar este comando no es necesario que esté corriendo el servicio.

- **ntp-keygen**

genera los ficheros de datos criptográficos usados por los esquemas de autenticación y de identificación de NTPv4.

Ejercicio 7.1

Configura un equipo como servidor de horas (en la figura se le ha asignado la dirección IP 192.168.1.128). Este servidor estará conectado a su vez a algún otro servidor de Internet, por ejemplo `pool.ntp.org`.

Para comprobar su funcionamiento configura otro equipo que hará exclusivamente de cliente; este equipo en la figura se corresponde con la máquina con dirección IP 192.168.1.1. Utilizando el comando `ntpdate` en este equipo cliente, comprueba que la hora se actualiza.

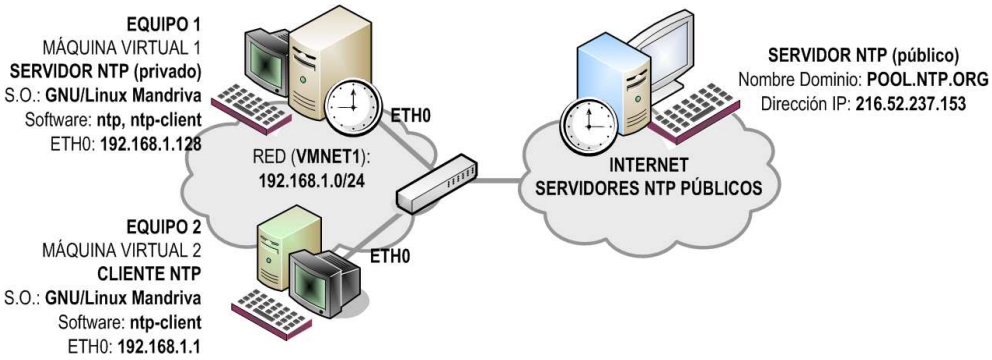


Figura 7.2: Estructura propuesta para el ejercicio práctico.

Solución del ejercicio 7.1

En primer lugar edita el fichero de configuración `/etc/ntp.conf` (equipo servidor)y escribe:

```
server      pool.ntp.org
server      1.europe.pool.ntp.org
server      2.europe.pool.ntp.org
driftfile   /var/lib/ntp/drift
logfile     /var/log/ntpd.log
```

A continuación rearranca el servicio:

```
[root@linux]# service ntpd restart
```

Una vez arrancado nuestro equipo servidor comenzará a buscar algún servidor NTP de los contenidos en `pool.ntp.org` que nos pueda servir. Este proceso puede tardar una cantidad de tiempo considerable en algunas ocasiones.

Nota: Si el tiempo de espera a la conexión es demasiado elevado puedes elegir otro servidor. Un servidor altamente recomendado es `swisstime.ethz.ch`:

```
server swisstime.ethz.ch
```

A continuación tomaremos un equipo (IP 192.168.1.1) e instalaremos sobre él únicamente el paquete `ntp` cliente:

```
[root@linux]# urpmi ntp-client
```

Tras la instalación el equipo cliente tendrá instalado el programa `ntpdate` que nos permitirá la actualización remota.

Teniendo entonces configurada una estructura de trabajo similar a la mostrada en la figura 7.2 sólo tendremos que escribir (puedes cambiar la hora en el equipo cliente mediante el comando `date` antes de hacer el `ntpdate`):

```
[root@localhost prueba]# ntpdate -u 192.168.1.128
Looking for host 192.168.1.128 and service ntp
host found : 192.168.1.128
20 Jul 17:30:12 ntpdate[5687]: step time server 192.168.1.128 offset 25270.297079 sec
```

y la hora en el equipo cliente quedará actualizad con la proporcionada por el servidor. Observa que en este caso había una desviación de 25270.297079 segundos (unas siete horas).

Ejercicio 7.2

Supondremos que tu equipo se encuentra en una red privada (192.168.1.0/24) similar a la mostrada en la figura 7.2. Configura tu máquina para que cada vez que se inicie actualize la hora mediante el comando `ntpdate`.

Solución del ejercicio 7.2

Como ya fue explicado en el capítulo 6 para que un programa (o script) se ejecute al arrancar el equipo es necesario que se encuentre en el directorio `/etc/init.d/`. Además sabemos que para que este programa realmente se ejecute será necesario colocar en `/etc/rc5.d/` un enlace simbólico que apunte a dicho programa.

Nota: Estamos suponiendo que nuestro ordenador arranca en modo 5 (ver `/etc/inittab`), por está razón colocamos el enlace simbólico en `/etc/rc5.d/`. Todo lo concerniente a la ejecución de programas en el arranque puedes volverlo a leer en la página 244. Por tanto los pasos serían:

1. Escribir el programa que provoque la actualización del reloj del sistema y guardarlo en `/etc/int.d/`, por ejemplo con el nombre `actuhora`:

```
#Contenido de /etc/int.d/actuhora
ntpdate -u 192.168.1.128
```

2. Damos permisos de ejecución a este archivo:

```
[root@linux]# chmod u+x /etc/int.d/actuhora
```

3. Creamos el enlace simbólico en la carpeta: `/etc/rc5.d/`:

```
[root@linux]# ln -s /etc/init.d/actuhora /etc/rc5.d/S96actuhora
```

El archivo `S96actuhora` es el vínculo simbólico al archivo `/etc/int.d/actuhora`. Cuando el ordenador arranque en modo 5 llamará al fichero `S96actuhora` que a su vez hará que se ejecute `/etc/int.d/actuhora`.

Ejercicio 7.3

Suponiendo que tu equipo se encuentra en la red 192.168.1.0, dentro de una estructura como la mostrada en la figura 7.2. Configuralo como cliente del servidor de horas, de forma que haga una actualización `ntpdate` cada semana.

Solución del ejercicio 7.3

Para que tu equipo realize una tarea semanalmente basta con configurar el servicio `crond`. Este demonio es un programa encargado de ejecutar tareas de forma periódica. La explicación del funcionamiento de este demonio podría llevar un capítulo completo, pero para configurar tareas sencillas como la propuesta en este ejercicio basta con saber lo siguiente:

En `/etc/` se encuentran los directorios `cron.daily`, `cron.monthly`, `cron.weekly` y `cron.hourly`. Los ficheros contenidos en estos directorios son ejecutados por `crond` diariamente (`cron.daily`), mensualmente (`cron.monthly`), semanalmente (`cron.weekly`) y cada hora `cron.hourly`.

Por tanto para realizar el ejercicio propuesto basta con que introduzcas en el directorio `/etc/cron.weekly/` un archivo con un contenido similar a,

```
#Contenido del archivo actualizar-hora
ntpdate -u 192.168.1.128
```

Este archivo debe pertener a `root` y tendrá permisos de ejecución para todos los usuarios. Suponiendo que lo llamamos `/etc/cron.weekly/actualizar-hora`,

```
[root@linux]# chmod u+x,g+x,o+x /etc/cron.weekly/actualizar-hora
```

Ejercicio 7.4

Configura un servidor de horas que trabajará dentro de una red de área local dando servicio a los ordenadores de la misma. La red tiene configurado un firewall para proteger la intranet, por lo que, deberás configurarlo para que deje pasar el tráfico NTP y así el servidor pueda tomar únicamente las actualizaciones del exterior.

Vuelve a realizar la configuración del firewall para que el servidor de nuestra LAN acepte clientes del exterior

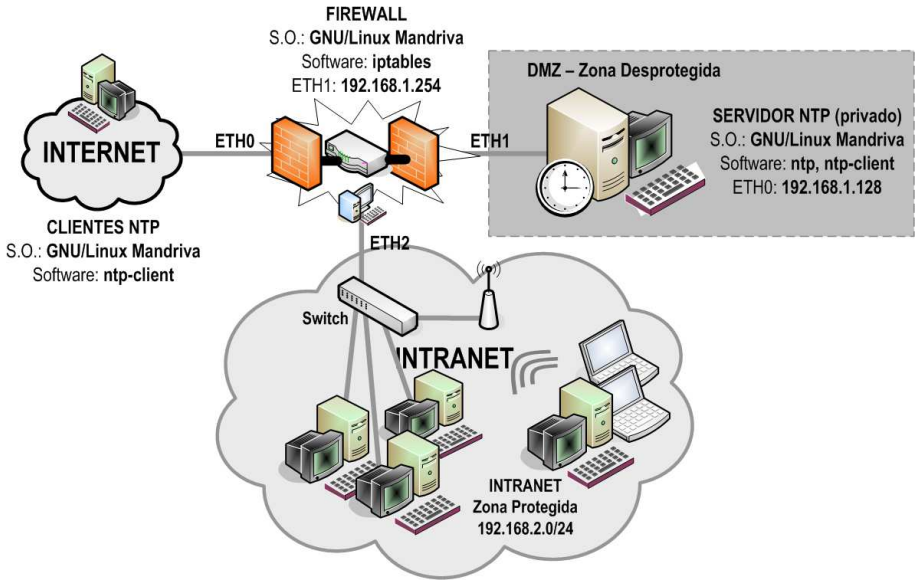


Figura 7.3: Esquema de la red propuesta en el ejercicio 7.4

Solución del ejercicio 7.4

En primer lugar editaremos el fichero de configuración `/etc/ntp.conf` que deberá contener algo similar a:

```
server      swisstime.ethz.ch
server      pool.ntp.org
driftfile   /var/lib/ntp/drift
logfile     /var/log/ntpd.log
```

Antes de arrancar el servicio deberemos asegurarnos que el firewall no corta el paso de los paquetes UDP que necista el servidor de horas. El puerto de comunicación utilizado por el servicio es el 123, así la línea `iptables` a introducir en el equipo que hace de firewall será:

```
iptables -t filter -A FORWARD -i eth1 -o eth0 -p udp --dport 123 -j ACCEPT
iptables -t filter -A FORWARD -m state --state STABLISH,RELATED -j ACCEPT
```

La primera de las reglas permite únicamente el peticiones desde la LAN hacia el exterior por el puerto 123. La segunda regla permite que las contestaciones a las posibles peticiones de la LAN atraviesen el firewall.

Si deseamos que nuestro equipo tenga la posibilidad de hacer de servidor de horas para equipos clientes de la Internet es necesario permitir el tráfico en los dos sentidos. En este caso las reglas serían:

```
iptables -t nat -A PREROUTING -p udp --dport 123 -j DNAT --to 192.168.1.128
iptables -t filter -A FORWARD -p udp --dport 123 -j ACCEPT
```

Apéndice A

Configuración de redes virtuales mediante VMware

VMware es un software que crea máquinas virtuales. Permite comunicar estas y la máquina física entre sí mediante dispositivos de interconexión virtuales en red (red virtual). De esta forma ofrece la posibilidad de implementar cualquier configuración en red que imaginemos. El único requisito es disponer de los suficientes recursos (memoria, periféricos, etc.) en la máquina física, ya que evidentementes estos recursos se deben repartir entre las distintas máquinas virtuales sin que se produzca una degradación desmesurada de rendimiento.

Como instalar VMware, que requisitos son necesarios, que posibilidades nos ofrece y demás aspectos de su funcionamiento podemos encontrarlo detallado en el libro de **Instalación y Mantenimiento de Servicios de Redes de Área Local**[1]. En las próximas líneas se hará una pequeña descripción sobre la gestión de las redes virtuales que pueden ser creadas entre nuestra máquina física y el resto de máquinas virtuales utilizando las herramientas de VMware. Aunque a los autores les gustaría pensar que los principales destinatarios de este libro, alumnos de ciclos formativos, disponen de VMware sobre GNU/Linux, lo cierto es que en la mayoría de los I.E.S. las máquinas funcionan bajo Windows. Por comodidad hacia estos usuarios se asumirá que en la máquina física corre Windows.

Para realizar los casos prácticos, expuestos a lo largo del libro, mediante el uso de máquinas virtuales será necesaria la configuración de las redes virtuales (VMnet). Para ello, deberemos ejecutar `Manage Virtual Networks`[2] (ver figura A.1).

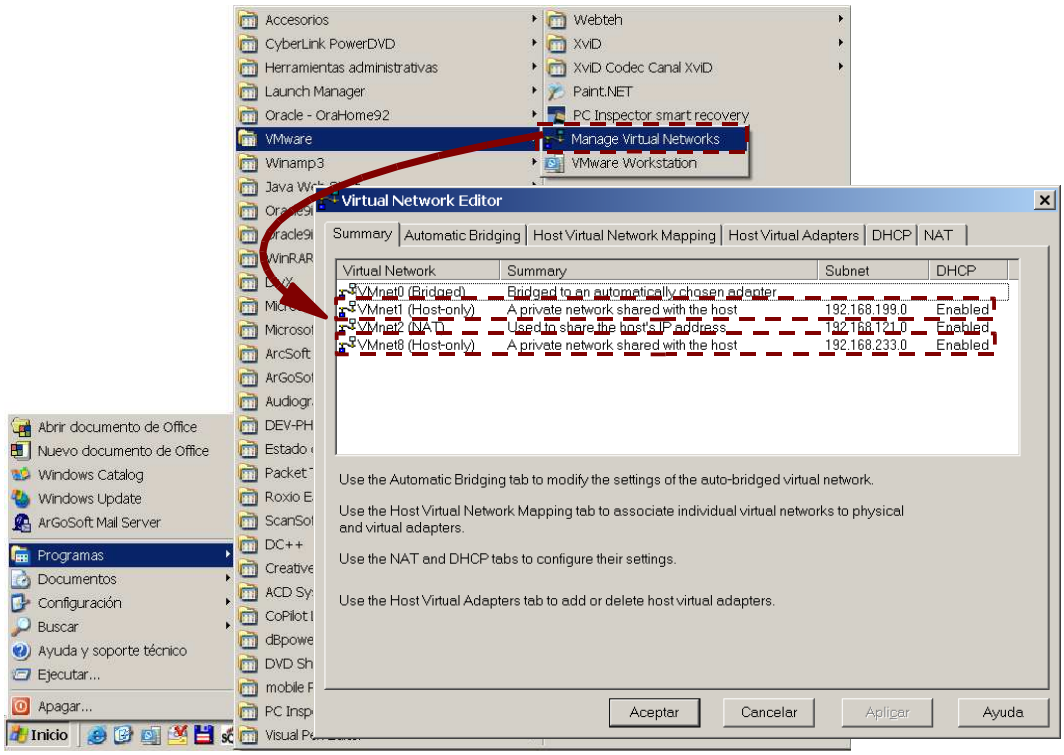


Figura A.1: Acceso en Windows al Mange Virtual Networks.

Como se puede observar en la captura de pantalla anterior, tras instalar en nuestro equipo VMware quedan habilitadas tres opciones de configuración en red: `Bridged`, `Host-only` y `NAT`. De las tres vamos a describir únicamente la `VMnet Host-only` ya que se va a asumir que no disponemos de dispositivos físicos de interconexión externos para poder implementar la opción `Bridged`; por su parte la opción `NAT` no nos sirve ya que configuraría de forma automática NAT, y esto es precisamente uno de los conceptos que se desarrolla en este libro (capítulo 6).

[1] Publicado por `Mira Editores`, ISBN 84-8465-185-1 y escrito por los mismos autores que este libro.
[2] Herramienta software de gestión de las redes virtuales VMware.

En concreto, se encuentran habilitadas dos redes lógicas virtuales (VMnet) del tipo `Host-only`: la `VMnet1` `192.168.199.0` y la `VMnet8` `192.168.233.0`. Esto permite implementar y probar entornos de red como el que se presenta en la figura A.2.

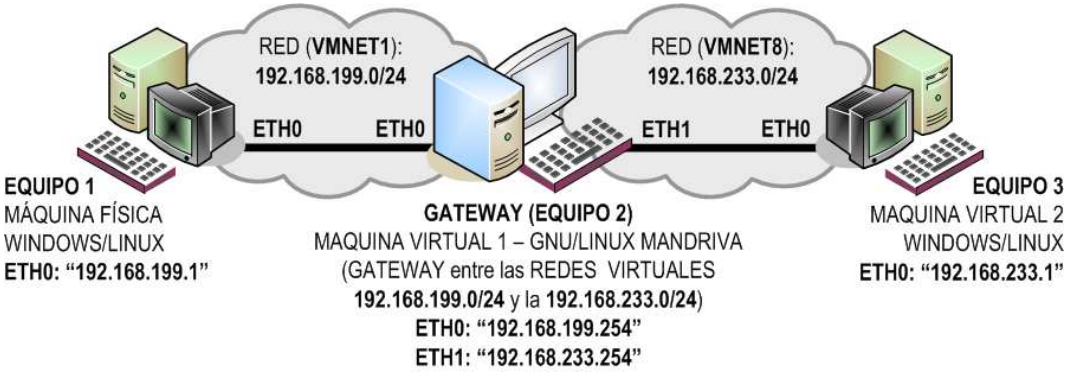


Figura A.2: Redes virtuales comunicadas a través de un `gateway`.

Si se desea emular y probar el funcionamiento de un `gateway` en GNU/Linux sin más equipación que nuestro equipo físico y VMware, sería necesario disponer de dos `VMnet Host-only` habilitadas y tener creadas al menos dos máquinas virtuales (al menos una de ellas corriendo bajo GNU/Linux para que haga de `gateway`). Para garantizar que el equipo 1 y el equipo 3 tan sólo puedan comunicarse a través del `gateway` será necesario asegurarnos que la máquina física (equipo 1) tan sólo tiene habilitada la interfaz de red virtual asociada a la `VMnet1` estando el resto deshabilitadas. La segunda máquina virtual sólo tiene configurada una única interfaz de red asociada a la `VMnet8`. En caso contrario, es posible que la comunicación entre ellas se realice a través de otro mecanismo diferente al `gateway`[3].

Para advertir que interfaces de red tenemos activadas en nuestra máquina física (Windows) tan sólo deberemos pinchar con el botón derecho del ratón sobre `Mis sitios de red` y seleccionar `propiedades`:

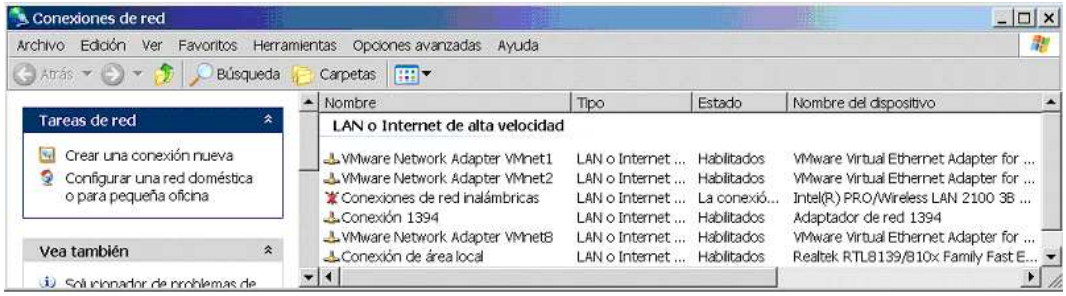


Figura A.3: Vista en Windows de las conexiones de red disponibles.

Para deshabilitar una de estas interfaces de red tan sólo es necesario pinchar con el botón derecho del ratón sobre ella y seleccionar `desactivar`.

Por otro lado, para hacer cambios en una máquina virtual, esta debe estar apagada. Imaginad entonces que disponemos de una máquina virtual apagada, a la que deseamos configurar las interfaces de red. Deberemos hacer doble click sobre su tarjeta de red (NIC) y en la ventana que aparece, `settings`, impondremos a través de la opción `Custom` la VMnet a la que deseamos incorporarla.

Si se desea asignar más de una interfaz de red a nuestra máquina virtual pincharemos[4] en la opción `Edit Virtual Machine Settings` tras lo cual nos aparecerá el `Virtual Machine Control`

[3] Si se encuentra habilitada la interfaz de red asociada a la `VMnet8` en la máquina física, se establecerá comunicación con la segunda máquina virtual a través de esa red virtual sin necesidad de pasar por ningún `gateway`.

[4] Recuerda que la máquina virtual debe estar apagada, en `off`

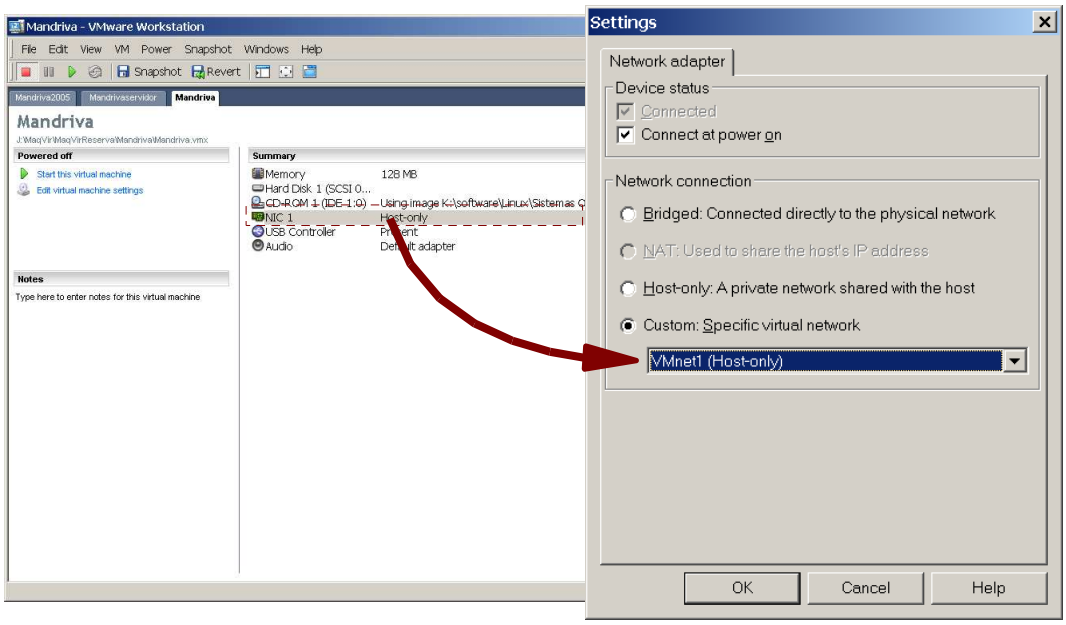


Figura A.4: Procedimiento de configuración de las interfaces de red de una máquina virtual (debe estar a `off`).

`Panel`. Desde allí podremos tanto quitar (Remove) como añadir (Add) hardware a la máquina virtual. Como en este caso estamos interesados en añadir una nueva interfaz de red, pinchando en `Add` nos aparecerá un asistente, `Add Hardware Wizard`, que nos preguntará que deseamos añadir, `Ethernet Adapter`, y a que red virtual (`VMnet`) deseamos agregarla, `Custom` (por ejemplo, `VMnet8`). En la figura A.5 se muestra gráficamente este procedimiento.

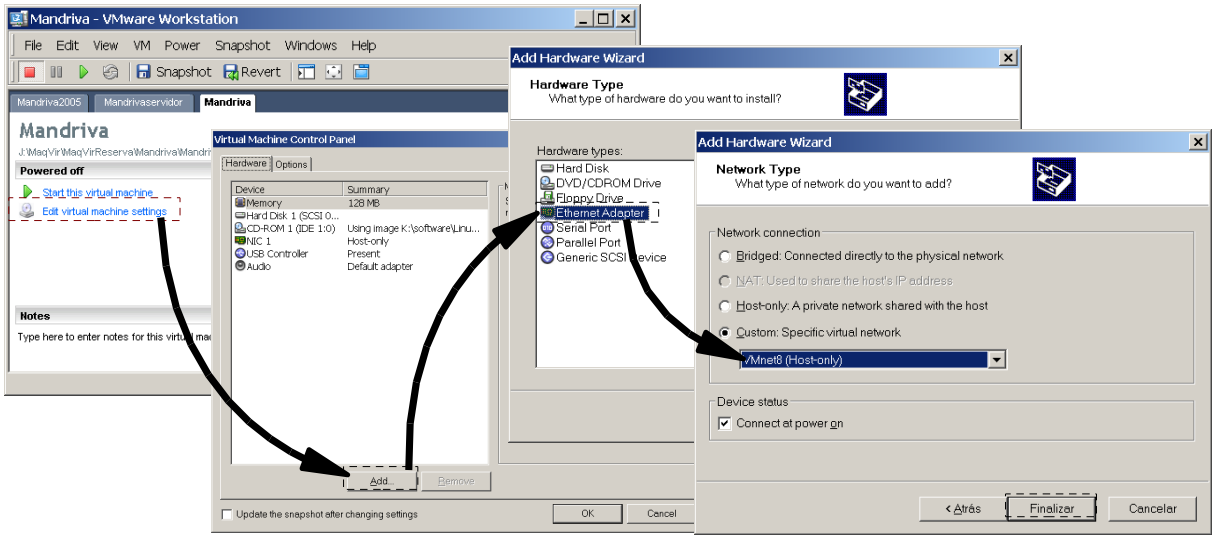


Figura A.5: Procedimiento para asignar varias interfaces de red a una máquina virtual.

Así, nuestra máquina virtual estaría configurada con dos interfaces de red. La NIC1 dentro de la VMnet1, y la NIC2 dentro de la VMnet8.

También puede darse la situación de que el supuesto práctico que se plantee sea más complejo y sea necesario formar un número mayor a dos redes. En este caso se debe habilitar una nueva VMnet. La figura A.6 muestra los pasos necesarios para ello. Ejecutaremos nuevamente el `Manage Virtual Networks` (herramienta software de gestión de las redes virtuales VMware), y dentro de la pestaña

`Host Virtual Adapters` añadiremos un nuevo adaptador virtual (`Add new adapter..`) a nuestra máquina física habilitando de esta forma una nueva red virtual, `VMnet` (por ejemplo, `VMnet3`). La creación de una nueva red virtual la podemos ratificar a través de la pestaña `Summary`.

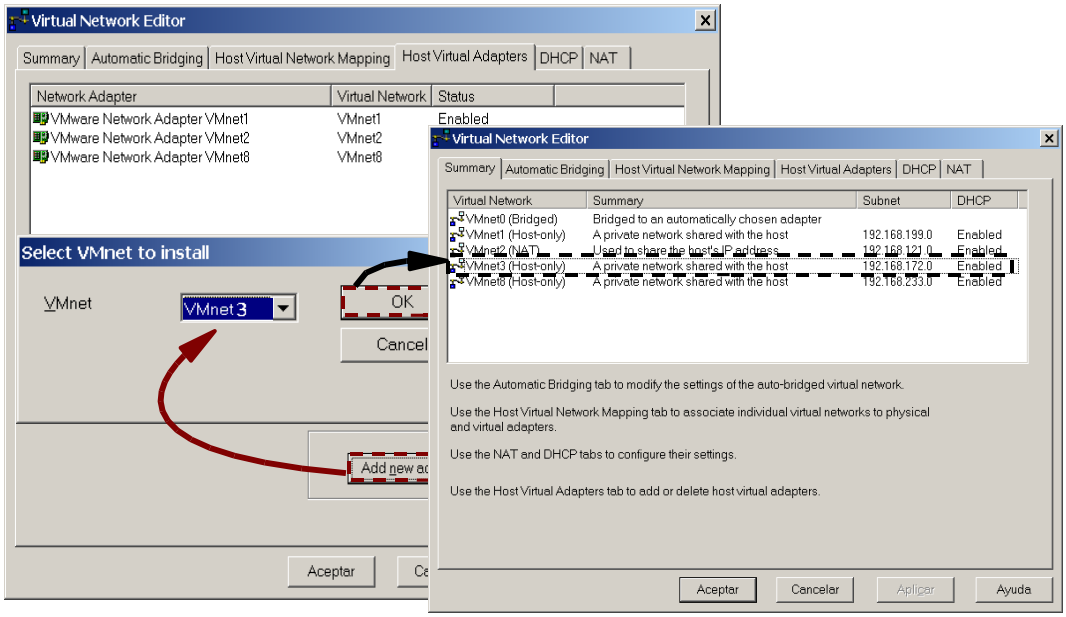


Figura A.6: Procedimiento para añadir nuevas redes virtuales.

Así podría ser comprobado, con tan sólo un equipo físico y `VMware`, el correcto funcionamiento de un `router/gateway` encargado de comunicar entre sí tres redes, tal y como se muestra en la figura A.7.

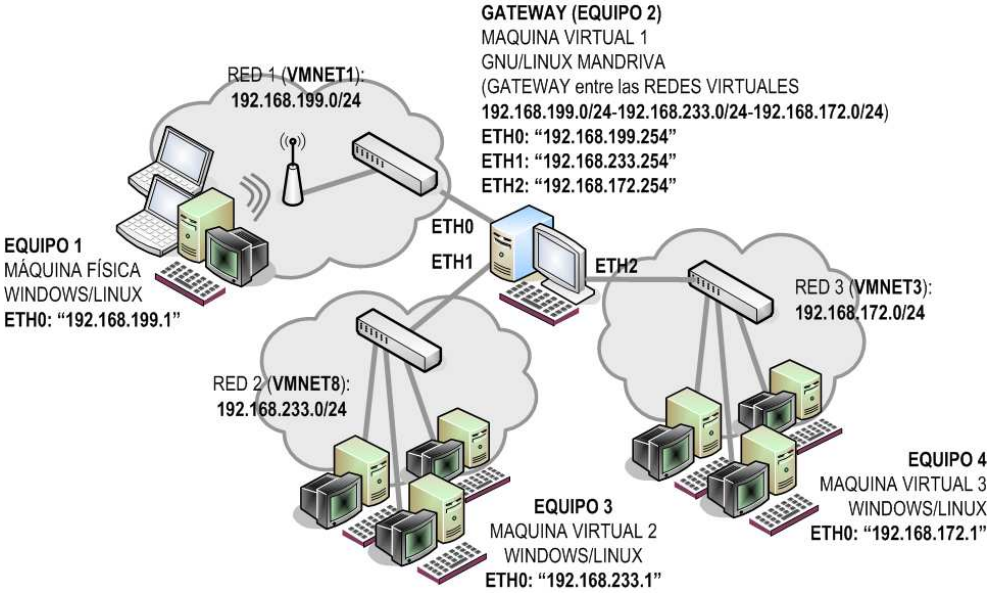


Figura A.7: Emulación de la comunicación entre tres redes mediante VMware.

Para que en nuestra máquina física pueda correr Windows, y tres máquinas virtuales con sus correspondientes sistemas operativos de manera simultánea se necesita cubrir unos requerimientos de hardware importantes de tal forma que al repartir estos entre todas las máquinas no se aprecien retardos excesivos (sobre todo en cuanto a cantidad de memoria RAM, al menos 1GBytes).

Apéndice B

Descripción de algunos comandos GNU/Linux utilizados

En el capítulo dedicado al servidor de páginas Web `apache` se hacen uso de comandos GNU/-Linux que quizás no conozcas. Aquí se hace un pequeño resumen de los que necesitas conocer para realizar los ejercicios propuestos en este libro relacionados con los scripts CGI.

Descripción de algunos de los comandos GNU/Linux empleados

- **read *variable***

Guarda en la variable especificada la información procedente de la entrada estándar. Si tenemos en cuenta que al pulsar sobre el botón `submit` del formulario este envía su contenido a la entrada estándar del fichero indicado en `action`, a través de `read`, la variable indicada recogerá toda la información introducida en el formulario.
Ejemplo:

```
read DATOS_ENVIADOS
```

La sintaxis con la que se almacena la información en la variable especificada es la siguiente: `<name campo1 formulario>=<value campo1>&<name campo2 formulario>=<value campo2>&...`. Es decir, cuando creamos un formulario en HTML cada uno de los campos que lo forman se identifican a través de su parámetro `name`, y el valor de este se almacena mediante un parámetro `value`, los cuales son enviados al script indicado en `action` en el formato anterior con la finalidad de que este sepa distinguir claramente quien es quien. Por ejemplo, si nuestro formulario fuese el siguiente:

```
<form method="post" action="/scripts/programa1.pl">
<input type="text" name="nombre" size="30">
<input type="text" name="num_registro" size="6">
</form>
```

Al escribir en la primera caja de texto (`input text`) del formulario `Manuel Morera Zapater` y el la segunda `182193`, cuando la instrucción `read DATOS_ENVIADOS` se ejecutase en el script CGI, su contenido sería a ser el siguiente:

```
read DATOS_ENVIADOS
```

⇓

```
[user@linux]$ echo $DATOS_ENVIADOS
nombre=Manuel+Morera+Zapater&num_registro=182193
```

Se encadenan los distintos campos que forman parte del formulario con el formato `name1=value&name2=value&name3=value&...` y en el caso de existan espacios en blanco en el valor asignada a los campos del formulario, estos se sustituyen por signos `"+"`. Esto nos obligará a que en nuestro `shellscript`, una vez recogida la información enviada desde el formulario, tengamos que trocear y reconstruir cada una de las variables:

```
read DATOS_ENVIADOS
NOMBRE=`echo $DATOS_ENVIADOS | cut -d"&" -f1 | cut -d"=" -f2 | tr -s "+" ""`
NUM_REGISTRO=`echo $DATOS_ENVIADOS | cut -d"&" -f2 | cut -d"&" -f2`
```

⇓

```
[user@linux]$ echo $NOMBRE
Manuel Morera Zapater
[user@linux]$ echo $NUM_REGISTRO
182193
```

- **cut**

Permite cortar las columnas de un fichero formado por múltiples filas divididas por campos. Para ello al menos será necesario acompañarlo de dos parámetros:
1. `-d"carácter"`: Informa al comando `cut` de cual es el carácter delimitador utilizado en el fichero especificado para dividir las filas en campos.
2. `-f"numeros de columna"`: informa a `cut` de que campos (columnas) desea recuperar.
Ejemplo:

```
cut -d":" -f"1,2" /home/arturo/base_datos_sencilla.db
```

lo cual sería equivalente a:

```
more /home/arturo/base_datos_sencilla.db | cut -d":" -f"1,2"
```

El comando `cut` también puede trocear el contenido de una variable, rescatando los campos que se deseen (ver explicación del comando `read` en donde se muestran ejemplos).

- **tr**

Permite intercambiar y eliminar los patrónes especificados mediante el uso de las opciones `-s` y `-d` respectivamente.
1. `-s` *patrón1* *patrón2*: Intercambia *patrón1* por *patrón2*.
Ejemplo:

```
[user@linux]$ PROVINCIAS="Albacete+Avila+...+Zaragoza"
[user@linux]$ echo $PROVINCIAS | tr -s "+" ":"
Albacete:Avila:...:Zaragoza
```

2. `-d` *patrón*: Elimina todas las coincidencias encontradas con el *patrón* especificado.
Ejemplo:

```
[user@linux]$ echo $PROVINCIAS | tr -d +

AlbaceteAvila...Zaragoza
```

- **grep *patrón***

Permite filtrar el contenido de un fichero o de una variable, según existan coincidencias o no con el patrón indicado. Por ejemplo, para comprobar que filas de un fichero contienen la palabra clave `ingeniero` ejecutaríamos el comando siguiente:

```
cat /home/usuario/empleados | grep "ingeniero"
```

Apéndice C

Pequeña descripción del lenguaje PERL

En el capítulo dedicado al servidor de páginas Web `apache` se hace uso del lenguaje `PERL` en algunos de los ejercicios propuestos. A continuación se muestra un pequeño listado con los comandos más relevantes de `PERL` y que nos permitirán entender perfectamente dichos ejercicios.

Definición de variables (no tipificadas): Escalares, Listas y Listas Asociativas

- **$variable**

Definición de una variable/lista no tipificada. Esto significa que permite almacenar indistintámente un valor numérico o una cadena alfanumérica:
Ejemplos:

```
$numero=int(rand($cantidad)); //Parte entera de un nº aleatorio entre 0 y $cantidad
```

```
$cadena="El numero de alumnos es $cantidad";
```

- **@lista**

Ejemplos:

```
@lista=("Manuel",76,"Cecilia",126,"Andres",23,...);
```

```
$lista[0]="pedro";
```

- **%lista_asociativa**

Ejemplos:

```
%notas=("Alonso" => 9.25, "Becerril" => 3.45, ...);
```

```
$resultado=$notas{"Becerril"};
```

Cabe mencionar la existencia de variables predefinidas en `perl` entre las cuales podríamos destacar `%ENV`. Esta es una lista asociativa compuesta por todas las variables de entorno y su valor correspondiente. Entre ellas señalar las siguientes:

`$ENV{"SERVER_NAME"}`: Contiene el nombre del equipo servidor que ejecuta el script perl.
`$ENV{"SERVER_PORT"}`: Informa del puerto por el que da servicio el servidor Web.
`$ENV{"SERVER_ADDR"}`: Contiene la dirección IP del equipo servidor.
`$ENV{"SERVER_SCRIPT_FILE_NAME"}`: Contiene el path o ruta absoluta del script perl que se esta ejecutando.
`$ENV{"REMOTE_USER"}`: Informa del nombre del usuario (login) que se autentifica en un sitio Web no anónimo.

`$ENV{"REMOTE_ADDR"}`: Contiene la dirección IP del equipo que llevo a cabo la solicitud Web sobre el script perl al servidor.
`$ENV{"QUERY_STRING"}`: Recoge la información que se envía desde un formulario HTML en el caso de hacer uso del `"method=GET"` en lugar de `POST`.

Recepción de la información enviada desde un formulario

- **<STDIN>**

Una de las finalidades más importantes de los sitios Web dinámicos es posibilitar la interacción con el usuario mediante el intercambio de información. Para ello, en la mayor parte de los casos se hace uso de formularios Web, donde mediante la cumplimentación de sus campos, se permite al usuario llevar a cabo acciones tan dispares como realizar cambios en la configuración del servidor (gestión vía Web), consultar y actualizar bases de datos, o alterar el propio contenido del sitio Web (por ejemplo, foros de discusión, biblioteca de imágenes, etc.). Su instrucción equivalente en lenguaje `shellscript` es `read` mostrada en el ejercicio anterior.

Mediante la instrucción <STDIN> el script `Perl` permanecerá a la espera de que le lleguen datos por su entrada estándar (`STanrD Input`). Esta instrucción es fundamental, ya que cuando, desde un formulario, se envían datos a una página Web compuesta por un script (<`form method="POST" action="script.pl"`>), se hace por defecto a su entrada estándar. *Ejemplo:*

```
$datos_recibidos=<STDIN>;
```

Salida de mensajes por la pantalla del cliente Web

- **print *mensaje*;**

Imprime el mensaje especificado. *Ejemplos:*

```
print "Bienvenido sr/sra. $nombre";
print "El resultado es ",log($cantidad); #logaritmo neperiano
```

- **print "Content-type: text/html\n\n";**

Por defecto la salida estándar de los mensajes es el monitor. Esta línea es la primera escrita en `perl` que debe aparecer en el `script`. Informará al sistema de que el destinatario de los mensajes es un intérprete `HTML` o cliente Web. De esta forma podremos imprimir cualquier etiqueta `HTML` como si se tratase de un mensaje más. Estos serán interpretados posteriormente en el cliente Web:
Ejemplos:

```
print "<hr> <br> <h1>MI TITULO</h1> <hr>";
print "<form method='post' action='script.pl'>...</form>";
```

Operadores matemáticos y de relación

- **Operadores +, -, *, /, %, **, +=, -=, *=, **=, /=, %=**

De los operadores matemáticos y de asignación, tan sólo comentaremos algunos de ellos.
`%` : Sirve para calcular el resto de una división (10 %3=1).
`**` : Sirve para calcular potencias (5**2=25).
Los operadores de asignación se entenderán mejor mediante los siguientes ejemplos:

```
$resultado += 12;   #equivale a $resultado=$resultado+12;
$valor **= 3;       #equivale a $valor=$valor**3; Eleva al cubo $valor.
$cantidad /= 5*8;   #equivale a $cantidad=$cantidad / (5*8);
```

- **Operadores ==, < ,> ,<= ,>=, !=, <=>**

Operadores utilizados para comparar números: (`==`) igual que, (<) menor que, (>) mayor que, (`<=`) menor o igual que, (`>=`) mayor o igual que y por último (`<=>` ó `!=`) distinto que.

- **Operadores gt, lt, eq, ge, le ne, cmp**

Estos operadores son utilizados para comparar cadenas alfanuméricas. (`gt`) mayor que, (`lt`) menor que, (`eq`) igual que, (`ge`) mayor o igual que, (`le`) menor o igual que, (`ne`) distinto que, (`cmp`) comparador.

- **Operadores &&, | |, !**

Estos son los operadores lógicos. (`&&`) AND, (| |) OR, (`!`) NOT.

Operadores y funciones sobre cadenas de caracteres

- **Operador "."**

Se utiliza para concatenar cadenas alfanuméricas.
Ejemplo:

```
$carta="El señor ".$nombre." ha obtenido una nota de ".$notas{"$nombre"}.".";
print "$carta";
```

- **Operador ".="**

El siguiente ejemplo mostrará su uso:

```
$carta .= "Enhorabuena."; #equivale a: $carta = $carta."Enhorabuena.";
```

- **length(*cadena*);**

Devuelve la cantidad de caracteres que componen la cadena especificada.
Ejemplo:

```
$longitud=length($contraseña);
```

- **index(*cadena,patron,ini*);**

Informa de la posición donde encuentra el patrón especificado buscada a partir de la posición especificada en `ini` (opcional).
Ejemplo:

```
$posicion=index("/home/juan","/",2);
```

- **substr(*cadena,n,m*);**

Extrae `"m"` caracteres de la cadena especificada a partir de la posición `"n"`.

```
$usuario=substr(/home/juan,$posicion+1,length(/home/juan)-($posicion+1));
print "La ruta especificada es el HOME del usuario $usuario";
```

- **rindex(*cadena,patrón,ini*);**

Realiza la búsqueda del `patrón` especificado en la cadena indicada, pero al contrario que `index`, comienza la búsqueda desde el final de la cadena hacia el principio de ésta, a partir de la posición `ini`.

- **split(/*separador*/,*cadena*);**

Trocea la cadena de caracteres indicada rompiéndola donde aparece el carácter delimitador o separador especificado. Como resultado devuelve una lista de variables con tantas posiciones como trozos se han obtenido. Es importante advertir que en aquellos casos en los que el carácter delimitador tiene otro sentido dentro de `perl` (por ejemplo "+","*","/",...), para evitar confusión debe ir acompañado de una contrabarra (\).
Ejemplos:

```
@nombres=split(/:/,"fernando:manolo:arturo:juanjo:celia:carol");
```

```
$informatica="analisis+programación+redes+bases datos+servicios Internet+SO";
@asignaturas=split(/\+/,$informatica);
```

Equivale al comando `cut` de los shellscript.

- **join(*separador,lista*);**

Agrupa los elementos que componen la lista en una única cadena de caracteres delimitados por el carácter separador especificado.

```
$encadenado=join(":",@usuarios);
```

- **chop(*contenido*);**

Elimina los saltos de línea, es decir los caracteres de control `"\n"`, `"\n\r"` en la cadena de caracteres suministrada.

- **sort(@*lista*); y reverse(@*lista*);**

`sort` devuelve una nueva lista con los componentes de lista ordenados de menor a mayor. Por su parte `reverse` hace la ordenación de mayor a menor.
Ejemplos:

```
@comunidades=("Madrid","La Rioja","Navarra","Aragon","Cantabria");
@ordenados=sort(@comunidades); #Ordena alfabéticamente la lista.
```

- **push(@*lista,valor*); y $elemento=pop(@*lista*);**

Formas de expresar que se añada (`push`) un nuevo elemento a la lista en la posición final, o que devuelva y elimine (`pop`) el último elemento de la lista especificada.

- **shift(@*lista*); y unshift(@*lista,elemento*);**

`shift` devuelve y elimina el primer elemento de la lista, desplazando una posición menos la ubicación del resto de elementos de la lista. Por su parte `unshift` inserta un elemento al comienzo de la lista, desplazando el resto de elementos una posición más.

- **delete(*elemento*);**

Devuelve y elimina el elemento especificado de una lista asociativa.
Ejemplo:

```
%nivel_pantanos=("Santolea" => 52, "Pena" => 65,...);
$nivel_borrado=delete($nivel_pantanos{"Santolea"});
```

- **grep(/*patrón*/,@*lista*);**

Devuelve una nueva lista con los elementos que dentro de la lista especificada encajan con el `patrón` de búsqueda indicado. Para especificar el patrón, al igual que con el comando `grep` de UNIX/Linux, puede hacerse uso de comodines como `?` (0 ó 1), `*` (0 ó más), `+` (1 ó más), de caracteres especiales como ^ (desde el comienzo de la cadena), `$` (fin de la cadena), o especificación de rangos [`A-Z`], [`bv`], ...
Ejemplos:

```
@alumnos=("cecilia";"ariadna";"hugo";"juanjo";"julian";"andres";"jose");
@filtrados=grep(/^a/,@alumnos); #filtra los que empiezan por "a": ariadna
@filtrados=grep(/jo/,@alumnos); #filtra los que contienen "jo": "juanjo", "jose"
```

Manejo de ficheros

- **open**

La instrucción `open` nos permite abrir el fichero que especifiquemos. La sintaxis a utilizar es:

```
open(nombre_MANEJADOR_del_fichero,modo_de_apertura nombre_del_fichero)
```

Como se puede observar, además de indicar a `open` cual es fichero que deseamos abrir, debemos indicar de que modo (lectura, escritura, ...), y el `MANEJADOR` del que haremos uso para hacer referencia a su contenido. Este `MANEJADOR` lo podemos asociar a un puntero que nos permite recorrer el contenido del fichero, permitiéndonos extraer la información que le indiquemos.

De entre los modos de apertura cabría destacar:
<: Indica a `open` que deseamos abrir el fichero en modo lectura.
>: Indica a `open` que deseamos abrir el fichero en modo escritura. Si el fichero no esta vacío se perderá su contenido, es decir, se sobreescribirá.
>>: Indica a `open` que deseamos abrir el fichero en modo escritura, pero colocando el puntero al final de su contenido. Esto nos permitirá añadir información al fichero, mantenido su contenido previo.
Por ejemplo, si quisiéramos abrir un fichero llamado `basedatos.txt` en modo escritura con la finalidad de añadir un nuevo registro ejecutaríamos:

```
open(BASEDATOS,">>basedatos.txt");
```

- **<MANEJADOR>**

Una vez abierto un fichero, haremos uso del `MANEJADOR` indicado en `open` para llevar a cabo lecturas o escrituras sobre el mismo. Para ello, encerraremos el nombre del manejador del fichero entre < >.

Por ejemplo, si deseamos leer el contenido del fichero a partir de la posición apuntada por el manejador simplemente tenemos que invocarlo y guardar este en una variable:

```
open(PRODUCTOS,"<listaproductos.txt");
@lineas=<PRODUCTOS>;
```

En este caso, la variable de tipo lista @lineas recogería el contenido total del fichero apuntado por el manejador `PRODUCTOS`, almacenando en cada una de las posiciones de la lista, cada una de las líneas del fichero.

En cambio, si el fichero ha sido abierto en modo escritura, mediante el comando `print` imprimemos la cadena de caracteres indicada a partir de la posición apuntada por el manejador del fichero:

```
open(ALUMNOS,">>/var/www/html/webies/alumnos.txt");
print ALUMNOS "$nombre:$apellidos:$edad:$curso:$codigo_alumno:$direccion";
```

- **close**

Cierra el fichero apuntado por el manejador indicado.
Ejemplo:

```
close(BASEDATOS);
```

- **tell**

Función que nos permite conocer la posición del puntero dentro del fichero (bytes). Esto nos informa de la posición a partir de la cual llevaremos a cabo la lectura o escritura.
Ejemplo:

```
$posicion=tell(MANEJADOR);
```

- **eof**

Función que nos informa si la posición apuntada por el manejador es el final del fichero. En concreto devuelve un `1` (`TRUE`, verdadero) si nos encontramos en tal posición, y un `0` (`FALSE`, falso) en caso contrario.
Ejemplo:

```
$final=eof(MANEJADOR);
```

- **seek**

Sitúa el puntero especificado en la posición deseada dentro del fichero. Esta posición se especifica en relación al comienzo del fichero (posición `0`), a la posición actual del puntero dentro del fichero (posición `1`), o en relación a la posición final (posición `2`). La sintaxis de la instrucción es la siguiente:

```
seek(MANEJADOR,desplazamiento,posición_relativa);
```

Por ejemplo, si quisiéramos colocar el puntero 20 posiciones (bytes) atrás en relación a la posición actual se debería escribir,

```
seek(BASEDATOS,-20,1);
```

- **read**

Permite leer del fichero especificado una cantidad de bytes dados a partir de la posición indicada. Por defecto, esta lectura se realizará contando a partir de la posición actual del puntero dentro del fichero. En caso contrario será necesario especificarlo explícitamente. Su sintaxis es:

```
read(MANEJADOR,variable_que_almacenara_los_bytes_leídos,cantidad,[posicion_relativa]);
```

Ejemplo:

```
read(BASEDATOS,$contenedor,50);
```

En el ejemplo mostrado se leerán 50 bytes a partir de la posición actual y los almacenará en la variable $contenedor.

Estructuras de control de flujo

- **if elseif else**

Sin lugar a duda, la estructura de control de uso más habitual es `if`. La sintaxis es:

```
if condición
  { intrucciones; }
elsif condición
  { intrucciones; }
else
  { intrucciones; }
```

Esta estructura nos permite reconducir el flujo de ejecución de un programa en función del resultado de una condición planteada. Además mediante `elsif` podemos establecer varias condiciones en cascada.
Ejemplo:

```
if ($empleado eq "director")
  { print "Bienvenido Señor Director, espero que su estancia sea grata."; }
elsif ($empleado eq "jefeseccion")
  { print "Hola. Recuerda que el trabajo debe estar acabado en Julio."; }
else
  { print "Buenos días, manos a la obra y a trabajar."; }
```

- **while**

La sintaxis es:

```
while (condición)
{ instrucciones; }
```

En el caso de que deseemos ejecutar un conjunto de instrucciones de manera iterativa mientas se cumpla una determinada condición haremos uso de la estructura `while`. Por ejemplo, si quisieramos imprimir la lista de impresoras que se encuentra en el fichero `lista_impresoras.txt` escribiríamos:

```
open(IMPRESORAS,"<lista_impresoras.txt");
while (!(eof(IMPRESORAS))) {
$linea=<IMPRESORAS>;
print "<br>$linea"; }
close(IMPRESORAS);
```

- **last**

En bucles iterativos (`while`, `until` o `for`) nos puede interesar interrumpir el bucle de ejecución. En tal caso haremos uso del comando `last`.

- **foreach**

La sintaxis es:

```
foreach $var (@lista)
{ instrucciones; }
```

En muchas ocasiones nos puede interesar rastrear el contenido de una lista para llevar a cabo alguna comprobación. En tal caso haremos uso de `foreach`. A modo de ejemplo, realizaremos la misma función que la mostrada con `while`:

```
open(IMPRESORAS,"<lista_impresoras.txt");
@lineas=<IMPRESORAS>;
foreach $linea (@lineas) {
print "<br>$linea"; }
close(IMPRESORAS);
```

- **goto *etiqueta***

Instrucción ampliamente utilizada en los lenguajes ensamblador, nos permite redirigir el flujo del programa hacia la instrucción del código que contiene la etiqueta especificada.

```
...#Líneas anteriores del código del script perl
if ($signal_externa == 12) {
goto signal12;
}
...#resto del código del script perl
signal12: print "Se acaba de recibir una señal de alarma ...";
```

Expresiones regulares en PERL

- **$var=~/*patrón de busqueda*/**

Las expresiones regulares son distintas formas de establecer un filtrado en la información que indiquemos. Es posible realizar, desde búsquedas en base a un patrón determinado, hasta sustituciones.
En este caso, `$var=~/patrón de busqueda/` permite realizar búsqueda de patrónes dentro de la variable especificada. Por ejemplo, si quisiéramos buscar dentro de una base de datos la información correspondiente a un usuario especificado ($nombre) escribiríamos lo siguiente:

```
open(USUARIOS,"</var/www/html/miweb/lista_usuarios.txt");
@lineas=<USUARIOS>;
foreach $linea (@lineas) {
if ($linea =~/$nombre/) {
print "$linea";} }
close(USUARIOS);
```

Nota: En GNU/Linux para escribir el carácter ~ puede hacerse pulsando la combinación de teclas `AltGr+ñ` o `AltGr+4`.

A la hora de establecer el patrón de búsqueda nos podemos ayudar de filtros parecidos a los utilizados en `shellscripts` (`?`, `*`, [ ], ^). Por ejemplo, si quisiéramos imprimir aquellos usuarios cuyo nombre empieza por la letra `A`, y terminan en `o`, con una longitud máxima de 8 caracteres escribiríamos algo parecido a esto:

```
open(USUARIOS,"</var/www/html/miweb/lista_usuarios.txt");
@lineas=<USUARIOS>;
foreach $linea (@lineas) {
if ($linea =~/A.{1,6}.o/) { #Entre la "A" y la "o" se permiten de 1 a 6 caract.
print "$linea"; } }
close(USUARIOS);
```

Si hubieramos querido rescatar todos los usuarios que empiezan por `A`, `B` y `L`, con una longitud mínima de 4 caracteres habríamos escrito,

```
$linea =~/[ABL].{3, }/
```

Los ejemplos anteriores, tan sólo comprueban si dentro de la línea especificada ($linea) aparece una palabra con el patrón dado. Si quisiéramos que la coincidencia se realice no sobre cualquier palabra sino en relación al comienzo de la línea deberemos hacer uso del carácter ^ (hay que tener cuidado porque si aparece entre [ ] se asocia a una negación). Si por el contrario la coincidencia quisiéramos buscarla comenzando por el final de la línea, haríamos uso de `$`.

```
$linea =~ /^[ABL].{3, }/
```

También se pueden establecer rangos. Por ejemplo si quisieramos imprimir aquellos empleados que sean jefes de departamento a través de su código de empleado, que debe empezar por `jf` seguido de 5 caracteres númericos y acabar en una letra, escribiríamos:

```
$codigo_empleado =~ /jf[^A-Za-z]5[^0-9]{1}/
```

Mediante | podemos establecer alternativas. Por ejemplo, si quisieramos conocer aquellos trabajadores que trabajan de `policia` o `bombero`:

```
$datos_trabajador =~ /policia|bombero/
```

- **s/ / / y s/ / /g**

Nos permite buscar un patrón determinado haciendo uso de las mismas reglas anteriores, y sustituirlo por otro. La sintaxis del filtro `s/ / /` es:

```
s/patrón_búsqueda/cadena_caracteres_de_sustitución/
```

Esto sólo es válido si lo que queremos es sustituir únicamente el primer patrón encontrado. Si queremos sustituir todos los que se encuentren deberemos hacer uso de `s/ / /g`.

```
$datos_usuario="Alonso:Molledo:72637221H:Alcañiz:Teruel";
$datos_usuario =~ /:/ /g
print $datos_usuario; #Resul.: "Alonso Molledo 72637221H Alcañiz Teruel"
```

Apéndice D

Pequeña descripción del lenguaje PHP

En este apéndice se explicarán de manera escueta la sintaxis y semántica del lenguaje PHP. No es el objetivo de este libro hacer un manual de PHP, lo que aquí se expone son las ideas necesarias para comenzar a realizar una pequeña página .php y entender el ejercicio propuesto en el capítulo de apache.

Definición de variables en PHP

- **$variable y $vector[*índice*]**

Permite la definición de una variable o lista no tipificada. No tipificada significa que permite almacenar indistintámente un valor numérico ó una cadena alfanumérica.
Ejemplos:

```
$numero=Sqrt($cantidad); //raiz cuadrada
$cadena="alumnos";
$lista=array("pedro",456,"julian",12,"arturo",65,...);
$lista[0]="pedro";
$lista[$indice]="andres";
$bidimensional[2] [$columna]=Sin($angulo); //seno
```

Variables encargadas de recoger el paso de valores desde un formulario

- **$HTTP_POST_VARS, $_POST, $HTTP_GET_VARS y $_GET**

La sintaxis de estas variables es:

```
$HTTP_POST_VARS['name campo formulario']
$_POST['name campo formulario']
$HTTP_GET_VARS['name campo formulario']
$_GET['name campo formulario']
```

Permiten trabajar directamente, con los valores introducidos en los campos de un formulario, identificándolos con el valor del atributo name especificado en éstos. Se utilizarán unos u otros dependiendo del method utilizado en el formulario, POST o GET.

Impresión de mensajes por pantalla

- **echo *mensaje***

Imprime por pantalla el mensaje indicado.
Ejemplo:

```
echo "Hola, me llamo ",$nombre;
echo "El resultado es ",Acos($cantidad)," grados";
echo "Bienvenido ",$HTTP_POST_VARS['nombre'];
```

Operadores lógicos, de comparación, incremento y concatenación

- **== (===), !=, <, >, <=, >=**

Su empleo es similar al de otros lenguajes de programación. (==) igual que, (!=) distinto que, (<) menor que, (>) mayor que, (<=) menor o igual que y (>=) mayor o igual que.

- **&&, | |, !**

Los operadores lógicos son: (&&) AND, (| |) OR, y (!) es la XOR o también llamada o-exclusiva.

- **Operadores ++, –, +=n, -=n y operador .**

Los operadores ++, --, +=n, -=n son llamados operadores de incremento. El operador "." es el operador de concatenación (su aplicación es similar a la comentada en el apéndice C para los operadores PERL).

Funciones con cadenas de caracteres

- **strlen(*cadena*)**

Devuelve la longitud de la cadena de caracteres.
Ejemplo:

```
$minusculas=strolower($contraseña);
```

- **strolower(*cadena*)**

Devuelve el resultado de convertir a minúsculas la cadena de caracteres.

- **strtoupper(*cadena*)**

Devuelve el resultado de convertir a mayúsculas la cadena de caracteres.

- **chop(*cadena*)**

Elimina los espacios en blanco al final de la cadena de caracteres indicada.

- **substr(*cadena,n,m*)**

Extrae `m` caracteres de la cadena a partir de la posición `n`.

- **ucwords(*cadena*)**

Convierte a mayúsculas la primera letra de cada palabra de la cadena.

- **ucfirst(*cadena*)**

Convierte a mayúsculas la primera letra, y pone en minúsculas las demás.

- **parse_str(*cadena*)**

Devuelve un conjunto de variables especificadas en la cadena. Se escribe la variable seguida de un igual, `=`; a continuación el valor de la variable (los espacios en blanco de este valor son señalados por un signo por `+`). Las variables y sus respectivos valores deben ir separadas por `&`.
Ejemplo:

```
parse_str(nombre=arturo+matin&correo=amartinr@educa.aragon.es);
```

Generaría dos variables con el valor,

```
$nombre="arturo martin"; $correo=amartinr@educa.aragon.es;
```

- **explode(*sep,cadena,n*)**

Devuelve una lista o vector de variables de hasta `n` posiciones, a partir de la cadena indicada, troceada a través del carácter delimitador indicado en `sep`.
Ejemplo:

```
$lista=explode(":","arturo:martin:romero:amartinr@educa.aragon.es",3);
```

Equivaldría a,

```
$lista[0]="arturo"; $lista[1]="martin"; $lista[2]="romero";
```

- **implode(*sep,$lista*)**

Operación inversa a la indicada en `explode` a partir de una lista de variables.
Ejemplo:

```
$cadena=implode(":",$lista);
```

Funciones de fecha y hora

- **mktime() y strftime()**

Para obtener la feha y la hora del sistema haremos uso de la función `mktime()`, y para mostrarlas en el formato deseado `strftime()`. Para establecer este formato se hace uso de una lista de variables entre las cuales destariamos: %A (nombre completo del día de la semana), %B (nombre completo del mes), %d (día del mes en número), %m (mes en número 1-12), %y (año en número 0-99), %Y (año en número con cuatro cifras), %H (hora en número, 0-23), %M (mínutos en número), %S (segundos en número), etc. Por ejemplo, si queremos mostrar la hora de la siguiente forma: `Hoy es` *`día_de_la_semana, dia-mes-año`*, escribiríamos:

```
$fecha=mktime();
echo strftime(Hoy es %A, %d-%B-%Y,$fecha);
```

- **setlocale()**

Por defecto la fecha se muestra en idioma inglés: monday, tuesday, ..., march, april, ... En el caso de que queramos que nos aparezca en español deberemos indicarlo mediante la función `setlocale()`.
Ejemplo:

```
$fecha=mktime();
setlocale(LC_TIME,"es_ES");
echo strftime("Hoy es %A %d-%B-%Y",$fecha);
```

Estructuras de control en PHP

- **if elseif else**

La estructura condicional es:

```
if (condición)
  { instrucciones; }
else/elseif (condición)
  { instrucciones; }
```

Un ejemplo de ejecución de código condicionado podría ser:

```
if ($HTTP_POST_VARS['password'] == "admin13")
  { echo "Bienvenido ",$HTTP_POST_VARS['nombre']; }
else { echo "Perdon ",$_POST['nombre'],"compruebe ..." };
```

- **switch case**

La estructura es:

```
switch ($variable) {
case valor1:
   instrucciones;
   break;
case valor2:
   instrucciones;
   break;
   ...
default:
   instrucciones;
}
```

La estructura `switch case` es muy recomendable en situaciones en las que aparecen estructuras `if` muy anidadas (multicondiciones), ya que el código se simplifica mediante su uso.
Ejemplo:

```
switch ($HTTP_POST_VARS['idioma']) {
case "castellano":
  ...
break;
case "ingles":
  ...
default:
  echo "Lo sentimos, no podemos informarle en el idioma indicado"; }
```

- **do while y while**

En los casos en que deseemos que sean ejecutadas un conjunto de instrucciones varias veces mientras se este cumpliendo una condición, haremos uso de `while`. Las posibles estructuras sintácticas son:

```
do
{ instrucciones; }
while (condicion)
```

y

```
while (condicion)
{ instrucciones; }
```

Si queremos que al menos se ejecute una vez sin tener en cuenta la condición haremos uso de `do while`, y en caso de que deseemos que se cumpla la condición para que se pueda ejecutar cualquiera de la instrucciones indicadas se hará uso de `while` .

- **foreach**

Cuando se desea llevar a cabo un conjunto de instrucciones sobre una lista haremos uso de `foreach ($lista as $valor)`.

```
foreach ($lista as $variable)
{ instrucciones; }
```

```
foreach ($lista as $indice => $valor)
{ instrucciones; }
```

El bucle se ejecutará tantas veces como valores tiene la lista, adquiriendo `$valor` en cada iteración el valor de `$lista`[`nº iteración`].

Si además de recuperar el valor de la lista, queremos tener constancia de la iteración y posición de la lista en la que nos encontramos, haremos uso de `foreach ($lista as $indice => $valor)`.
Ejemplo:

```
$nombres="arturo:juanjo:carol:andres:julian:celia";
$lista_nombres=explode(":",$nombres,1000);
echo "La lista de los usuarios permitidos es:<br>";
foreach ($lista_nombres as $posicion => $nombre)
{ echo $posicion,".- ",$nombre,".<br>"; }
```

- **for**

Sintaxis:

```
for (inicio;cond finalización;cadencia)
{ instrucciones; }
```

Con esta estructura de control ejecutaremos un conjunto de instrucciones un número de veces concreto.
Ejemplo:

```
for ($contador; $contador<=$cantidad; $contador++)
{ instrucciones; }
```

Creación de una biblioteca de funciones

- **function**

Con la finalidad de reutilizar el código y estructurarlo para su posterior mantenimiento, el lenguaje PHP, pueden definirse funciones propias.

```
function nombre_función (parámetros) {
Instrucciones que forman la función;
return(); }
```

Para llamar a la función definida tan sólo será necesario escribir su nombre junto con los parámetros a pasarle entre paréntesis. En el caso de querer devolver un valor tenemos la opción de utilizar la función `return()`.

Por ejemplo, es muy habitual mostrar en las páginas Web servidas por un servidor la fecha y la hora. Ante esta situación, tenemos la opción de incluir el conjunto de instrucciones que permiten visualizarlas en todas las páginas que lo requieran, o crear una función, y llamarla cuando se desee. En el caso de que tengamos dos formatos definidos dentro de la función elegiremos el deseado mediante el paso de un parámetro (1 ó 2):

```
function fechayhora ($tipoformato) {
$fecha=mktime();
setlocale(LC_TIME,"es_ES");
if ($tipoformato == 1)
{
return (strftime("Hoy es%A%d-%B-%Y",$fecha));
}
if ($tipoformato == 2)
{
return (strftime("Son las%H:%M:%S del%d-%B-%Y",$fecha));
} }
```

Para que esta función sea accesible desde diferentes páginas Web deberemos guardar la función en un fichero `.php` independiente (por ejemplo, `biblioteca.php`), e incluir su contenido en la página que requiera de su utilización mediante la función `include`.

```
include "biblioteca.php";
...
fechayhora("2");
...
```

Operaciones de manejo de ficheros

- **$lista=file(*ruta*);**

`file()` devuelve una lista compuesta por cada una de las líneas del fichero. No requiere su apertura previa (`fopen()`), ni su posterior cierre (`fclose()`). Es decir, el valor de cada uno de los elementos de la lista, son cada una de las líneas que compone el fichero indicado a través de su ruta absoluta o relativa dentro del sistema de ficheros:

```
$lineas=file("doc/usuarios.dat"); //file(/var/www/html/web/doc/usuarios.dat);
echo "Los usuarios registrados hasta el momento son:<br>";
foreach ($lineas as $indice => $usuario)
{ echo "Usuario nº",$indice,".- ",$usuario,".<br>"; }
```

- **$fichero=fopen(*ruta,modo*); y fclose($*fichero*);**

`fopen()` permite abrir el fichero especificado en *`ruta`* para su lectura (`r/r+`), escritura (`w/w+`), o añadir información a su contenido (`a/a+`). Una vez tratado el fichero, deberá ser cerrado mediante `fclose()`.

- **fgets($*fichero*,*cantidad*);**

En el caso de desear leer el contenido del fichero haremos uso de `fgets()`, indicando el manejador del fichero a leer y la cantidad de caracteres. Esta cantidad se puede ver interrumpida tanto por el final del fichero, como por un salto de línea.

- **fwrite($*fichero*,*cadena*);**

Para escribir en un fichero utilizamos este comando. Deberemos indicar la cadena de caracteres alfanuméricos que se desea insertar.

- **$pos_puntero=ftell($*fichero*);**

Teniendo en cuenta que todas las acciones de lectura y escritura son relativas a la posición de un puntero interno de lectura/escritura, a veces puede resultar interesante conocer dicha posición (bytes). De ello nos informa `ftell()`.

- **feof($*fichero*);**

`feof()` nos devuelve verdadero/falso (1/0) en función de si el puntero se encuentra o no al final del fichero.

- **fseek($*fichero*,$*posicion*);**

`fseek()` situa al puntero de lectura/escritura en la posición que indiquemos.

- **rewind($*fichero*);**

`rewind()` coloca el puntero al comienzo del fichero.

```
switch ($_POST['accion']) {
case "leer":
$fichero=fopen("informacion/puertos_teruel.dat","r");
echo "Los puertos de montaña de la provincia de Teruel son:<br>";
while (!feof($fichero)) {
$linea=fgets($fichero,800);
echo "$linea"; }
fclose($fichero);
break;
case "escribir":
$fichero=fopen("informacion/puertos_teruel.dat","a+");
fwrite($fichero,$_POST['nuevo_puerto']);
fwrite($fichero,"\n\r");
fclose($fichero); break; }
```

- **file_exists(*fichero*);**

`file_exists()` devuelve verdadero/falso (1/0) tras comprobar si existe el fichero indicado (ruta absoluta/relativa).

- **$tamaño=filesize(*fichero*);**

`filesize()` devuelve el tamaño (cantidad de bytes) del fichero indicado (ruta).

- **copy(*fichero1*,*fichero2*);**

`copy()` realiza una copia del fichero indicado (rutas absolutas/relativas).

- **rename(*nombre1*,*nombre2*);**

`rename()` renombra el fichero indicado.

- **unlink(*fichero*);**

`unlink()` borra el fichero indicado.

Índice alfabético