

Administración de servidores

Remo Suppi Boldrito

PID_00174427



Universitat Oberta
de Catalunya

www.uoc.edu

Índice

Introducción	5
Objetivos	6
1. Administración de servidores	7
1.1. Sistema de nombres de dominio (<i>Domain Name System, DNS</i>) ..	7
1.1.1. Servidor de nombres caché	8
1.1.2. <i>Forwarders</i>	11
1.1.3. Configuración de un dominio propio	11
1.2. NIS (YP)	13
1.2.1. ¿Cómo iniciar un cliente local de NIS en Debian?	14
1.2.2. ¿Qué recursos se deben especificar para utilizar el NIS?	15
1.2.3. ¿Cómo se debe ejecutar un <i>master NIS server</i> ?	16
1.2.4. ¿Cómo se debe configurar un servidor?	17
1.3. Servicios de conexión remota: telnet y ssh	18
1.3.1. Telnet y telnetd	18
1.3.2. SSH, <i>Secure shell</i>	19
1.4. Servicios de transferencia de ficheros: FTP	22
1.4.1. Cliente ftp (convencional)	22
1.4.2. Servidores FTP	23
1.5. Servicios de intercambio de información a nivel de usuario ...	25
1.5.1. El <i>Mail Transport Agent</i> (MTA)	25
1.6. <i>Internet Message Access Protocol</i> (IMAP)	26
1.6.1. Aspectos complementarios	27
1.7. Grupos de discusión	30
1.8. <i>World Wide Web</i> (httpd)	30
1.8.1. Configuración manual (mínima) de <code>httpd.conf</code>	31
1.8.2. Apache 2.2 + SSL + PHP + MySQL	32
1.9. Servidor de WebDav	35
1.10. Servicio de <i>proxy</i> : Squid	36
1.10.1. Squid como acelerador de http	36
1.10.2. Squid como <i>proxy-caching</i>	37
1.11. OpenLdap (Ldap)	38
1.11.1. Creación y mantenimiento de la base de datos	40
1.11.2. Instalación (básica) del servidor	41
1.12. Servicios de archivos (NFS, <i>Network File System</i>)	44
1.13. Servidor de wiki	46
1.13.1. Instalación rápida	46
1.13.2. Instalación de servidor	47
1.14. Gestión de copias de respaldo (<i>backups</i>)	50
Actividades	54
Bibliografía	54

Introducción

La interconexión entre máquinas y las comunicaciones de alta velocidad han permitido que los recursos que se utilicen no estén en el mismo sitio geográfico del usuario. UNIX (y por supuesto GNU/Linux) es probablemente el máximo exponente de esta filosofía, ya que desde su inicio ha fomentado el intercambio de recursos y la independencia de dispositivos. Esta filosofía se ha plasmado en algo común hoy en día como son los servicios. Un servicio es un recurso (que puede ser universal o no) y que permite, bajo ciertas condiciones, obtener información, compartir datos o simplemente procesar la información a distancia. Nuestro objetivo es analizar los servicios que permiten el funcionamiento de una red. Generalmente, dentro de esta red existirá una máquina (o varias, según las configuraciones) que hará posible el intercambio de información entre las demás. Estas máquinas se denominan servidores y contienen un conjunto de programas que permiten que la información esté centralizada y sea fácilmente accesible. Estos servicios permiten la reducción de costes y amplían la disponibilidad de la información, pero se debe tener en cuenta que un servicio centralizado presenta inconvenientes, ya que puede quedar fuera de servicio y dejar sin atención a todos los usuarios. En este módulo se verán los principales servicios que permiten que una máquina GNU/Linux juegue un papel muy importante en una infraestructura tecnológica, tanto en centralizar y distribuir datos como en ser punto de información, acceso o comunicación.

Servicios replicados

Una arquitectura de servidores debe tener los servicios replicados (*mirrors*) para solventar los inconvenientes que supone.

Objetivos

En los materiales didácticos de este módulo encontraréis los contenidos y las herramientas procedimentales para conseguir los objetivos siguientes:

- 1.** Presentar los aspectos más relevantes de los conceptos involucrados, tanto a nivel teórico como práctico, en la estructura de servidores/servicios en un sistema GNU/Linux.
- 2.** Analizar los conceptos relativos a servicios y servidores específicos de un sistema GNU/Linux.
- 3.** Experimentar con la configuración y adaptar la instalación de servicios a un entorno determinado.
- 4.** Analizar y participar en discusiones sobre las posibilidades actuales y futuras de nuevos servicios y los obstáculos que existen básicamente en aspectos de seguridad en los diferentes entornos de trabajo GNU/Linux (servidor, escritorio multimedia, escritorio ofimática, enrutador o *router*,...).

1. Administración de servidores

Los servicios se pueden clasificar en dos tipos: de vinculación ordenador-ordenador o de relación hombre-ordenador. En el primer caso, se trata de servicios requeridos por otros ordenadores, mientras que en el segundo, son servicios requeridos por los usuarios (aunque hay servicios que pueden actuar en ambas categorías). Dentro del primer tipo se encuentran servicios de nombres, como el *Domain Name System* (DNS), el servicio de información de usuarios (NIS-YP), el directorio de información LDAP o los servicios de almacenamiento intermedio (*proxies*). Dentro de la segunda categoría se contemplan servicios de conexión interactiva y ejecución remota (ssh, telnet), transferencia de ficheros (ftp), intercambio de información a nivel de usuario, como el correo electrónico (MTA, IMAP, POP), *news*, *World Wide Web*, *Wiki* y archivos (NFS). Para mostrar las posibilidades de GNU/Linux Debian-FC, se describirá cada uno de estos servicios con una configuración mínima y operativa, pero sin descuidar aspectos de seguridad y estabilidad.

1.1. Sistema de nombres de dominio (*Domain Name System, DNS*)

La funcionalidad del servicio de DNS es convertir nombres de máquinas (legibles y fáciles de recordar por los usuarios) en direcciones IP o viceversa.

A la consulta de cuál es la IP de *pirulo.remix.com*, el servidor responderá 192.168.0.1 (esta acción es conocida como *mapping*); del mismo modo, cuando se le proporcione la dirección IP, responderá con el nombre de la máquina (conocido como *reverse mapping*).

El *Domain Name System* (DNS) es una arquitectura arborescente que evita la duplicación de la información y facilita la búsqueda. Por ello, un único DNS no tiene sentido sino como parte del árbol. La aplicación que presta este servicio se llama *named*, se incluye en la mayoría de distribuciones de GNU/Linux (`/usr/sbin/named`) y forma parte de un paquete llamado *bind* (actualmente versión 9.x), coordinado por el ISC (Internet Software Consortium). El DNS es simplemente una base de datos, por lo cual es necesario que las personas que la modifiquen conozcan su estructura ya que, de lo contrario, el servicio quedará afectado. Como precaución debe tenerse especial cuidado en guardar las copias de los archivos para evitar cualquier interrupción en el servicio. El paquete sobre Debian se encuentra como *bind* y *bind.doc*. [5, 7, 11].

Las configuraciones son similares a las de FC pero necesitará instalar `bind`, `bind-utils` y `caching-nameserver`, que serán gestionadas por el `yum`, por ejemplo.

1.1.1. Servidor de nombres caché

En primer lugar, se configurará un servidor de DNS para resolver consultas que actúe como caché para las consultas de nombres (*resolver, caching only server*). Es decir, la primera vez consultará al servidor adecuado porque se parte de una base de datos sin información, pero las veces siguientes responderá el servidor de nombres caché, con la correspondiente disminución del tiempo de respuesta. Para configurar el servidor de nombres caché, se necesita el archivo `/etc/bind/named.conf` (en Debian), que tiene el siguiente formato (se han respetado los comentarios originales dentro del archivo, indicados por `//`):

```
options {
    directory "/var/cache/bind";
    // query-source address * port 53;
    // forwarders {
    // 0.0.0.0;
    //};
    auth-nxdomain no; # conform to RFC1035
};
// prime the server with knowledge of the root servers
zone "." {
    type hint;
    file "/etc/bind/db.root"; };
    // be authoritative for the localhost forward and reverse zones, and for
    // broadcast zones as per RFC 1912
zone "localhost" {
    type master;
    file "/etc/bind/db.local";
};
zone "127.in-addr.arpa" {
    type master;
    file "/etc/bind/db.127";
};
zone "0.in-addr.arpa" {
    type master;
    file "/etc/bind/db.0";
};
zone "255.in-addr.arpa" {
    type master;
    file "/etc/bind/db.255";
};
// add entries for other zones below here
```

La sentencia `directory` indica dónde se encontrarán los archivos de configuración restantes (`/var/cache/bind` en nuestro caso).

El archivo `/etc/bind/db.root` contendrá algo similar a lo siguiente (se muestran algunas líneas donde los comentarios son la líneas que comienzan por `;`; se debe tener cuidado además con los puntos (`.`) al inicio de algunas líneas ya que forman parte del formato del archivo [este archivo se puede obtener directamente de Internet actualizado]):


```

...
; formerly NS.INTERNIC.NET
;
. 3600000 IN NS A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000 A 198.41.0.4
;
; formerly NS1.ISI.EDU
;
. 3600000 NS B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000 A 128.9.0.107
;
; formerly C.PSI.NET
;
. 3600000 NS C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET. 3600000 A 192.33.4.12
;
...

```

Este archivo describe los *root name servers* en el mundo. Estos servidores cambian, por lo que el archivo se debe actualizar periódicamente. Las siguientes secciones son las zonas; las zonas `localhost` y `127.in-addr.arpa`, que se vinculan a los ficheros al archivo `etc/bind/db.local` y `etc/bind/db.127`, se refieren a la resolución directa e inversa para la interfaz local. Las zonas siguientes son para las zonas de difusión (según RFC 1912) y al final se deberían agregar las propias. Por ejemplo, el archivo `db.local` podría ser (“;” significa *comentario*):

```

; BIND reverse data file for local loopback interface
$TTL 604800
@ IN SOA ns.remix.bogus. root.remix.bogus. (
    1      ; Serial
    604800 ; Refresh
    86400  ; Retry
    2419200 ; Expire
    604800) ; Negative Cache TTL
@ IN NS   ns.remix.bogus.
1.0.0 IN PTR localhost.

```

Explicaremos su utilización más adelante. Lo siguiente es poner como *name server* en el `/etc/resolv.conf`:

```

search subdominio.su-dominio.dominio
su-dominio.dominio
# por ejemplo, search remix.bogus bogus
nameserver 127.0.0.1

```

donde se deberán reemplazar los `subdominio.su-dominio.dominio` por los valores adecuados. La línea `search` indica qué dominios se buscarán para cualquier *host* que se quiera conectar* y `nameserver` especifica la dirección de su *nameserver* (en este caso su propia máquina, que es donde se ejecutará el `named`). El `search` tiene este comportamiento: si un cliente busca la máquina `pirulo`, primero buscará `pirulo.subdominio.su-dominio.dominio`, lue-

*Es posible sustituir `search` por `domain`, aunque tienen comportamientos diferentes.

go `pirulo.su-dominio.dominio` y, finalmente, `pirulo`. Esto implica tiempo de búsqueda; ahora bien, si se tiene la seguridad de que `pirulo` está en `subdominio.su-dominio.dominio`, no es necesario poner los restantes.

El paso siguiente es poner en marcha el `named` y mirar los resultados de la ejecución. Para poner en marcha el *daemon*, podéis hacer directamente con el *script* de inicialización `/etc/init.d/bind9 start` (verificad el nombre del *script* en el directorio `/etc/init.d`; en caso de que el `named` ya se esté ejecutando, entonces haced `/etc/init.d/bind9 reload`) o, si no, haced también `/usr/sbin/named`. Mirando el registro del sistema en el archivo `/var/log/daemon.log` veremos algo como lo siguiente:

```
Sep 1 20:42:28 remolix named[165]: starting BIND 9.2.1
Sep 1 20:42:28 remolix named[165]: using 1 CPU
Sep 1 20:42:28 remolix named[167]: loading configuration from "/etc/bind/named.conf"
```

Aquí se indica el arranque del servidor y los mensajes de errores (si los hay), los cuales se deberán corregir y se deberá volver a empezar. Ahora se puede verificar la configuración con comandos como `nslookup` (original y fácil pero obsoleto según algunos autores), `host` o `dig` (recomendado). La salida de `dig -x 127.0.0.1` será algo como:

```
# dig -x 127.0.0.1
;; <<>> DiG 9.2.1 <<>> -x 127.0.0.1
;; global options: printcmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 31245
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
;; QUESTION SECTION: ;1.0.0.127.in-addr.arpa. IN PTR
;; ANSWER SECTION: 1.0.0.127.in-addr.arpa. 604800 IN PTR localhost.
;; AUTHORITY SECTION: 127.in-addr.arpa. 604800 IN NS ns.remix.bogus.
;; Query time: 1 msec
;; SERVER: 127.0.0.1 \#53(127.0.0.1)
;; WHEN: Mon Sep 1 22:23:35 2010
;; MSG SIZE rcvd: 91
```

donde se puede ver que la consulta ha tardado 1 milisegundo. Si se dispone de conexión a Internet, se podría buscar alguna máquina dentro de vuestro dominio y ver el comportamiento de vuestro servidor. En *BIND9* existe el `lwresd` (*lightweight resolver daemon*), que es el *daemon* que provee servicios de nombres a clientes que utilizan la biblioteca de *BIND9 lightweight resolver*. Es esencialmente un servidor caché (como el que se ha configurado) el que realiza las consultas utilizando el *BIND9 lightweight resolver protocol* en lugar del protocolo DNS. Este servidor escucha por la interfaz 127.0.0.1 (por lo cual, solo atiende a procesos de la máquina local) en UDP y el puerto 921. Las consultas de los clientes se decodifican y se resuelven utilizando el protocolo DNS. Cuando se obtienen las respuestas, el `lwresd` las codifica en el formato *lightweight* y las retorna al cliente que las ha solicitado.

Por último, como ya se ha mencionado, el núcleo utiliza diversas fuentes de información para la red, que se obtienen desde `/etc/nsswitch.conf`. Este archivo indica desde donde obtener la fuente de información y para el caso de los nombres de máquinas e IP hay una sección como:

```
hosts: files dns
```

Esta línea (si no existe se debe agregar) indica que quien necesite un nombre de una máquina o una IP consulte primero en `/etc/hosts` y luego en DNS, de acuerdo a los dominios indicados en `/etc/resolv.conf`.

1.1.2. Forwarders

En redes con una considerable carga, es posible equilibrar el tráfico utilizando la sección de *forwarders*. Si vuestro proveedor de red (ISP) tiene uno o más *nameservers* estables, es recomendable utilizarlos para descongestionar las consultas sobre su servidor. Para ello, debe quitarse el comentario (`//`) de cada línea de la sección *forwarders* del archivo `/etc/bind/named.conf` y reemplazar el `0.0.0.0` con las IP de los *nameservers* de su ISP. Esta configuración es recomendable cuando la conexión es lenta.

1.1.3. Configuración de un dominio propio

DNS posee una estructura en árbol y el origen es conocido como `."` (ver `/etc/bind/db.root`). Bajo el `."` existen los TLD (*Top Level Domains*) como **org**, **com**, **edu**, **net**, etc. Cuando se busca en un servidor, si este no conoce la respuesta, se buscará recursivamente en el árbol hasta encontrarla. Cada `."` en una dirección (por ejemplo, `pirulo.remix.com`) indica una rama del árbol de DNS diferente y un ámbito de consulta (o de responsabilidad) diferente que se irá recorriendo en forma recursiva de izquierda a derecha.

Otro aspecto importante, además del dominio, es el `in-addr.arpa` (*inverse mapping*), el cual también está anidado como los dominios y sirve para obtener nombres cuando se consulta por la dirección IP. En este caso, las direcciones se escriben al revés, en concordancia con el dominio. Si `pirulo.remix.com` es la `192.168.0.1`, entonces se escribirá como `1.0.168.192`, en concordancia con `pirulo.remix.com`. A continuación, configuraremos el dominio propio `remix.bogus` en el archivo `/etc/bind/db.127` [11]:

```
; BIND reverse data file for local loopback interface
$TTL 604800
@ IN SOA ns.remix.bogus. root.remix.bogus. (
    1          ; Serial
```

```

604800      ; Refresh
86400      ; Retry
2419200    ; Expire
604800 )    ; Negative Cache TTL
@ IN NS ns.remix.bogus.
1.0.0 IN PTR localhost.

```

Se debe tener en cuenta el “.” al final de los nombres de dominio. El origen de la jerarquía de una zona está especificada por la identificación de la zona, en nuestro caso `127.in-addr.arpa`. Este archivo (`db.127`) contiene 3 registros: `SOA`, `NS`, `PTR`. El `SOA` (*Start of Authority*) debe estar en todos los archivos de zona al inicio, después de `TTL`, y el símbolo `@` significa el origen del dominio; `NS`, el servidor de nombres para el dominio, y `PTR` (*Domain Name Pointer*), que es el *host* 1 en la subred (`127.0.0.1`) y se denomina *local host*. Este es el archivo serie 1 y el responsable del mismo es `root@remix.bogus` (último campo de la línea `SOA`). Ahora se podría reiniciar el `named` de la forma antes indicada y con el `dig -x 127.0.0.1`, ver su funcionamiento (que sería idéntico al mostrado anteriormente). A continuación, habrá que añadir una nueva zona en el `named.conf`:

```

zone "remix.bogus" {
    type master;
    notify no;
    file "/etc/bind/remix.bogus";
};

```

Se debe recordar que en el `named.conf` los dominios van sin el “.” final. En el archivo `remix.bogus` se pondrán los *hosts* de los cuales seremos responsables:

```

; Zone file for remix.bogus
$TTL 604800
@ IN SOA ns.remix.bogus. root.remix.bogus. (
    199802151      ; serial, todays date + todays serial
    604800      ; Refresh
    86400      ; Retry
    2419200    ; Expire
    604800 )    ; Negative Cache TTL
@ NS ns      ; Inet Address of name server
    MX 10 mail.remix.bogus.      ; Primary Mail Exchanger
localhost    A      127.0.0.1
ns           A      192.168.1.2
mail        A      192.168.1.4
    TXT "Mail Server"
ftp         A      192.168.1.5
    MX 10 mail
www         CNAME   ftp

```

Aquí aparece un nuevo registro `MX` que es el *Mail eXchanger*. Es el lugar donde se enviarán los correos electrónicos que lleguen, `alguien@remix.bogus`, y será a `mail.remix.bogus`. (El número indica la prioridad si tenemos más de un `MX`.) Tened presente que el “.” siempre es necesario en los archivos de zona al final del dominio (si no se ponen, el sistema agrega el dominio `SOA` al final, lo cual transformaría, por ejemplo, `mail.remix.bogus` en `mail.remix.bogus.remix.bogus`, que es incorrecto). `CNAME` (*canonical name*) es la forma de dar a una máquina uno o varios alias. A partir de ahora se

estaría en condiciones (después de `/etc/init.d/bind9 reload`) de probar, por ejemplo, `dig www.remix.bogus`.

El último paso es configurar la zona inversa, es decir, para que pueda convertir direcciones IP en nombres, por ejemplo, agregando una nueva zona:

```
zone "1.168.192.in-addr.arpa" {
    type master;
    notify no;
    file "/etc/bind/192.168.1";
};
```

Y el archivo `/etc/bind/192.168.1` similar al anterior:

```
$TTL 604800
@ IN SOA ns.remix.bogus. root.remix.bogus. (
    199802151      ; serial, todays date + todays serial
    604800        ; Refresh
    86400         ; Retry
    2419200      ; Expire
    604800 )      ; Negative Cache TTL
@ NS      ns.remix.bogus.
2 PTR    ns.remix.bogus
4 PTR    mail.remix.bogus
5 PTR    ftp.remix.bogus
```

Este nuevamente se podría probar con `dig -x 192.168.1.4`. Se debe tener en cuenta que estos ejemplos son sobre IP privadas, es decir, no IP de Internet. Otra cuestión importante es no olvidar el `notify no`, ya que de lo contrario, nuestros experimentos con DNS se propagarán a los servidores del árbol de DNS (incluso modificando los DNS de nuestro proveedor o institución). Solo se debe modificar cuando estamos seguros de que funciona y queremos propagar los cambios*. Una vez creado un servidor maestro (*master server*), debe crearse un servidor esclavo (*slave server*) por seguridad, que es idéntico al maestro, excepto que la zona en lugar de `type master` deberá tener `slave` y la IP del maestro. Por ejemplo:

```
zone "remix.bogu" {
    type slave;
    notify no;
    masters {192.168.1.2;}
};
```

1.2. NIS (YP)

Con el fin de facilitar la administración y dar comodidad al usuario en redes de diferentes tamaños que ejecutan GNU/Linux (o Sun o algún otro sistema operativo con soporte para este servicio), se ejecutan servicios de *Network Information Service*, NIS (o *Yellow Pages*, YP, en la definición original de Sun). GNU/Linux puede dar apoyo como cliente/servidor de NIS y puede actuar como cliente (versión "beta") de NIS+, que es una versión más segura y optimi-

*Para ver un ejemplo real, consultad DNS-HOWTO en <http://tldp.org/HOWTO/DNS-HOWTO-7.html>.

zada de NIS. La información que se puede distribuir en NIS es: usuarios (*login names*), contraseñas (*passwords*, */etc/passwd*), directorios de usuario (*home directories*) e información de grupos (*group information*, */etc/group*), lo cual presenta la ventaja de que, desde cualquier máquina cliente o desde el mismo servidor, el usuario se podrá conectar con la misma cuenta y contraseña y al mismo directorio (aunque el directorio deberá ser montado anteriormente sobre todas las máquinas cliente por NFS o mediante el servicio de automount). [17, 10]

La arquitectura NIS es del tipo cliente-servidor, es decir, existe un servidor que dispondrá de todas las bases de datos y unos clientes que consultarán estos datos en forma transparente para el usuario. Por ello, se debe pensar en la posibilidad de configurar servidores “de refuerzo” (llamados secundarios) para que los usuarios no queden bloqueados ante la caída del servidor principal. Es por ello que la arquitectura se denomina realmente de múltiples servidores (*master+mirrors-clients*).

1.2.1. ¿Cómo iniciar un cliente local de NIS en Debian?

Un cliente local es el que anexa el ordenador a un dominio NIS ya existente: primero se debe verificar que se tienen instalados los paquetes `netbase` (red básica TCP/IP) y `portmap` (servidor que convierte números RPC *–Remote Procedure Call–* en puertos DARPA) y `nis` (específico). Es necesario verificar la instalación de los dos primeros paquetes para todos los programas que ejecutan RPC, incluyendo NFS y NIS. Se recomienda usar el comando `apt-get` (o también `kpackage`) y se puede verificar si está instalado con `apt-cache pkgnames` en modo texto. El procedimiento de instalación del paquete NIS solicitará un dominio (`NIS domainname`). Este es un nombre que describirá el conjunto de máquinas que utilizarán el NIS (no es un nombre de *host*). Hay que tener en cuenta que `NISpirulo` es diferente de `Nispirulo` como nombre de dominio. Para configurarlo, se puede utilizar el comando `nisdomainname`, dominio que se almacenará en `/proc/sys/kernel/domainname`. En primer lugar, se debe iniciar el servicio `portmap` con:

```
/etc/init.d/portmap start
```

Se puede comprobar si estos servicios están activos con `rpcinfo -p`. Si el servidor NIS no es local, se deberá utilizar el comando `yplibind`. El comando `yplibind` se utiliza para encontrar un servidor para el dominio especificado, ya sea mediante `broadcast` (no aconsejado por inseguro) o buscando el servidor indicado en el archivo de configuración `/etc/yp.conf` (recomendable).

Sintaxis del archivo `/etc/yp.conf`

`domain nisdomain server hostname`: indica que se utiliza el *hostname* para el dominio *nisdomain*. Se podría tener más de una entrada de este tipo para un único dominio.

`domain nisdomain broadcast`: indica que se utiliza *broadcast* sobre la red local para descubrir un servidor de dominio *nisdomain*.

`ypserver hostname`: indica que se utiliza *hostname* como servidor. Es recomendable utilizar esta línea (*ypserver*) donde se deberá introducir la dirección IP del servidor NIS. Si se indica el nombre, asegúrese de que se puede encontrar la IP por DNS o que la misma figura en el archivo `/etc/hosts` ya que, de otro modo, el cliente se bloqueará.

Para iniciar el servicio se debe ejecutar:

```
/etc/init.d/nis stop      para detenerlo
/etc/init.d/nis start     para iniciarlo
```

A partir de este momento, el cliente NIS estará funcionando. Ello se puede confirmar con `rpcinfo -u localhost ypbind`, que mostrará las dos versiones del protocolo activo o se puede utilizar el comando `ypcat mapname` (por ejemplo, `ypcat passwd`, que mostrará los usuarios NIS definidos en el servidor) donde las relaciones entre *mapnames* y las tablas de la base de datos NIS están definidas en `/var/yp/nicknames`.

1.2.2. ¿Qué recursos se deben especificar para utilizar el NIS?

Consideraremos que tenemos instalada una de las últimas distribuciones de Debian (cualquiera a partir de Sarge) que soporta la Libc6 (igualmente para FC4 o superior) y se quiere que los usuarios de una máquina cliente puedan acceder a la información del servidor. En este caso, se debe orientar la consulta del *login* a las bases de datos adecuadas con los pasos siguientes:

1) Verificar el fichero `/etc/nsswitch.conf` y asegurarse de que las entradas `passwd`, `group`, `shadow` y `netgroup` son similares a:

```
passwd: compat
group: compat
shadow: compat ...
netgroup: nis
```

Consultad la sintaxis de este archivo en `man nsswitch.conf`.

2) Agregar la siguiente línea a las máquinas clientes NIS al final del fichero `/etc/passwd` (indicará que si el usuario no es local, se lo preguntará al servidor de NIS):

```
+::: (un "+" y seis ":")
```

Debe tenerse en cuenta que en el `/etc/passwd` se puede utilizar el `+` y el `?` delante de cada nombre de usuario en el `/etc/passwd` para incluir o excluir el *login* de estos usuarios (*override*). Si se están utilizando contraseñas con *shadow* (opción más segura, puesto que no permite que un usuario normal pueda ver la contraseña encriptada de otros usuarios) se deberá incluir la siguiente línea al final del archivo `/etc/shadow`:

```
+::: (un "+" y ocho ":")
```

3) Se debe añadir también la siguiente línea al final de `/etc/group`:

```
+::: (un "+" y tres ":")
```

4) Las búsquedas de *hosts* (*hosts lookups*) se realizarán mediante DNS (y no por NIS), por lo cual, para aplicaciones Libc6 en el fichero `/etc/nsswitch.conf` habrá que cambiar la entrada `hosts` por la siguiente línea: `hosts: files dns`. O, si se prefiere hacer por NIS, `hosts: files nis`. Para aplicaciones Libc5, se deberá modificar el fichero `host.conf` poniendo `order hosts,dns` o `order hosts,nis` según desee.

Con esta configuración se podrá realizar una conexión local (sobre el cliente NIS) a un usuario que no esté definido en el fichero `/etc/passwd`, es decir, un usuario definido en otra máquina (*ypserver*). Por ejemplo, se podría hacer `ssh -l user localhost`, donde `user` es un usuario definido en *ypserver*.

1.2.3. ¿Cómo se debe ejecutar un *master NIS server*?

Consideramos que la máquina tiene instalado el paquete `nis` y el `portmap` (este último en funcionamiento) y que las bases de datos del NIS están creadas (ved el subapartado 1.2.4). Habrá que asegurarse de que en el `/etc/hosts` se encuentren todas las máquinas que formarán parte del dominio en el formato FQDN (*Fully Qualified Domain Name*), que es donde se indican la IP, el nombre con dominio y el nombre sin dominio de cada máquina (por ejemplo, `192.168.0.1 pirulo.remix.com pirulo`). Esto es necesario solo en el servidor, ya que el NIS no utiliza DNS. Además, existe en el archivo `/etc/defaultdomain` con el nombre del dominio escogido. No utilizéis vuestro dominio DNS para no poner en riesgo la seguridad, excepto si configuráis adecuadamente los archivos `/etc/ypserv.securenets` (que indican con un par `netmask/network` desde qué sitio se podrán conectar los clientes) y `/etc/ypserv.conf` (que realiza un control más detallado porque indica qué *hosts* pueden acceder a qué mapas; por ejemplo: `passwd.byname` o `shadow.byname`).

Verificad que existe `NISSERVER = master` en `/etc/default/nis`. Se puede agregar el número de red local al archivo `/etc/ypserv.securenets` por motivos de seguridad. Inicial el servidor ejecutando primero el comando `/etc/init.d/nis stop` y luego `/etc/init.d/nis start`. Esta sentencia iniciará el servidor (`ypserv`) y el *password daemon* (`yppasswdd`), cuya activación se podrá consultar con `ypwich -d domain`.

1.2.4. ¿Cómo se debe configurar un servidor?

La configuración del servidor se hace con el comando `/usr/lib/yp/ypinit -m`; sin embargo, es necesario verificar que existe el archivo `/etc/networks`, que es imprescindible para este *script*. Si este archivo no existe, cread uno vacío con `touch/etc/networks`. También se puede ejecutar el cliente `ypbind` sobre el servidor; así, todos los usuarios entran por NIS, como se indicó anteriormente, modificando el fichero `/etc/passwd` donde todas las entradas normales antes de la línea `+: :: ::` serán ignoradas por el NIS (solo podrán acceder localmente), mientras que las posteriores podrán acceder por el NIS desde cualquier cliente [17]. Considerad que a partir de este momento los comandos para cambiar la contraseña o la información de los usuarios, como `passwd`, `chfn` o `adduser`, no son válidos. En su lugar, se deberán utilizar comandos tales como `yppasswd`, `ypchsh` y `ypchfn`. Si se cambian los usuarios o se modifican los archivos mencionados, habrá que reconstruir las tablas de NIS ejecutando el comando `make` en el directorio `/var/yp` para actualizar las tablas. Tened en cuenta que Libc5 no soporta `shadow passwd` (contraseñas en el archivo `/etc/shadow`), por lo cual no se debe utilizar *shadow* con NIS si se tienen aplicaciones con Libc5. No habrá ningún problema si se tiene Libc6, que acepta NIS con soporte *shadow*. La configuración de un servidor esclavo es similar a la del maestro, excepto si `NISSERVER = slave` en `/etc/default/nis`. Sobre el maestro se debe indicar que distribuya las tablas automáticamente a los esclavos, poniendo `NOPUSH = "false"` en el archivo `/var/yp/Makefile`. Ahora se debe indicar al maestro quién es su esclavo por medio de la ejecución de:

```
/usr/lib/yp/ypinit -m
```

e introduciendo los nombres de los servidores esclavos. Esto reconstruirá los mapas pero no enviará los archivos a los esclavos. Para ello, sobre el esclavo, ejecutad:

```
/etc/init.d/nis stop
/etc/init.d/nis start
/usr/lib/yp/ypinit -s nombre_master_server
```

Así, el esclavo cargará las tablas desde el maestro. También se podría poner en el directorio `/etc/cron.d` el archivo NIS con un contenido similar a (recordad hacer un `chmod 755 /etc/cron.d/nis`):

```
20 * * * * root /usr/lib/yp/ypxfr_1perhour >/dev/null 2>&1
40 6 * * * root /usr/lib/yp/ypxfr_1perday >/dev/null 2>&1
55 6,18 * * * root /usr/lib/yp/ypxfr_2perday >/dev/null 2>&1
```

Con ello, se garantizará que todos los cambios del maestro se transfieran al servidor NIS esclavo.

Actualización de las tablas NIS

Es recomendable que después de usar `adduser` para agregar un nuevo usuario sobre el servidor, ejecutad `make -C /var/yp` para actualizar las tablas NIS (y siempre que se cambie alguna característica del usuario, por ejemplo la palabra clave con el comando `passwd` que solo cambiará la contraseña local y no el de NIS). Para probar que el sistema está funcionando y que el usuario dado de alta está en el NIS podéis hacer `ypmatch userid passwd`, donde `userid` es el usuario dado de alta con `adduser` antes y después de haber hecho el `make`. Para verificar el funcionamiento del sistema NIS podéis utilizar el *script* de <http://tldp.org/HOWTO/NIS-HOWTO/verification.html> que permite una verificación más detallada del NIS.

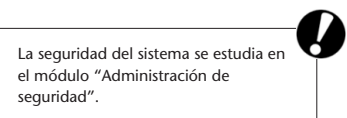
1.3. Servicios de conexión remota: telnet y ssh

1.3.1. Telnet y telnetd


Telnet es un comando (cliente) utilizado para comunicarse interactivamente con otro *host* que ejecuta el *daemon* `telnetd`. El comando `telnet` se puede ejecutar como `telnet host` o interactivamente como `telnet`, el cual pondrá el *prompt* “telnet>” y luego, por ejemplo, `open host`. Una vez establecida la comunicación, se deberá introducir el usuario y la contraseña bajo la cual se desea conectar al sistema remoto. Se dispone de diversos comandos (en modo interactivo) como `open`, `logout` y `mode` (se deben definir las características de visualización), `close`, `encrypt`, `quit`, `set` y `unset` o se pueden ejecutar comandos externos con “!”. Se puede utilizar el archivo `/etc/telnetrc` para definiciones por defecto o `.telnetrc` para definiciones de un usuario particular (deberá estar en el directorio *home* del usuario).

El *daemon* `telnetd` es el servidor de protocolo telnet para la conexión interactiva. Generalmente, es el *daemon* `inetd` que pone en marcha `telnetd`; se recomienda incluir un *wrapper* `tcpd` (que utiliza las reglas de acceso en `host.allow` y `host.deny`) en la llamada al `telnetd` dentro del archivo `/etc/inetd.conf`. Para incrementar la seguridad del sistema se debería, por ejemplo, incluir una línea como:

```
telnet stream tcp nowait telnetd.telenetd /usr/sbin/tcpd /usr/bin/in.telnetd)
```



En algunas distribuciones (Debian 3.0 o superiores), la funcionalidad de `inetd` se puede reemplazar por la de `xinetd`, el cual requiere la configuración del archivo `/etc/xinetd.conf`. Si se quiere poner en marcha `inetd` a modo de pruebas, se puede usar la sentencia `/etc/init.d/inetd.real start`. Si el archivo `/etc/uissue.net` está presente, el `telnetd` mostrará su contenido al inicio de la sesión. También se puede usar `/etc/security/access.conf` para habilitar y deshabilitar *logins* de usuario, *host* o grupos de usuarios, según se conecten.



El archivo `/etc/xinetd.conf` se estudia en el módulo "Administración de seguridad".

Se debe recordar que, si bien el par `telnet-telnetd` puede funcionar en modo *encrypt* en las últimas versiones (transferencia de datos encriptados, que deben estar compilados con la opción correspondiente), es un comando que ha quedado en el olvido por su falta de seguridad (transmite el texto en claro sobre la red, lo que permite la visualización del contenido de la comunicación desde otra máquina, por ejemplo, con el comando `tcpdump`); no obstante, puede utilizarse en redes seguras o situaciones controladas.

Si no está instalado, se puede utilizar (Debian) `apt-get install telnetd` y después verificar que se ha dado de alta, bien en `/etc/inetd.conf`, bien en `/etc/xinetd.conf` (o en el directorio en que estén definidos los archivos; por ejemplo, `/etc/xinetd.d` según se indique en el archivo anterior con la sentencia `include /etc/xinetd.d`). El `xinetd.conf` o el archivo `/etc/xinetd.d/telnetd` deberán incluir una sección como*:

```
service telnet {
  disable = no
  flags = REUSE
  socket_type = stream
  wait = no
  user = root
  server = /usr/sbin/in.telnetd
  log_on_failure += USERID
}
```

*Cualquier modificación en `xinetd.conf` deberá arrancar nuevamente el servicio con `service xinetd restart`.

SSL telnet(d)

Se recomienda que en lugar de usar `telnetd` se utilice `SSL telnet(d)`, que reemplaza a `telnet(d)` utilizando encriptación y autenticación por SSL, o que se utilice SSH. El `SSLTelnet(d)` puede funcionar con el `telnet(d)` normal en ambas direcciones ya que al inicio de la comunicación verifica si el otro lado (*peer*) soporta SSL, de lo contrario, continúa con el protocolo `telnet` normal. Las ventajas con respecto al `telnet(d)` son que sus contraseñas y datos no circularán por la red en modo de texto plano y nadie que utilice el comando antes mencionado (`tcpdump`) podrá ver el contenido de la comunicación. También `SSL telnet` se puede utilizar para conectarse, por ejemplo, a un servidor web seguro (por ejemplo `https://servidor.web.org`) simplemente haciendo `telnet servidor.web.org 443`.

1.3.2. SSH, *Secure shell*

Un cambio aconsejable hoy en día es utilizar `ssh` en lugar de `telnet`, `rlogin` o `rsh`. Estos tres últimos comandos son inseguros (excepto `SSLTelnet`) por varias razones: la más importante es que todo lo que se transmite por la red, in-

cluidos nombres de usuarios y contraseñas, es en texto plano (aunque existen versiones de telnet-telnetd encriptados, debe coincidir que ambos los sean), de modo que cualquiera que tenga acceso a esa red o a algún segmento de la misma puede obtener toda esta información y luego suplantar la identidad del usuario. La segunda es que estos puertos (telnet, rsh, etc.) son el primer lugar donde un *cracker* intentará conectarse. El protocolo `ssh` (en su versión OpenSSH) provee una conexión encriptada y comprimida mucho más segura que, por ejemplo, `telnet` (es recomendable utilizar la versión 2.0 o versiones superiores del protocolo). Todas las distribuciones actuales incorporan el cliente `ssh` y el servidor `sshd` por defecto.

ssh

Para ejecutar el comando, haced:

```
ssh -l login name host o ssh user@hostname
```

A través de SSH se pueden encapsular otras conexiones como X11 o cualquier otra TCP/IP. Si se omite el parámetro `-l`, el usuario se conectará con el mismo usuario local y en ambos casos el servidor solicitará la contraseña para validar la identidad del usuario. SSH soporta diferentes modos de autenticación (ved `man ssh`) basados en algoritmo RSA y clave pública.

Usando el comando `ssh-keygen -t rsa|dsa`, se pueden crear las claves de identificación de usuario. El comando crea en el directorio del `.ssh` del usuario los ficheros* `id_rsa` y `id_rsa.pub`, las claves privada y pública, respectivamente. El usuario podría copiar la pública (`id_rsa.pub`) en el archivo `$HOME/.ssh/authorized_keys` del directorio `.ssh` del usuario de la máquina remota. Este archivo podrá contener tantas claves públicas como sitios desde donde se quiera conectar a esta máquina de forma remota. La sintaxis es de una clave por línea y su funcionamiento es equivalente al archivo `.rhosts` (aunque las líneas tendrán un tamaño considerable). Después de haber introducido las claves públicas del usuario-máquina en este archivo, este usuario se podrá conectar sin contraseña desde esa máquina.

*Por ejemplo, para el algoritmo de encriptación RSA.

En forma normal (si no se han creado las claves), se le preguntará al usuario una contraseña, pero como la comunicación será siempre encriptada, nunca será accesible a otros usuarios que puedan escuchar sobre la red. Para mayor información, consultad `man ssh`. Para ejecutar remotamente un comando, simplemente haced:

```
ssh -l login name host_comando_remoto  
Por ejemplo: ssh -l user localhost ls -al
```

sshd

El `sshd` es el servidor (*daemon*) para el `ssh` (si no están instalados, se puede hacer con `apt-get install ssh`, que instala el servidor y el cliente). Juntos reemplazan a `rlogin`, `telnet`, `rsh` y proveen una comunicación segura y encriptada en dos *hosts* inseguros de la red. Este se arranca generalmente a través de los archivos de inicialización (`/etc/init.d` o `/etc/rc`) y espera conexiones de los clientes. El `sshd` de la mayoría de las distribuciones actuales soporta las versiones 1 y 2 (o 3) del protocolo SSH. Cuando se instala el paquete, se crea una clave RSA específica del *host* y cuando el *daemon* se inicia, crea otra, la RSA para la sesión, que no se almacena en el disco y que cambia cada hora. Cuando un cliente inicia la comunicación, genera un número aleatorio de 256 bits que está encriptado con las dos claves del servidor y enviado. Este número se utilizará durante la comunicación como clave de sesión para encriptar la comunicación que se realizará a través de un algoritmo de encriptación estándar. El usuario puede seleccionar cualquiera de los algoritmos disponibles ofrecidos por el servidor. Existen algunas diferencias (más seguro) cuando se utiliza la versión 2 (o 3) del protocolo. A partir de este momento, se inician algunos de los métodos de autenticación de usuario descritos en el cliente o se le solicita la contraseña, pero siempre con la comunicación encriptada. Para mayor información, consultad `man sshd`.

Túnel sobre SSH

Muchas veces tenemos acceso a un servidor `sshd` pero por cuestiones de seguridad no podemos acceder a otros servicios que no están encriptados (por ejemplo un servicio de consulta de mail POP3 o un servidor de ventanas X11) o simplemente se quiere conectar a un servicio al cual solo se tiene acceso desde el entorno de la empresa. Para ello, es posible establecer un túnel encriptado entre la máquina cliente (por ejemplo con Windows y un cliente `ssh` llamado `putty` de software libre) y el servidor con `sshd`. En este caso, al vincular el túnel con el servicio, el servicio verá la petición como si viniera de la misma máquina. Por ejemplo, si queremos establecer una conexión para POP3 sobre el puerto 110 de la máquina remota (y que también tiene un servidor `sshd`) haremos:

```
ssh -C -L 1100:localhost:110 usuario-id@host
```

Este comando pedirá la contraseña para el `usuario-id` sobre *host* y una vez conectado se habrá creado el túnel. Cada paquete que se envíe en la máquina local sobre el puerto 1100 se enviará a la máquina remota *localhost* sobre el puerto 110, que es donde escucha el servicio POP3 (la opción `-C` comprime el tráfico por el túnel).

Hacer túneles sobre otros puertos es muy fácil. Por ejemplo, supongamos que *solo* tenemos acceso a un *remote proxy server* desde una máquina remota (*remote login*), no desde la máquina local; en este caso, se puede hacer un túnel para conectar el navegador a la máquina local. Consideremos que tenemos *login* sobre una máquina pasarela (*gateway*), que puede acceder a la máquina llamada *proxy*, la cual ejecuta el *Squid Proxy Server* sobre el puerto 3128. Ejecutamos:

```
ssh -C -L 8080:proxy:3128 user@gateway
```

Después de conectarnos, tendremos un túnel escuchando sobre el puerto local 8080 que reconducirá el tráfico desde *gateway* hacia *proxy* al 3128. Para navegar de forma segura solo se deberá hacer `http://localhost:8080/`.

1.4. Servicios de transferencia de ficheros: FTP

El FTP (*File Transfer Protocol*) es un protocolo cliente/servidor (bajo TCP) que permite la transferencia de archivos desde y hacia un sistema remoto. Un servidor FTP es un ordenador que ejecuta el *daemon ftpd*.

Algunos sitios que permiten la conexión anónima bajo el usuario *anonymous* son generalmente repositorios de programas. En un sitio privado, se necesitará un usuario y una contraseña para acceder. También es posible acceder a un servidor ftp mediante un navegador y generalmente hoy en día los repositorios de programas se sustituyen por servidores web (p. ej. Apache) u otras tecnologías como Bittorrent (que utiliza redes *peer to peer*, P2P) o servidores web con módulos de WebDav. No obstante, se continúa utilizando en algunos casos y Debian, por ejemplo, acceso con usuario o contraseña o la posibilidad de subir archivos al servidor (si bien con servicios web también es posible hacerlo). El protocolo (y los servidores/clientes que lo implementan) de ftp por definición no es encriptado (los datos, usuarios y contraseñas se transmiten en texto claro por la red) con el riesgo que ello supone. Sin embargo, hay una serie de servidores/clientes que soportan SSL y por lo tanto encriptación.

1.4.1. Cliente ftp (convencional)

Un cliente ftp permite acceder a servidores ftp y hay una gran cantidad de clientes disponibles. El uso del ftp es sumamente simple; desde la línea de comando, ejecutad:

```
ftp nombre -servidor
```

```
O también ftp y luego, de forma interactiva,: open nombre  
-servidor
```

El servidor solicitará un nombre de usuario y una contraseña (si acepta usuarios anónimos, se introducirá *anonymous* como usuario y nuestra dirección de correo electrónico como contraseña) y a partir del *prompt* del comando (después de algunos mensajes), podremos comenzar a transferir ficheros.

El protocolo permite la transferencia en modo ASCII o binario. Es importante decidir el tipo de fichero que hay que transferir porque una transferencia de un binario en modo ASCII inutilizará el fichero. Para cambiar de un modo a otro, se deben ejecutar los comandos `ascii` o `binary`. Los comandos útiles del cliente ftp son el `ls` (navegación en el directorio remoto), `get nombre_del_fichero` (para descargar ficheros) o `mget` (que admite *), `put nombre_del_fichero` (para enviar ficheros al servidor) o `mput` (que admite *); en estos dos últimos se debe tener permiso de escritura sobre el directorio del servidor. Se pueden ejecutar comandos locales si antes del comando se inserta un `!`. Por ejemplo, `!cd /tmp` significará que los archivos que bajen a la máquina local se descargarán en `/tmp`. Para poder ver el estado y el funcionamiento de la transferencia, el cliente puede imprimir marcas, o *ticks*, que se activan con los comandos `hash` y `tick`. Existen otros comandos que se pueden consultar en la hoja del manual (`man ftp`) o haciendo `help` dentro del cliente. Contamos con numerosas alternativas para los clientes, por ejemplo en modo texto: `ncftp`, `lukemftp`, `lftp`, `cftp`, `yafc` `Yafc` o, en modo gráfico: `gFTP`, `WXftp`, `LLNL XFTP`, `guiftp`.

Enlace de interés

En la Wikipedia disponéis de una comparativa de diversos clientes FTP: http://en.wikipedia.org/wiki/Comparison_of_FTP_client_software

1.4.2. Servidores FTP

El servidor tradicional de UNIX se ejecuta a través del puerto 21 y es puesto en marcha por el *daemon* `inetd` (o `xinetd`, según se tenga instalado). En `inetd.conf` conviene incluir el *wrapper* `tcpd` con las reglas de acceso en `host.allow` y el `host.deny` en la llamada al `ftpd` por el `inetd` para incrementar la seguridad del sistema. Cuando recibe una conexión, verifica el usuario y la contraseña y lo deja entrar si la autenticación es correcta. Un FTP *anonymous* trabaja de forma diferente, ya que el usuario solo podrá acceder a un directorio definido en el archivo de configuración y al árbol subyacente, pero no hacia arriba, por motivos de seguridad. Este directorio generalmente contiene directorios `pub/`, `bin/`, `etc/` y `lib/` para que el *daemon* de ftp pueda ejecutar comandos externos para peticiones de `ls`. El *daemon* `ftpd` soporta los siguientes archivos para su configuración:

- `/etc/ftpusers`: lista de usuarios que no son aceptados en el sistema. Un usuario por línea.
- `/etc/ftproot`: lista de usuarios a los que se les cambiará el directorio base `chroot` cuando se conecten. Necesario cuando deseamos configurar un servidor anónimo.
- `/etc/ftpwelcome`: anuncio de bienvenida.
- `/etc/motd`: noticias después del *login*.

El tema de la seguridad del sistema se estudia en el módulo "Administración de seguridad".



- **/etc/nologin**: mensaje que se muestra después de negar la conexión.
- **/var/log/ftpd**: *log* de las transferencias.

Si en algún momento queremos inhibir la conexión al ftp, se puede incluir el archivo `/etc/nologin`. El `ftpd` muestra su contenido y termina. Si existe un archivo `.message` en un directorio, el `ftpd` lo mostrará cuando se acceda al mismo. La conexión de un usuario pasa por cinco niveles diferentes:

- 1) tener una contraseña válida;
- 2) no aparecer en la lista de `/etc/ftpusers`;
- 3) tener un *shell* estándar válido;
- 4) si aparece en `/etc/ftpchroot`, se le cambiará al directorio `home` (incluido si es *anonymous* o `ftp`);
- 5) si el usuario es *anonymous* o `ftp`, entonces deberá tener una entrada en el `/etc/passwd` con usuario `ftp`, pero podrá conectarse especificando cualquier contraseña (por convención se utiliza la dirección de correo electrónico).

Es importante prestar atención a que los usuarios habilitados únicamente para utilizar el servicio ftp no dispongan de un *shell* a la entrada correspondiente de dicho usuario en `/etc/passwd` para impedir que este usuario tenga conexión, por ejemplo, por `ssh` o `telnet`. Para ello, cuando se cree el usuario, habrá que indicar, por ejemplo:

```
useradd -d/home/nteum -s /bin/false nteum
y luego: passwd nteum,
```

lo cual indicará que el usuario `nteum` no tendrá *shell* para una conexión interactiva (si el usuario ya existe, se puede editar el fichero `/etc/passwd` y cambiar el último campo por `/bin/false`). A continuación, se debe agregar como última línea `/bin/false` en `/etc/shells`. En [15] se describe paso a paso cómo crear tanto un servidor ftp seguro con usuarios registrados como un servidor ftp *anonymous* para usuarios no registrados. Dos de los servidores no estándares más comunes son el WUFTP y el ProFTP [4, 15].

Para instalar el Proftpd sobre Debian, ejecutad: `apt-get install proftpd`. Una vez descargado `debconf`, preguntará si se quiere ejecutar por `inetd` o manualmente (es recomendable elegir la última opción). Si se debe detener el servicio (para cambiar la configuración, por ejemplo): `/etc/init.d/proftpd stop` y para modificar el fichero: `/etc/proftpd.conf`. Un servidor (Debian) muy interesante es el PureFtpd (`pure-ftpd`), que es muy seguro, permite usuarios virtuales, cuotas, SSL/TSL y un conjunto de características interesantes.

Enlaces de interés

Para saber más sobre WUFTP y ProFTP podéis visitar las siguientes páginas web:
<http://en.wikipedia.org/wiki/WU-FTP> y
<http://www.proftpd.org>.

Enlaces de interés

Para configurar un servidor FTP en modo encriptado (TSL) o para tener acceso *anonymous* podéis consultar:
<http://www.debian-administration.org/articles/228>.
Por otro lado, para saber más sobre la instalación y configuración de PureFtpd podéis consultar:
<http://www.debian-administration.org/articles/383>.

1.5. Servicios de intercambio de información a nivel de usuario

1.5.1. El *Mail Transport Agent* (MTA)

Un MTA (*Mail Transport Agent*) se encarga de enviar y recibir los correos desde un servidor de correo electrónico hacia y desde Internet, que implementa el protocolo SMTP (*Simple Mail Transfer Protocol*). Debian utiliza por defecto `exim` ya que es más fácil de configurar que otros paquetes MTA, como son `smail` o `sendmail` (este último es uno de los precursores). `Exim` presenta características avanzadas tales como rechazar conexiones de sitios de *spam* conocidos, posee defensas contra correo basura (*junk mails*) o bombardeo de correo (*mail bombing*) y es extremadamente eficiente en el procesamiento de grandes cantidades de correos. Su ejecución se realiza a través de `inetd` en una línea en el archivo de configuración `/etc/inetd.conf` (o también a través de `xinetd`). `Exim` utiliza un archivo de configuración en `/etc/exim/exim.conf`, que puede ser modificado manualmente, si bien es recomendable hacerlo con un *shell script* llamado `eximconfig`, para poder configurar `exim` de forma interactiva. Los valores de la configuración dependerán de la situación de la máquina; sin embargo, su conexión es sumamente fácil ya que el mismo *script* sugiere valores por defecto. No obstante, en `/usr/doc/exim` pueden encontrarse ejemplos de configuraciones típicas.

Se puede probar si la configuración es válida con `exim -bv` y, si hay errores en el archivo de configuración, el programa los mostrará en pantalla o, si todo está correcto, solo pondrá la versión y la fecha. Para probar si puede reconocer un buzón (mailbox) local, se puede utilizar:

```
exim -v -bt usuario_local
```

Con este comando se mostrarán las capas de transporte utilizadas y la dirección local del usuario. También se puede hacer la siguiente prueba con un usuario remoto reemplazando el usuario local por una dirección remota para ver su comportamiento. A continuación, se deberá intentar enviar un correo local y otro remoto, pasando directamente los mensajes a `exim` (sin utilizar un agente, por ejemplo `mailx`) y tecleando, por ejemplo (todo junto):

```
exim postmaster@SuDominio
From: user@dominio
To: postmaster@SuDominio
Subject: Test Exim
Mensaje de prueba
Ctrl D
```

A continuación, se pueden analizar los archivos de traza `mainlog` y `paniclog` en `/var/log/exim/` para ver su comportamiento y cuáles son los mensajes de error generados. Obviamente, también se puede conectar al sistema como

usuario *postmaster* (o al que se haya enviado el mensaje) y leer los correos para ver si todo es correcto. Otra forma consiste en ejecutarlo en modo *debug* utilizando como parámetro `-dNro`, donde `Nro` es el nivel de *debug* (1-9). El parámetro normal con el cual se debe poner en marcha es `exim -bs`, ya sea por `inetd` o por `xinetd`. También es posible ejecutarlo como *daemon* a través de `/etc/init.d/exim start` en sistemas que necesiten prestaciones elevadas al tratamiento de los correos. Consultad la documentación (incluida en Debian en el paquete *exim-doc-html*) para configurar filtros, verificar *hosts*, *senders*, etc. También es interesante instalar el paquete *eximon*, que es un monitor del *exim* y permite al administrador ver la cola de correos, registros y realizar diferentes acciones con los mensajes en cola para distribuirlos (*freezing*, *bouncing*, *thawing*, etc.).

La última versión de Exim es **Exim4**. Se puede instalar con `apt-get install exim4-daemon-heavy` (instalad también `exim4-config` que servirá para configurar *exim4*); hay que tener en cuenta que si bien hay diferentes paquetes con diferentes posibilidades, *exim4-daemon-heavy* es la más completa. Una pequeña diferencia a tener en cuenta en su configuración es que en lugar de tener una única configuración `exim.conf` (es lo que tendrá si se instala *exim* desde los fuentes), el paquete `exim4-config` (es conveniente instalarlo para configurar *exim4*) utiliza pequeños archivos de configuración en lugar de uno único, que estarán en `/etc/exim4/config.d/*` y que se concatenarán en un único archivo (`/var/lib/exim4/config.autogenerated` por defecto) por `update-exim4.conf`.

Lectura recomendada

Sobre Exim4 es aconsejable que leáis `/usr/share/doc/exim/README.Debian.gz` y `update-exim4.conf(8)`. Para más información, podéis consultar el *HowTo* disponible en: <http://www.exim.org/docs.html>.

1.6. Internet Message Access Protocol (IMAP)

Este servicio permite acceder a los correos alojados en un servidor a través de un cliente de correo como por ejemplo Thunderbird o el cliente de correo de Seamonkey (ambos en mozilla.org). Este servicio soportado por el *daemon* `imapd` (los actuales soportan el protocolo IMAP4rev1) permite acceder a un archivo de correo electrónico (*mail file*) que se encuentra en una máquina remota. El servicio `imapd` se presta a través de los puertos 143 (`imap2`) o 993 (`imaps`) cuando soporta encriptación por SSL. Si se utiliza `inetd`, este servidor se pone en marcha a través de una línea en `/etc/inetd.conf` como:

```
imap2 stream tcp nowait root /usr/sbin/tcpd /usr/sbin/imapd
imap3 stream tcp nowait root /usr/sbin/tcpd /usr/sbin/imapd
```

En este ejemplo se llama al *wrapper* `tcpd` que funciona con `hosts.allow` y `hosts.deny` para incrementar la seguridad. Las aplicaciones más populares son `uw-imapd` (Universidad de Washington e instalado por defecto en Debian) o su versión segura `uw-imapd-ssl`, `cyrus-imap` o `courier-imap`. Para probar que el servidor `imap` funciona, se podría utilizar un cliente, por ejemplo Thunderbird, crear una cuenta para un usuario local, configurarlo adecuadamente para que se conecte sobre la máquina local y verificar el funcionamiento de `imap`.

Sobre Debian, la versión de `imap` ha sido compilada para soportar MD5 como método de autenticación de los usuarios remotos, para encriptar las contraseñas de conexión y evitar la suplantación de identidad por *sniffing* en la red (el cliente utilizado para conectarse al servidor `imap` también debe soportar el método de autenticación por MD5). El método es muy simple y seguro, pero el servidor debe conocer las contraseñas en texto plano de los usuarios de correo y por ello se recomienda utilizar la versión de `imapd` sobre SSL, que funciona sobre el puerto 993. El protocolo `imaps`, al igual que `ssh`, se basa en cifrar la comunicación a través de un certificado del *host* (el cliente utilizado para conectarse al servidor también debe soportar este método de conexión, por ejemplo, Thunderbird). Para configurar el servidor `imaps`, instalad el paquete `uw-imap-dssl` de Debian, que es el servidor `imap` con soporte SSL.

La instalación genera un certificado autofirmado válido por un año y lo almacena en `/etc/ssl/certs/imapd.pem`. Este certificado se puede reemplazar por uno firmado por una compañía certificadora o se puede generar uno propio con OpenSSL (o conseguir uno gratuito como, por ejemplo, de StartCom <https://www.startssl.com/?app=1>). Es conveniente dejar solo la entrada `imaps` en el archivo `/etc/inetd.conf` y quitar las entradas `imap2` e `imap3` si únicamente se quiere que el acceso a `imap` sea por SSL. Otro protocolo de similares características que ha tenido mucha popularidad en el pasado, pero que hoy se ha visto superado por IMAP, es POP (*Post Office Protocol*), versiones 2 y 3. Su instalación y puesta en marcha es análoga a la de IMAP. Existen multitud de servidores POP, pero los más comunes son `courier-pop`, `cyrus-pop3d` y `ipopd` (Universidad de Washington), `qpopper`, `solid-pop3d`.

1.6.1. Aspectos complementarios

Supongamos que como usuarios tenemos cuatro cuentas de correo en servidores diferentes y queremos que todos los mensajes que llegan a estas cuentas se recojan en una única, a la que podamos acceder externamente, y que haya también un filtro de correo basura (*antispam*).

Primero se deben instalar `Exim + Imap` y comprobar que funcionan. Se debe tener en cuenta que si se instala `courier-imap` (que según algunos autores es mejor que `uw-imapd`), este funciona sobre un formato de correo llamado `maildir` y que se debería configurar `Exim` para que también funcione sobre `maildir` con la siguiente configuración en `/etc/exim/exim.conf` (o en la correspondiente si se tiene `exim4`) cambiando la opción `mail_dir format = true` (los correos se guardarán en la cuenta del usuario local en un directorio llamado `Maildir`). A continuación, se debe reiniciar el servidor `exim` con `/etc/init.d/exim restart`, repetir la prueba de funcionamiento enviándonos un mensaje y verificar que se puede leer con un cliente que soporte `maildir`*

*Por ejemplo, `mutt`; `mailx` no lo soporta. Consultad <http://www.mutt.org>.

Para recoger los mensajes de diferentes cuentas se utilizará `Fetchmail`, (que se instala con `apt-get install fetchmail`). A continuación, se debe crear el fichero `.fetchmailrc` en nuestro `$HOME` (también se puede utilizar la herramienta `fetchmailconf`) que deberá tener ser algo así como:

```
set postmaster "pirulo"
set bouncemail
set no spambounce
set flush
poll pop.domain.com proto pop3
  user 'user1' there with password 'secret' is pirulo here
poll mail.domain2.com
  user 'user5' there with password 'secret2' is 'pirulo' here
  user 'user7' there with password 'secret3' is 'pirulo' here
```

La acción `set` indica a `Fetchmail` que esta línea contiene una opción global (envío de errores, eliminación de los mensajes de los servidores, etc.). A continuación, se especifican dos servidores de correo: uno para que compruebe si hay correo con el protocolo POP3 y otro para que pruebe a usar varios protocolos con el fin de encontrar uno que funcione. Se comprueba el correo de dos usuarios con la segunda opción de servidor, pero todo el correo que se encuentre se envía al *pool* de correo de `pirulo`. Esto permite comprobar varios buzones de diversos servidores como si se tratara de un único buzón MUA. La información específica de cada usuario comienza con la acción `user`. El `Fetchmail` se puede poner en el `cron` (por ejemplo, en `/var/spool/cron/crontabs/pirulo` agregando `1 * * * * /usr/bin/fetchmail -s`) para que se ejecute automáticamente o ejecutarlo en modo *daemon* (poned `set daemon 60` en `.fetchmailrc` y ejecutarlo una vez, por ejemplo, en `autostart` de `Gnome/KDE` o en el `.bashrc` `-se` ejecutará cada 60 segundos-).

Para quitar el correo basura se utilizará `SpamAssassin` y se puede configurar `Kmail` o `Evolution` (consultad la bibliografía para ver cómo configurarlo) para que lo ejecuten. En esta configuración se utilizará `Procmail`, que es una herramienta muy potente (permite repartir el correo, filtrarlo, reenviarlo automáticamente, etc.). Una vez instalado (`apt-get install procmail`), se debe crear un fichero llamado `.procmailrc` en el home de cada usuario, que llamará al `SpamAssassin`:

```
# Poned yes para mensajes de funcionamiento o depuración
VERBOSE=no
# Consideramos que los mensajes están en "~/Maildir", cambiar si es otro
PATH=/usr/bin:/bin:/usr/local/bin:
MAILDIR=$HOME/Maildir
DEFAULT=$MAILDIR/

# Directorio para almacenar los ficheros
PMDIR=$HOME/.procmail
# Comentar si no queremos log de Procmail
LOGFILE=$PMDIR/log
# filtro de Smap
INCLUDERC=$PMDIR/spam.rc
```

Podéis instalar `SpamAssassin` mediante `apt-get install spamassassin`.

El archivo `~/ .procmail/spam.rc` contiene:

```
# si el SpamAssassin no está en el PATH, agregar a la variable PATH el directorio
:Ofw: spamassassin.lock
| spamassassin -a

# La tres líneas siguientes moverán el correo Spam a un directorio llamado
# "spam-folder". Si se quiere guardar el correo en la bandeja de entrada,
# para luego filtrarlo con el cliente, comentad las tres líneas.

:0:
* ^X-Spam-Status: Yes
spam-folder
```

El archivo `~/ .spamassassin/user_prefs` contiene algunas configuraciones útiles para SpamAssassin (consultad la bibliografía).

```
# user preferences file. Ved man Mail::SpamAssassin::Conf

# Umbral para reconocer un Spam.
# Default 5, pero con 4 funciona un poco mejor
required_hits 4

# Sitios de los que consideraremos que nunca llegará Spam
whitelist_from root@debian.org
whitelist_from *@uoc.edu

# Sitios de los que siempre llega SPAM (separados por comas)
blacklist_from viagra@dominio.com

# las direcciones en Whitelist y Blacklist son patrones globales como:
# "amigo@lugar.com", "*@isp.net", o "*.domain.com".

# Insertad la palabra SPAM en el subject (facilita hacer filtros).
# Si no se desea comentar la línea.
subject_tag [SPAM]
```

Esto generará un *tag* `X-Spam-Status: Yes` en la cabecera del mensaje si se cree que el mensaje es *spam*. Luego se deberá filtrar y poner en otra carpeta o borrarlo directamente. Se puede usar el `procmail` para filtrar mensajes de dominios, usuarios, etc. Por último, se puede instalar un cliente de correo y configurar los filtros para que seleccionen todos los correos con `X-Spam-Status: Yes` y los borre o los envíe a un directorio. Después verificaremos los falsos positivos (correos identificados como basura pero que no lo son). Un aspecto complementario de esta instalación es que si se desea tener un servidor de correo a través de correo web (*webmail*, es decir poder consultar los correos del servidor a través de un navegador sin tener que instalar un cliente ni configurarlo, igual que consultar una cuenta de Gmail o Hotmail) es posible instalar Squirrelmail (`apt-get install squirrelmail`) para dar este servicio.

Enlace de interés

Hay otras posibilidades como instalar MailDrop en lugar de Procmail, Postfix en lugar de Exim, o incluir Clamav/Amavisd como antivirus (Amavisd permite vincular Postfix con SpamAssassin y Clamav). Para saber más sobre este tema podéis visitar la siguiente página web: <http://www.debian-administration.org/articles/364>.

Enlace de interés

Para más información sobre `procmail` y el filtrado de mensajes, consultad: <http://www.debian-administration.org/articles/242>.

Enlace de interés

Sobre Squirrelmail en Debian, consultad: <http://www.debian-administration.org/articles/200>.

1.7. Grupos de discusión

Las *news* o grupos de discusión son soportados a través del protocolo NNTP. Instalar un servidor de grupos de discusión es necesario cuando se desea leer *news* fuera de línea, cuando se quiere tener un repetidor de los servidores centrales o se quiere un propio servidor maestro de *news*. Los servidores más comunes son INN o CNEWS, pero son paquetes complejos y destinados a grandes servidores. Leafnode es un paquete USENET que implementa el servidor TNP, especialmente indicado para sitios con grupos reducidos de usuarios, pero donde se desea acceder a gran cantidad de grupos de noticias. Este servidor se instala en la configuración básica de Debian y se pueden reconfigurar con `dpkg-reconfigure leafnode` todos parámetros como los servidores centrales, el tipo de conexión, etc. Este *daemon* se pone en marcha desde `inetd` de forma similar al `imap` (o con `xinetd`). Leafnode soporta filtros a través de expresiones regulares indicadas (del tipo `^Newsgroups:. * [,] alt.flame$`) en `/etc/news/leafnode/filters`, donde para cada mensaje se compara la cabecera con la expresión regular y, si existe coincidencia, el mensaje se rechaza.

La configuración de este servidor es simple y todos los archivos deben ser propiedad de un usuario de *news* con permiso de escritura (se debe verificar que dicho propietario existe en `/etc/passwd`). Todos los archivos de control, *news* y la configuración se encuentran en `/var/spool/news`, excepto la configuración del propio servidor que está en el fichero `/etc/news/leafnode/config`. En la configuración existen algunos parámetros obligatorios que se deben configurar (por ejemplo, para que el servidor pueda conectarse con los servidores maestros), como son `server` (servidor de *news* desde donde se obtendrán y enviarán las *news*) y `expire` (número de días a los que un hilo o sesión se borrará tras haber sido leído). Tenemos, asimismo, un conjunto de parámetros opcionales de ámbito general o específico del servidor que podrían configurarse. Para más información, consultad la documentación (`man leafnode` o `/usr/doc/leafnode/README.Debian`). Para verificar el funcionamiento del servidor, se puede hacer `telnet localhost nntp` y, si todo funciona correctamente, saldrá la identificación del servidor y se quedará esperando un comando. Como prueba, se puede introducir `help` (para abortar, haced `Ctrl+C` y luego `Quit`).

1.8. World Wide Web (httpd)

Apache es uno de los servidores más populares y con mayores prestaciones de HTTP (*HyperText Transfer Protocol*). Apache tiene un diseño modular y soporta extensiones dinámicas de módulos durante su ejecución. Es muy configurable en cuanto al número de servidores y de módulos disponibles y soporta diversos mecanismos de autenticación, control de acceso, *metafiles*, *proxy caching*, servidores virtuales, etc. Con módulos (incluidos en Debian) es posible tener PHP3, Perl, Java Servlets, SSL y otras extensiones*.

*Podéis consultar la documentación en <http://www.apache.org>.

Apache está diseñado para ejecutarse como un proceso *daemon standalone*. En esta forma, crea un conjunto de procesos hijo que gestionarán las peticiones de entrada. También puede ejecutarse como *Internet daemon* a través de `inetd`, por lo que se pondrá en marcha cada vez que se reciba una petición. La configuración del servidor puede ser extremadamente compleja según las necesidades (consultad la documentación); sin embargo, aquí veremos una configuración mínima aceptable. Los archivos de configuración se encuentran en `/etc/apache` y son `httpd.conf` (archivo principal de configuración), `srm.conf`, `access.conf` (estos dos últimos se mantienen por compatibilidad y su funcionalidad está en el anterior), `mime.conf` (formatos MIME) y `magic` (número de identificación de archivos). Los archivos de registro se encuentran en `/var/log/apache` y son `error.log` (registra los errores en las peticiones del servidor), `access.log` (registra quién ha accedido y a qué) y `apache.pid` (identificador del proceso). Apache se pone en marcha desde el *script* de inicio `/etc/init.d/apache` y los `/etc/rcX.d`, pero puede controlarse manualmente mediante el comando `apachectl`. También se puede utilizar el comando `apacheconfig` para configurar el servidor. Los directorios por defecto (en Debian) son:

- 1) `/var/www`: directorio de documentos HTML
- 2) `/usr/lib/cgi-bin`: directorio de ejecutables (cgi) por el servidor
- 3) `http://server.dominio/~user`: páginas personales de los usuarios
- 4) `/home/user/public.html`: directorio de páginas personales

El archivo que se lee por defecto de cada directorio es `index.html`. Una vez instalados los paquetes `apache` y `apache-common`, Debian configura básicamente el servidor y lo pone en marcha. Se puede comprobar que funciona abriendo un navegador (por ejemplo, el Konqueror) y poniendo en la barra de URL `http://localhost`, lo cual cargará la página `/var/www/index.html`.

1.8.1. Configuración manual (mínima) de `httpd.conf`

Vamos a ver algunos de los parámetros más importantes en la configuración de Apache (el ejemplo está tomado de la versión 1.X de Apache y existen algunos cambios menores si se utiliza la versión 2).

```
ServerType standalone
    Recomendado, más eficiente
ServerRoot /etc/apache
    Donde están los archivos de configuración
Port 80
    Donde el servidor escuchará las peticiones
User www-data
    Los user y group con los cuales se ejecutará el servidor (importante
    por seguridad) deben ser usuarios válidos (pueden estar locked)
Group www-data
```



```

ServerAdmin webmaster@pirulo.remix.com
    Dirección de usuario que atenderá los errores
ServerName pirulo.remix.com
    Nombre del servidor enviado a los usuarios (debe ser
    un nombre válido en /etc/host o DNS)
DocumentRoot /var/www
    Directorio donde estarán los documentos
Alias /icons/ /usr/share/apache/icons/
    Donde se encuentran los iconos
ScriptAlias /cgibin/ /usr/lib/cgibin/
    Donde se encuentran los script CGI

```

1.8.2. Apache 2.2 + SSL + PHP + MySQL

Una cuestión importante para los servidores web dinámicos es aprovechar las ventajas de Apache en modo seguro (SSL), PHP (un lenguaje de programación usado generalmente para la creación de contenido para sitios web) y MySQL+PHPAdmin, todo ello funcionando conjuntamente. Partiremos de la instalación sobre un Debian pero no a través de paquetes `deb` sino desde el software bajado de los sitios respectivos; así se puede repetir la experiencia sobre otras distribuciones. Obviamente, después estos paquetes no podrán ser controlados por `apt` u otro gestor de paquetes. Se debe tener cuidado con las versiones que pueden cambiar y de no superponer la instalación a paquetes ya instalados.

1) Descarga de los ficheros necesarios (por ejemplo, dentro del directorio `/root` ->`cd /root`):

```

Apache: desde http://httpd.apache.org/download.cgi: httpd-2.2.4.tar.bz2
PHP: desde http://www.php.net/downloads.php PHP 5.2.1 (tar.bz2)
MySQL desde http://mysql.org/get/Downloads/MySQL-4.1/mysql-standard-4.1.21-pc-
linux-gnu-i686.tar.gz/from/pick
PHPAdmin desde http://prdownloads.sourceforge.net/phpmyadmin/phpMyAdmin
-2.9.1-all-languages.tar.bz2?download

```

2) Utilidades: `bzip2 libssl-dev openssl gcc g++ cpp make`. Verificad que no se tienen instaladas; de lo contrario, haced `apt-get install bzip2 libssl-dev openssl gcc g++ cpp make`

3) Apache:

```

cd /root
tar jxvf httpd-2.2.4.tar.bz2
cd httpd-2.2.4

```

Con `prefix` indicamos que se instalará, por ejemplo, `/usr/local/apache2`

```

./configure --prefix=/usr/local/apache2 --with-ssl=/usr/include/openssl --enable-ssl
make
make install

```

Modificamos el fichero de configuración `/usr/local/apache2/conf/httpd.conf` y cambiamos el usuario y el grupo de trabajo por `www-data`

```

User www-data
Group www-data

```


Cambiamos el dueño y el grupo del directorio de datos a `www-data`:

```
chown -R www-data:www-data /usr/local/apache2/htdocs
```

Modificamos el usuario `www-data` para cambiar su directorio `home` en `/etc/passwd`:

```
www-data:x:33:33:www-data:/usr/local/apache2/htdocs:/bin/sh
```

Servidor Apache instalado. Para iniciarlo (para detenerlo, cambiad `start` por `stop`):

```
/usr/local/apache2/bin/apachectl start
```

Se puede colocar un *script* para arrancar el servidor Apache en el *boot*.

```
ln -s /usr/local/apache2/bin/apachectl /etc/rcS.d/S99apache
chmod 755 /etc/rcS.d/S99apache
```

4) SSL:

En `/usr/local/apache2/conf/httpd.conf`, quitamos el comentario de la línea:

```
Include conf/extra/httpd-ssl.conf
```

Se generan los ficheros con las claves para el servidor seguro, en `/root` hacemos (hay que adecuar las versiones a las que se hayan descargado). El primer comando `openssl` es una línea entera y acaba con `1024`-

```
openssl genrsa -rand ../httpd-2.2.4.tar.bz2:../php-5.2.1.tar.bz2:../phpMyAdmin-2.9.1-
all-languages.tar.bz2 -out server.key 1024
openssl rsa -in server.key -out server.pem
openssl req -new -key server.key -out server.csr
openssl x509 -req -days 720 -in server.csr -signkey server.key -out server.crt
```

Se copian los ficheros...

```
cp server.crt /usr/local/apache2/conf/
cp server.key /usr/local/apache2/conf/
```

Reiniciamos el servidor...

```
/usr/local/apache2/bin/apachectl restart
```

5) MySQL:

Creamos un grupo y un usuario para MySQL, si no existe:

```
groupadd mysql
useradd -g mysql mysql
```

En el directorio donde se instalará MySQL (`/usr/local/`), hacemos:

```
cd /usr/local/
gunzip </root/mysql-standard-4.1.21-pc-linux-gnu-i686.tar.gz | tar xvf -
ln -s mysql-standard-4.1.21-pc-linux-gnu-i686 mysql
cd mysql
```

Creamos una base de datos y cambiamos los permisos:

```
scripts/mysql_install_db --user=mysql
chown -R root .
chown -R mysql data
chgrp -R mysql .
```

Se puede colocar un *script* para iniciar el servidor MySQL.

```
ln -s /usr/local/mysql/support-files/mysql.server /etc/rcS.d/S99mysql.server
chmod 755 /etc/rcS.d/S99mysql.server
```

Iniciamos el servidor:

```
/etc/rcS.d/S99mysql.server start
```

Se puede entrar en la base de datos y cambiar la contraseña del `root` por seguridad*.

```
/usr/local/mysql/bin/mysql
```

Dentro, hacemos:

```
USE mysql
```

Colocamos la contraseña `pirulo` al usuario `root`

```
UPDATE user SET Password=PASSWORD('pirulo') WHERE User='root';
FLUSH privileges;
```

Para entrar en MySQL deberemos hacer:

```
/usr/local/mysql/bin/mysql -u root -ppirulo
```

Enlace de interés

Se puede consultar cómo agregar el módulo SSL a un servidor que no lo tenga instalado en <http://www.debian-administration.org/articles/349>.

*Consultad

<http://dev.mysql.com/doc/refman/5.0/en/index.html>
para la sintaxis

6) PHP (reemplazad con las versiones adecuadas):

Utilidades necesarias:

```
apt-get install libxml2-dev curl libcurl3-dev libjpeg-mmx-dev zlib1g-dev libpng12-dev
```

Con el servidor Apache parado, hacemos:

```
cd /root
tar jxvf php-5.2.0.tar.bz2
cd php-5.2.0
```

Con `prefix` se puede indicar dónde se quiere instalar (toda una línea):

```
./configure - -prefix=/usr/local/php5 - -enable-mbstring - -with-apxs2=/usr/local/apache2/bin/apxs
- -with-mysql=/usr/local/mysql - -with-curl=/usr/include/curl
- -with-jpeg-dir=/usr/include - -with-zlib-dir=/usr/include - -with-gd - -with-xml - -enable-
ftp - -enable-bcmath
make
make install
cp php.ini-dist /usr/local/php5/lib/php.ini
```

Modificamos Apache (`/usr/local/apache2/conf/httpd.conf`) en la parte indicada:

```
<IfModule mime_module>
    AddType application/x-httpd-php .php .phtml
    AddType application/x-httpd-php-source .phps
```

Y también:

```
DirectoryIndex index.php index.html
```

Reiniciamos el servidor:

7) PHPAdmin

```
cd /usr/local/apache2/
```

Se descomprime `phpmyadmin` en el directorio de `apache2` (hay que ir con cuidado con las versiones):

```
tar jxvf /root/phpMyAdmin-2.9.1-all-languages.tar.bz2
mv phpMyAdmin-2.9.1-all-languages phpmyadmin
cd phpmyadmin
cp config.sample.inc.php config.inc.php
```

Se debe modificar el fichero de configuración (`config.inc.php`):

```
$cfg['blowfish_secret'] = 'pirulo';
```

Quitamos el usuario y la contraseña del usuario por defecto con dos (') seguidas:

```
$cfg['Servers'][$i]['controluser'] = "";
$cfg['Servers'][$i]['controlpass'] = "";
```

Cambiamos Apache (`/usr/local/apache2/conf/httpd.conf`) añadiendo en `<IfModule alias_module>`:

```
<IfModule alias_module>
    Alias /phpmyadmin "/usr/local/apache2/phpmyadmin/"
<Directory "/usr/local/apache2/phpmyadmin/">
    Order allow,deny
    Allow from all
</Directory>
```

Reiniciamos el servidor, que se puede llamar con `http://localhost/phpadmin`.

Enlace de interés

Se puede tener más información en los sitios respectivos de cada aplicación y en LWP: http://www.lawebdelprogramador.com/temas/tema_stablephpapachemysql.php.

1.9. Servidor de WebDav

El nombre webDAV son las siglas de *Web Based Distributed Authoring and Versioning* (también se refiere al grupo de trabajo de Internet Engineering Task Force) y es un protocolo que permite que la web se transforme en un medio legible y editable y proporciona funcionalidades para crear, cambiar y mover documentos en un servidor remoto (típicamente un servidor web). Esto se utiliza sobre todo para permitir la edición de los documentos que envía un servidor web, pero puede también aplicarse a sistemas de almacenamiento generales basados en la web y a los que se puede acceder desde cualquier lugar. En este subapartado instalaremos un servidor WebDav sobre Apache. El proceso es el siguiente:

- 1) Instalar el servidor Apache, realizar las configuraciones mínimas y verificar que funciona (instalado, por ejemplo, con `apt-get install apache2`).
- 2) Habilitar los módulos de Apache que son necesarios para WebDav: `a2enmod dav_fs` y `a2enmod dav`.
- 3) Crear el directorio para el directorio virtual (podemos hacer por ejemplo `mkdir -p /var/www/test`) y permitir que Apache sea el propietario del directorio `chown www-data /var/www/test/`.
- 4) En el archivo `/etc/apache2/sites-available/default` crear un directorio virtual,

```
<VirtualHost *>
  ServerAdmin root@localhost
  DocumentRoot /var/www/test/
  <Directory /var/www/test>
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>
  Alias /webdav /var/www/test
  <Location /webdav>
    DAV On
    AuthType Digest
    AuthName "webdav"
    AuthDigestProvider file
    AuthUserFile /var/www/test/digest-password
    Require valid-user
  </Location>
</VirtualHost>
```

en el que se puede ver que se ha decidido utilizar una autenticación segura con MD5 Digest. Se puede comprobar que la configuración es correcta con `apache2ctl configtest`.

- 5) Se crea el sistema de autenticación habilitando el módulo `digest` de Apache `a2enmod auth_digest` y se crea la contraseña para un usuario: `htdigest -c /var/www/test/digest-password webdav admin`, que nos pedirá la contraseña para el usuario `admin` (dos veces para verificarlo) y lo guardará en el fichero encriptado.

Enlace de interés

Sobre la integración de WebDav con Apache podéis consultar el artículo "WebDAV on Apache2" disponible en: <http://www.debian-administration.org/articles/285>

6) Se reinicia Apache para que lea la configuración `/etc/init.d/apache2 reload` y ya nos podemos conectar a `http://localhost/webdav`, previa autenticación.

7) Otra forma de probarlo es con un cliente WebDav, por ejemplo Cadaver*, con `apt-get install cadaver`. A continuación, nos conectamos al servidor con `cadaver http://localhost/webdav` y después de autenticarnos, podemos crear un directorio (`mkdir`), editar un archivo, listar un directorio (`ls`), cambiar de directorio (`cd`), cambiar los permisos de ejecución (`chexec`), borrarlo (`rm`), etc.

*<http://www.webdav.org/cadaver>

1.10. Servicio de *proxy*: Squid

Un servidor *proxy* (*proxy server*, PS) se utiliza para ahorrar ancho de banda de la conexión de red, mejorar la seguridad e incrementar la velocidad para obtener páginas de la red (*web-surfing*).

Squid es uno de los principales PS, ya que es *OpenSource* y acepta ICP (características que le permiten intercambiar *hints* con otros PS), SSL (para conexiones seguras entre *proxies*) y soporta objetos FTP, Gopher, HTTP y HTTPS (seguro). Su funcionamiento es simple, almacena los objetos más solicitados en memoria RAM y los menos, en una base de datos en el disco. Los servidores Squid, además, pueden configurarse de forma jerárquica para formar un árbol de *proxies* dependiendo de las necesidades. Existen dos configuraciones posibles:

- 1) como acelerador de `httpd` para lograr más prestaciones al servicio de la web;
- 2) como *proxy-caching server* para permitir a los usuarios de una corporación utilizar el PS para salir hacia Internet.

En el primer modo, actúa como *proxy* inverso, es decir, acepta una petición del cliente, sirve el objeto si lo tiene y si no, lo solicita, se lo pasa al cliente cuando lo tiene y lo almacena para la próxima vez. En la segunda opción, se puede utilizar como control y para restringir los sitios donde se puede conectar en Internet o autorizar el acceso a determinadas horas del día. Una vez instalado (paquete `squid` en Debian, también se puede instalar `squid-cgi`, `squidguard` o `squidtailed`), se generan tres archivos: `/etc/squid.conf` (configuración), `/etc/init.d/squid` (inicialización) y `/etc/logrotate.d/squid` (de control de los registros).

1.10.1. Squid como acelerador de http

En este modo, si el servidor de web está en la misma máquina que el PS, se deberá reconfigurar para que atienda peticiones del puerto 81 (en Apache,

cambiar Port 80 por Port 81 en `httpd.conf`). El archivo de configuración (`/etc/squid.conf`) contiene una gran cantidad de entradas, pero aquí solo veremos las indispensables [15]:

```
http_port 80    Donde escucha httpd
icp_port 0     Donde escucha ICP
hierarchy_stoplist cgi-bin \?
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY
cache_mem 100 MB    Memoria para objetos en curso
redirect_rewrites_host_header off
cache_replacement_policy lru
memory_replacement_policy lru
cache_dir ufs /var/spool/squid 100 16 256
    Tipo y lugar donde está la base de datos de caché de disco
emulate_httpd_log on
acl all src 0.0.0.0/0.0.0.0    Acceso para todos
http_access allow all    Y a todo
cache_mgr root    Mail responsable
cache_effective_user proxy    UID
cache_effective_group proxy    GID
httpd_accel_host 192.168.1.1    Servidor real de web
httpd_accel_port 81    Puerto
logfile_rotate 0
log_icp_queries off
buffered_logs on
```

De este modo, la opción `httpd_accel_host` desactiva la posibilidad de que se ejecute como *proxy-caching*.*

*Para más información,
consultad
<http://www.squid-cache.org>.

1.10.2. Squid como *proxy-caching*

De esta manera se habilita el Squid para que controle el acceso a Internet, cuándo accederán y a qué accederán los usuarios. En este caso, el archivo de configuración deberá incluir la siguientes modificaciones o agregados en `/etc/squid.conf`:

```
acl localnet src 192.168.1.0/255.255.255.0
acl localhost src 127.0.0.1/255.255.255.255
acl Safe_ports port 80 443 210 70 21 102565535
acl CONNECT method CONNECT
acl all src 0.0.0.0/0.0.0.0
http_access allow localnet
http_access allow localhost
http_access deny Puerto seguros
http_access deny CONNECT
http_access deny all
cache_emulate_httpd_log on
```

La gran diferencia con el otro modo son las líneas `acl`, donde, en este caso, se permitirá a los clientes de la clase C 192.168.1.0 acceder al PS; también el `localhost` IP y otros puertos que podrán acceder a Internet 80(http), 443(https), 210(wais), 70(gopher) y 21(ftp). Además, se niega el método `connect` para evitar que desde fuera se puedan conectar al PS y luego se niegan todos los IP y puertos sobre el PS [15].

Enlaces de interés

Más información en <http://www.squid-cache.org/> y para un *transparent-proxy*, en <http://tldp.org/HOWTO/TransparentProxy-1.html>.

1.11. OpenLdap (Ldap)

LDAP significa *Lightweight Directory Access Protocol* y es un protocolo para acceder a datos basados en un servicio X.500. Este se ejecuta sobre TCP/IP y el directorio es similar a una base de datos que contiene información basada en atributos. El sistema permite organizar esta información de manera segura y utiliza réplicas para mantener su disponibilidad, lo que asegura la coherencia y la verificación de los datos accedidos-modificados.

El servicio se basa en el modelo cliente-servidor, donde existen uno o más servidores que contienen los datos; cuando un cliente se conecta y solicita información, el servidor responde con los datos o con un puntero a otro servidor de donde podrá extraer más información; sin embargo, el cliente solo verá un directorio de información global [15, 18]. Para importar y exportar información entre servidores ldap o para describir una serie de cambios que se aplicarán al directorio, el formato utilizado se llama LDIF (*LDAP Data Interchange Format*). LDIF almacena la información en jerarquías orientadas a objetos que luego serán transformadas al formato interno de la base de datos. Un archivo LDIF tiene un formato similar a:

```
dn: o = UOC, c = SP o: UOC
objectclass: organization
dn: cn = Pirulo Nteum, o = UOC, c = SP
cn: Pirulo Nteum
sn: Nteum
mail: nteumuoc.edu
objectclass: person
```

Cada entrada se identifica con un nombre indicado como DN (*Distinguished Name*). El DN consiste en el nombre de la entrada más una serie de nombres que lo relacionan con la jerarquía del directorio y donde existe una clase de objetos (`objectclass`) que define los atributos que pueden utilizarse en esta entrada. LDAP provee un conjunto básico de clases de objetos: **grupos** (incluye listas desordenadas de objetos individuales o grupos de objetos), **localizaciones** (tales como países y su descripción), **organizaciones** y **personas**. Una entrada puede, además, pertenecer a más de una clase de objeto, por ejemplo, un individuo es definido por la clase `person`, pero también puede ser definido por atributos de las clases `inetOrgPerson`, `groupOfNames` y `organization`.

La estructura de objetos del servidor (llamado *schema*) determina cuáles son los atributos permitidos para un objeto de una clase (que se definen en el archivo `/etc/ldap/schema` como `inetorgperson.schema`, `nis.schema`, `opeldap.schema`, `corba.schema`, etc.). Todos los datos se representan como un par atributo = valor, donde el atributo es descriptivo de la información que contiene; por ejemplo, el atributo utilizado para almacenar el nombre de una persona es `commonName`, o `cn`, es decir, una persona llamada Pirulo Nteum se representará por `cn: Pirulo Nteum` y llevará asociado otros atributos de la clase `person` como `givenname: Pirulo` `surname: Nteum` `mail:`

pirulo@uoc.edu. En las clases existen atributos obligatorios y optativos y cada atributo tiene una sintaxis asociada que indica qué tipo de información contiene el atributo, por ejemplo, `bin` (*binary*), `ces` (*case exact string*, debe buscarse igual), `cis` (*case ignore string*, pueden ignorarse mayúsculas y minúsculas durante la búsqueda), `tel` (*telephone number string*, se ignoran espacios y '-') y `dn` (*distinguished name*). Un ejemplo de un archivo en formato LDIF podría ser:

```
dn: dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit
```

```
dn: ou = groups, dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit
ou: groups
```

```
dn: ou = people, dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit
ou: people
```

```
dn: cn = Pirulo Nteum, ou = people, dc = UOC, dc = com
cn: Pirulo Nteum
sn: Nteum
objectclass: top
objectclass: person
objectclass: posixAccount
objectclass: shadowAccount
uid:pirulo userpassword:{crypt}p1ps2ii(0pgbs*do& = )eksd uidnumber:104
gidnumber:100
gecos:Pirulo Nteum
loginShell:/bin/bash
homeDirectory: /home/pirulo
shadowLastChange:10877
shadowMin: 0
shadowMax: 999999
shadowWarning: 7
shadowInactive: -1
shadowExpire: -1
shadowFlag: 0
```

```
dn:
cn = unixgroup, ou = groups, dc = UOC, dc = com
objectclass: top
objectclass: posixGroup
cn: unixgroup
gidnumber: 200
memberuid: pirulo
memberuid: otro-usuario
```

Las líneas largas pueden continuarse debajo comenzando por un espacio o un tabulador (formato LDIF). En este caso, se ha definido la base DN para la institución `dc = UOC, dc = com`, la cual contiene dos subunidades: `people` y `groups`. A continuación, se ha descrito un usuario que pertenece a `people` y a `group`. Una vez preparado el archivo con los datos, este debe ser importado al servidor para que esté disponible para los clientes LDAP. Existen herramientas para transferir datos de diferentes bases de datos a formato LDIF [18]. Sobre Debian, se debe instalar el paquete `slapd`, que es el servidor de OpenLdap. Durante la instalación, realizará una serie de pregun-

tas como: método de instalación del directorio: auto; extensiones al directorio (*domain-host*, sitio, institución): *host*, *domain*, contraseña del administrador; replicar cambios locales a otros servidores: no. Esta instalación generará un archivo de configuración en `/etc/ldap/slapd.conf` y la base de datos sobre `/var/lib/ldap`. También existe otro archivo `/etc/ldap/ldap.conf` (o puede existir el `$HOME/.ldaprc`), que es el archivo de configuración utilizado para inicializar valores por defecto cuando se ejecutan clientes ldap. En este se indica cuál es la base de datos, cuál es el servidor ldap, los parámetros de seguridad, el tamaño de la búsqueda, etc.

El archivo de configuración del servidor `/etc/ldap/slapd.conf` (ved `man slapd.conf`) está compuesto por diferentes secciones, cada una de ellas indicada por una de las siguientes directivas: *global*, *backend specific* y *database specific*, en este orden. La directiva *global* es de carácter general, se aplica a todos los *backends* (bases de datos) y define cuestiones generales como los permisos de acceso, los atributos, los tiempos de espera, los *schemas*, etc. La directiva *backend specific* define los atributos al *backend* específico que define (*bdb*, *dnssrv*, *ldbm*, etc.) y el *database specific*, los atributos específicos para esa base de datos que define. Para poner en marcha el servidor, se debe ejecutar: `/etc/init.d/slapd start` (o `/etc/init.d/slapd stop` para detenerlo). Durante la instalación, el sistema habrá creado los enlaces adecuados para ejecutarlo después del inicio.

1.11.1. Creación y mantenimiento de la base de datos

Existen dos métodos para introducir datos en la base de datos de LDAP. El primero es fácil y adecuado para pequeñas cantidades de datos, es interactivo y se deben utilizar herramientas tales como `ldapadd`, `gq`, `phpldapadmin`, `JXplorer`, etc. para introducir nuevas entradas. Con el segundo se debe trabajar fuera de línea, es el adecuado para grandes BD y se utiliza el comando `slapadd` incluido con `slapd`. Por ser más general, en este subapartado describiremos sintéticamente el segundo método (dejaremos el primero para un caso de uso que será explicado a continuación), donde primero se debe verificar que contiene los siguientes atributos en `slapd.conf`: `suffix` (*top* del directorio, por ejemplo, sufijo "o = nteum, c = org"); directorio `/var/lib/ldap` (directorio donde se crearán los índices y que pueda escribir `slapd`). Se debe verificar además que la base de datos contiene las definiciones de los índices que se desean:

```
index cn,sn,uid
index objectClass pres,eq
```

Una vez definido el `slapd.conf`, se debe ejecutar el comando:

```
slapadd -l entrada -f configuración [-d nivel] [-n
entero | -b sufijo]
```


Los argumentos son:

- l: archivo en formato LDIF.
- f: archivo de configuración del servidor, donde se indica cómo crear los índices.
- d: nivel de depuración.
- n: n.º de base de datos, si se tiene más de una.
- b: especifica qué base de datos hay que modificar.

Existen otros comandos con `slapd`, tales como `slapindex`, que permite regenerar los índices, y `slapcat`, que permite volcar la BD en un archivo en formato LDIF.

1.11.2. Instalación (básica) del servidor

En primer lugar, debemos tener bien configurado el *host* en cuanto al nombre (`hostname` o FQDN, Fully Qualified Domain Name); por ejemplo, en la máquina que se ha utilizado en las pruebas (`debian.nteum.org`), se ha configurado `hostname debian` y en `/etc/hosts` se ha agregado la dirección IP de la máquina `10.0.2.15 debian.nteum.org debian`. También es necesario definir la variable `ServerName debian.nteum.org` en el servidor de Apache (archivo `/etc/apache2/httpd.conf`) para no tener problemas posteriores en la configuración.

El servidor OpenLDAP está disponible en el paquete `slapd`, por lo que para instalarlo se debe hacer `apt-get install slapd dbx.x-util ldap-utils` (esto también instala `dbx.x-util`, que son las utilidades para la base de datos `db` y se deben reemplazar las `x.x` por el número de versión, por ejemplo `db4.2-util`). Durante la instalación, nos pedirá que introduzcamos la contraseña de administrador del servidor `ldap` (por ejemplo `'ldapadmin'`). Como ya hemos comentado, la configuración del servidor LDAP se almacena en el archivo `/etc/ldap/slapd.conf` y se puede editar manualmente o utilizar el asistente de configuración de `slapd`: `dpkg-reconfigure slapd`. El directorio LDAP debe tener una base, a partir de la cual cuelgan el resto de elementos. Como nombre de la base, habitualmente se utiliza el nombre del dominio, por ejemplo, si nuestro dominio es `nteum.org`, la base para nuestro directorio LDAP será `dc=nteum,dc=org`. El *script* de inicialización nos pedirá:

- 1) omitir la configuración (respuesta NO),
- 2) el dominio y lo utilizará para crear el nombre distinguido (DN),
- 3) el nombre de nuestra organización (`nteum`),
- 4) la contraseña para el usuario `admin` (administrador) del servidor LDAP,
- 5) los posibles gestores de datos para almacenar el directorio (es recomendable utilizar el sistema BDB),

Lectura recomendada

Sobre la instalación básica del servidor podéis leer el documento *Redes de área local: Aplicaciones y Servicios Linux* disponible en:
http://www.isftic.mepsyd.es/formacion/materiales/85/cd/REDES_LINUX/index.htm

- 6) si queremos que se elimine la base de datos cuando se elimine slapd (recomendable, NO),
- 7) si existe una base de datos LDAP previa en caso que deseemos moverla (recomendado, SI) y
- 8) si se desea utilizar LDAP versión 2 (recomendado, NO) y ya se tendrá el servidor LDAP configurado.

Para arrancar el servidor LDAP se puede usar `/etc/init.d/slapd restart` y para detenerlo, `/etc/init.d/slapd stop` (el arranque automático se puede hacer con `update-rc.d slapd defaults`).

Como se indicó en el punto anterior, para acceder al directorio LDAP y poder crear y modificar elementos en dicho directorio, hay diferentes formas y herramientas (por ejemplo, `gq`, `phpldapadmin` y `JXplorer`). En esta instalación práctica utilizaremos un explorador de directorios LDAP (LDAP *browser*) como el `phpldapadmin`. Para ello, ejecutamos `apt-get install phpldapadmin`, reiniciamos el servidor Apache `/etc/init.d/apache2 restart` y nos conectamos a través de un navegador a `http://localhost/phpldapadmin/`. Saldrá la página de bienvenida y en la banda izquierda podremos hacer el *login* (con la contraseña del LDAP). Una vez conectado al servidor, seleccionamos la raíz (`dc=nteum,dc=org`) y creamos entre los *templates* que aparecen una *Organizational Unit* (*ou*) a la que llamaremos `users`, repetimos los pasos (opción `create new child` desde la raíz) y creamos otra *ou* llamada `groups`. Ahora solamente nos queda crear los usuarios y los grupos y asignar los usuarios a sus grupos. Dentro de la unidad organizativa `groups` crearemos los grupos `ventas` (`gid=1001`) y `compras` (`gid=1002`) y en la unidad organizativa `users` crearemos los usuarios `juan pirulo` (`uid=1001`, `ventas`, `id=jpirulo`) y `ana pirulo` (`uid=1002`, `compras`, `id=apirulo`). Para ello, dentro de `groups` hacemos `create new child` y seleccionamos `Posix Group`. Creamos los dos grupos indicados pero deberemos modificar el `gid` ya que los asigna por defecto a partir de 1.000 y nosotros los queremos diferentes. Repetimos la operación en `users` seleccionando `User Account`. Aquí nos pedirá los datos de los usuarios (nombre, apellido, grupo, contraseñas, *home directory*, *shell*, etc.) y luego deberemos cambiar el `uid`, ya que los asigna por defecto.

Ahora veremos la autenticación en un sistema LDAP (OpenLDAP), ya que una de las utilidades más importantes de un servidor LDAP es como servidor de autenticación, y veremos las modificaciones que hay que realizar en un sistema Linux para autenticar a los usuarios en lugar de utilizar los archivos `/etc/passwd`, `/etc/group` y `/etc/shadow`. Para ello, es necesario instalar y configurar los paquetes `libpam-ldap` y `libnss-ldap`, donde el primero (`pam-ldap`) permite utilizar el LDAP a las aplicaciones que utilizan PAM, como en el caso de Linux (el archivo de configuración es `/etc/pam_ldap.conf`. Para especificar el modo de autenticación de cada servicio, es necesario configurar los archivos de `/etc/pam.d/`). La segunda biblioteca (`nss-ldap`) permite al servidor LDAP suplantar los archivos `/etc/passwd`, `/etc/group` y `/etc/shadow`

como bases de datos del sistema*. Finalmente, deberemos configurar el archivo `/etc/nsswitch.conf` para que se utilice LDAP como base de datos del sistema en lugar de los archivos `passwd`, `group` y `shadow`. Para instalar `pam-ldap`, se debe ejecutar `apt-get install libpam-ldap` y contestar a todas las preguntas (*host* y raíz del `ldap`, usuario `admin`, contraseña, etc.) y se puede reconfigurar con `dpkg-reconfigure libpam-ldap`. Las opciones se guardan en `/etc/pam_ldap.conf`. En este archivo habrá algunos parámetros que ya estarán configurados, pero debemos agregar los restantes, verificar qué tenemos (en función de lo que ya hemos puesto en el asistente de la instalación):

```
host 192.168.1.239 //nombre o IP del servidor
base dc=nteum,dc=org
ldap_version 3
rootbinddn cn=admin,dc=nteum,dc=org
nss_base_passwd ou=users,dc=nteum,dc=org?one
nss_base_shadow ou=users,dc=nteum,dc=org?one
nss_base_group ou=groups,dc=nteum,dc=org?one
```

Para instalar la biblioteca `libnss-ldap`, ejecutamos la orden `apt-get install libnss-ldap` y se iniciará el asistente de configuración de dicha librería (para reconfigurarla, `dpkg-reconfigure libnss-ldap`). El asistente modificará el archivo `/etc/libnss-ldap.conf`, que es donde se almacena la configuración de la biblioteca, que posteriormente tendremos que editar manualmente para introducir algún cambio que no realiza el asistente. Finalmente, deberemos agregar a este archivo (`/etc/libnss-ldap.conf`) las dos líneas siguientes para indicarle las unidades organizativas:

```
nss_base_passwd ou=users,dc=nteum,dc=org?one
nss_base_group ou=groups,dc=nteum,dc=org?one
```

Para que el servidor LDAP actúe como si se tratara de los archivos `passwd`, `group` y `shadow`, se debe añadir en las líneas que hacen referencia a `passwd`, `group` y `shadow` del archivo `/etc/nsswitch.conf` la palabra `'ldap'` después de la palabra `'compat'` (o después de la palabra `'files'` dependiendo de como esté configurado el `/etc/nsswitch.conf`).

Por último, nos queda editar los archivos que hay en `/etc/pam.d` para configurar la autenticación de cada uno de los servicios. Para no tener que hacer la configuración independientemente, están los archivos que comienzan por `common`, que solo los generales y que los específicos los incluyen (línea `@include`). Los archivos son:

- `/etc/pam.d/common-auth` (para autenticar),
- `/etc/pam.d/common-account` (para disponer de una cuenta),
- `/etc/pam.d/common-session` (para poder iniciar sesión) y
- `/etc/pam.d/common-password` (para poder cambiar contraseña).

*El archivo de configuración es `/etc/libnss-ldap.conf` o `/etc/ldap.conf`, según la versión.

Todos estos archivos contienen una línea que hace referencia a la biblioteca `pam_unix.so`, que corresponde a la autenticación contra los archivos UNIX y debemos agregar las bibliotecas `pam_ldap.so` para autenticar al usuario (las agregaremos encima, así autenticará primero contra el servidor LDAP y después, si la autenticación falla, probará con los archivos UNIX). Para ello, editad los archivos siguientes agregando encima de la librería `pam_unix.so` lo siguiente:

- `/etc/pam.d/common-auth: auth sufficient pam_ldap.so`
- `/etc/pam.d/common-account: account sufficient pam_ldap.so`
- `/etc/pam.d/common-session: session sufficient pam_ldap.so`
- `/etc/pam.d/common-password: password sufficient pam_ldap.so`

Si deseamos que algún servicio se autentifique de forma diferente, podemos editar el archivo del servicio (por ejemplo `/etc/pam.d/su`, `/etc/pam.d/ssh`, `/etc/pam.d/ftp`, etc.), eliminar la línea que comienza por `@include` e introducir la configuración específica que deseemos.

El servidor LDAP ya debería autenticar correctamente, lo que se puede probar con el comando `pamtest`, que se encuentra en el paquete `libpam-dotfile`. Para instalarlo, se debe hacer `apt-get install libpam-dotfile` y podremos probar que el servicio `passwd` (cambiar contraseña) funciona sobre un usuario del directorio LDAP (por ejemplo, `ana`) ejecutando `pamtest passwd apirulo`. Nos deberá permitir cambiar la contraseña. Ejecutad también el comando `finger` sobre usuarios que estén únicamente en el directorio LDAP, (por ejemplo, `juan`) `finger jpirulo`, y nos mostrará toda la información de este usuario. Por ejemplo:

```
debian:# finger apirulo
Login: apirulo      Name: ana pirulo
Directory: /home/users/apirulo  Shell: /bin/sh
Last login Wed Nov 17 04:19 (EST) on tty6
No mail.
No Plan.
```

```
debian:# su - apirulo
apirulo@debian:~$
```

1.12. Servicios de archivos (NFS, *Network File System*)

El sistema NFS permite a un servidor exportar un sistema de archivo para que pueda ser utilizado de forma interactiva desde un cliente. El servicio se compone de un servidor `nfsd` y un cliente (`mountd`) que permiten compartir un sistema de archivo (o parte de él) a través de la red. En Debian, instalad `apt-get install nfs-common portmap` para el cliente, mientras que el servidor necesita `apt-get install nfs-kernel-server nfs-common portmap`. El servidor (en Debian) se pone en marcha a través de los *scripts* `nfscommon` y `nfs-kernel-server` en `/etc/init.d` (y los enlaces adecua-

dos en `/etc/rcX.d`). El servidor utiliza un archivo (`/etc/exports`) para gestionar el acceso remoto a los sistemas de archivo y el control de los mismos. Sobre el cliente (u otro usuario a través de `sudo`), el `root` puede montar el sistema remoto a través del comando:

```
mount Ipserver:directorio-remoto directorio_local
```

y a partir de este momento, el `directorio-remoto` se verá dentro de `directorio local` (este debe existir antes de ejecutar el `mount`). Esta tarea en el cliente se puede automatizar utilizando el archivo de `mount` automático (`/etc/fstab`) incluyendo una línea; por ejemplo:

```
pirulo.remix.com:/usr/local /pub nfs rsize=8192,wzise=8192,timeo=14.
```

Esta sentencia nos indica que se montará el directorio `/usr/local` del *host* `pirulo.remix.com` en el directorio local `/pub`. Además, los dos parámetros `rsize` y `wzise` son los tamaños de bloques de lectura y escritura, `timeo` es el *timeout* de RPC (si no se especifican estos tres valores, se toman los que incluye por defecto). El archivo `/etc/exports` sirve de ACL (lista de control de acceso) de los sistemas de archivo que pueden ser exportados a los clientes. Cada línea contiene un sistema de ficheros (*filesystem*) por exportar seguido de los clientes que lo pueden montar, separados por espacios en blanco. A cada cliente se le puede asociar un conjunto de opciones para modificar el comportamiento (consultad *man exports* para ver un lista detallada de las opciones). Un ejemplo de esto podría ser:

Ejemplo de /etc/exports

```
/          /master(rw) trusty(rw,no_root_squash)
/projects  proj*.local.domain(rw)
/usr       .local.domain(ro) @trusted(rw)
/pub      (ro,insecure,all_squash)
/home     195.12.32.2(rw,no_root_squash) www.first.com(ro)
/user     195.12.32.2/24(ro,insecure)
```

La primera línea exporta el sistema de archivos entero (`/`) a `master` y `trusty` en modo lectura/escritura. Además, para `trusty` no hay *uid squashing* (el `root` del cliente accederá como `root` a los archivos `root` del servidor, es decir, los dos `root` son equivalentes a pesar de ser de máquinas diferentes. Se indica para máquinas sin disco). La segunda y tercera líneas muestran ejemplos de `*` y de *netgroups* (indicados por `@`). La cuarta línea exporta el directorio `/pub` a cualquier máquina del mundo, solo de lectura, permite el acceso de clientes NFS que no utilizan un puerto reservado para el NFS (opción `insecure`) y todo se ejecuta bajo el usuario `nobody` (opción `all_squash`). La quinta línea

especifica un cliente por su IP y en la sexta, se especifica lo mismo pero con máscara de red (/24) y con opciones entre paréntesis sin espacio de separación. Solo puede haber espacios entre los clientes habilitados. Es importante tener en cuenta que existen 3 versiones de NFS (V2, V3 y recientemente V4). Las más comunes son V3 y en algunas instalaciones, V2. Si desde un cliente V3 se conecta a un servidor V2, esta situación se debe indicar con un parámetro.

1.13. Servidor de wiki

Un (o una) **wiki** (del hawaiano *wiki wiki*, “rápido”) es un sitio web colaborativo que puede ser editado por varios usuarios que pueden crear, editar, borrar o modificar el contenido de una página web, de forma interactiva, fácil y rápida; dichas facilidades hacen de una wiki una herramienta eficaz para la escritura colaborativa. La tecnología wiki permite que páginas web alojadas en un servidor público (las páginas wiki) sean escritas de forma colaborativa a través de un navegador, utilizando una notación sencilla para dar formato, crear enlaces, etc. y conservando un historial de cambios que permite recuperar de manera sencilla cualquier estado anterior de la página. Cuando alguien edita una página wiki, sus cambios aparecen inmediatamente en la web, sin pasar por ningún tipo de revisión previa. Wiki también se puede referir a una colección de páginas hipertexto, que cualquier persona puede visitar y editar (definición de Wikipedia). Debian tiene su wiki en <http://wiki.debian.org/> y Ubuntu, en <https://help.ubuntu.com/community/>. Ambas están basadas en **MoinMoin**. MoinMoin es una *Python WikiClone* que permite inicializar rápidamente su propia wiki y solo se necesitan un servidor de web y el lenguaje Python instalado. En la web de MoinMoin se encuentran las instrucciones detalladas para instalar MoinMoin, pero hay dos formas principales de hacerlo: instalación rápida e instalación de servidor.

Enlace de interés

Para saber más sobre MoinMoin podéis visitar su página web en: <http://moinmo.in>. En concreto encontraréis las instrucciones detalladas para instalar MoinMoin en: <http://master19.moinmo.in/InstallDocs>.

1.13.1. Instalación rápida

1) Descargar el paquete desde <http://moinmo.in/MoinMoinDownload> que será, por ejemplo, para la versión 1.9 `moin-1.9.0.tar.gz`. Si se quiere verificar la integridad del paquete, se puede hacer `md5sum moin-x.x.x.tar.gz` y verificar que coincidan el *hash* generado con el que existe en la página de descarga.

2) Desempaquetar MoinMoin `tar xvzf moin-x.x.x.tar.gz`. Esto creará un directorio `moin-x.x.x` en el directorio actual con los archivos en su interior.

3) Dado que MoinMoin está escrita en Python, es necesario utilizar el intérprete de Python:

```
cd moin-x.x.x; python wikiserver.py
```

Esta orden mostrará por pantalla los mensajes de ejecución del servidor. Entre esta información se mostrará la dirección IP sobre la cual está corriendo el servidor, que podrá ser algo como `http://127.0.0.1:8080`. Esta opción usa un servidor web interno, será accesible desde `http://localhost:8080/` y funcionará hasta que se presione `Ctrl-C` en el terminal.

1.13.2. Instalación de servidor

MoinMoin es una aplicación WSGI (*Web Server Gateway Interface*), por lo tanto, el mejor entorno para ejecutar Moin Moin es uno que permita WSGI como, por ejemplo, Apache con `mod_wsgi`. Por ello se recomienda utilizar este método en sistemas con mucha carga. Si todo se ha configurado adecuadamente, se puede utilizar `test.wsgi` para probar que todo funciona correctamente (es recomendable utilizar el *mod_wsgi daemon mode* y no el *embedded mode* ya que es más seguro).

Enlace de interés

Las instrucciones para instalar WSGI para Apache y configurar MoinMoin en este caso se pueden encontrar en la siguiente dirección:
<http://moinmo.in/HowTo/ApacheWithModWSGI>.

Instalación de MoinMoin

Para instalar MoinMoin en un directorio específico (que no es imprescindible, ya que se puede poner en un directorio llamado `MoinMoin/`), se puede utilizar el *script* `setup.py` o simplemente descomprimir el paquete y luego copiar los archivos. Si se elige utilizar `setup.py`, se deberá ejecutar:

```
python setup.py install - -force - -record=install.log - -prefix='/usr/local' - -install-data=/srv
```

O, si prefiere utilizar el directorio por defecto:

```
python setup.py install - -force - -record=install.log
```

Con `- -install-data=/path` se puede cambiar el directorio a `/path` y el `- -force` es importante en la segunda configuración para eliminar cualquier otra configuración.

Instalación de una única wiki

Para configurar MoinMoin asumiremos que el directorio donde lo hemos instalado es `/moin/code` (reemplácese por el adecuado) y el de la configuración, `/moin/config`.

1) El primer punto es configurar el archivo `moin.cgi` (o `moin.wsgi` si utiliza el entorno WSGI) para indicarle a Python dónde están los archivos de MoinMoin (reemplazando el `/path` indicado) y quitar el comentario en el código:

```
sys.path.insert(0, '/moin/code') sys.path.insert(0, '/moin/config').
```


Este archivo será el que luego deberá acceder al servidor web.

2) La siguiente acción es configurar la wiki desde el directorio `/moin/config/`, que se podrá hacer para una sola wiki o para un conjunto de ellas (*farmer*).

3) Copiar el archivo `wiki/config/wikiconfig.py` de la distribución en `/moin/config/`.

4) Editar este archivo que, como se verá, está totalmente comentado para facilitar la configuración. Es importante configurar bien los `/paths` como rutas absolutas y es esencial configurar las siguientes directivas: `data_dir` (donde están los archivos de la instalación), `data_underlay_dir` (MoinMoin viene con todo un sistema de páginas de ayuda y esta directiva apunta donde estén estas páginas), `interwikiname` (identificador para la wiki), `sitename` (el nombre de la wiki). Tened cuidado, ya que `data_dir` contiene información sensible sobre vuestra wiki que no debe ser leída por nadie excepto por el código de la wiki, por ello no permitáis que este directorio sea accesible a través de servidor web y tampoco debéis copiarlo dentro del directorio root de vuestro servidor web.

5) Si no lo habéis instalado, deberéis copiar el contenido de `wiki/data/` de la distribución al directorio especificado en `data_dir` y lo mismo con `wiki/underlay/` al directorio indicado en `data_underlay_dir`.

Instalación de múltiples wikis

Primero debéis copiar `wiki/config/wikifarm/*` de la distribución en el directorio `/moin/config/`. Luego se deben seguir las instrucciones anteriores para cada una de las wikis de la colección (*farm*) teniendo en cuenta que:

1) es necesario tener `data_dir` y `data_underlay_dir` separados para cada wiki,

2) si buscáis que compartan alguna configuración, entonces esta debe estar en `farmconfig.py` y las específicas deben estar en `mywiki.py`.

Instalación de MoinMoin sobre Ubuntu

En este subapartado instalaremos MoinMoin (version 1.9) sobre Ubuntu utilizando el servidor Apache con el módulo WSGI, que es mucho más eficiente que el CGI. Para ello, comenzamos instalando `sudo apt-get install apache2 libapache2-mod-wsgi`. Para utilizar la última versión de MoinMoin no instalaremos el paquete desde el repositorio de Ubuntu sino que explicaremos cómo hacerlo manualmente. Bajamos de <http://moinmo.in/> la última versión (en este caso `moin-1.9.0.tar.gz`) y desde un terminal hacemos `tar xvzf moin-1.9.0.tar.gz`. A continuación pasamos al directorio `cd moin-1.9.0` y ejecutamos `sudo python setup.py install -force -prefix /usr/local -record=install.log`.

Para hacer una prueba sencilla, podemos ejecutar simplemente la instrucción `cd /usr/local/share/moin/server` y hacer `sudo python test.wsgi`, que nos dará un mensaje como este: `Running test application - point your browser at http://localhost:8000/...` A continuación, podemos hacer `http://localhost:8000/` y tendremos la página de prueba WSGI con información de sistema (Ctrl-C para terminar la ejecución del servidor).

A continuación debemos configurar la wiki:

1) Para copiar los archivos de configuración, ejecutamos primero la orden `cd /usr/local/share/moin`; luego `sudo cp server/moin.wsgi` y finalmente `sudo cp config/wikiconfig.py`.

2) A continuación, debemos configurar el servidor Apache (ejecutamos `sudo gedit /etc/apache2/apache2.conf`) agregando al final (y salvado el archivo):

```
# Configuración MoinMoin WSGI
# Invocar la moin wiki en la siguiente url, http://servername/FrontPage:
WSGIScriptAlias /usr/local/share/moin/moin.wsgi

# crear los daemons wsgi daemons - utilizar user/group igual que same el data_dir:
WSGIDaemonProcess moin user=www-data group=www-data processes=5 threads=10 maximum-requests=1000 umask=0007

# utilizar los daemons definidos en la línea anterior
WSGIProcessGroup moin
```

3) Para configurar WSGI, primero tenemos que ejecutar la instrucción `sudo gedit /usr/local/share/moin/moin.wsgi` y luego agregamos al final de la sección *a2* la línea `sys.path.insert(0, '/usr/local/share/moin')`.

4) Agregamos seguridad al entorno con las siguientes cuatro órdenes:

```
cd /usr/local/share;
sudo chown -R www-data:www-data moin;
sudo chmod -R ug+rwX moin;
sudo chmod -R o-rwx moin,
```

lo cual permitirá que solo el *Web server service user* (www-data) pueda modificarlas.

5) A continuación aplicamos los cambios y reiniciamos Apache mediante la instrucción `sudo /etc/init.d/apache2 restart`. Tendremos la wiki activa en `http://localhost/` con la página inicial.

6) Finalmente aplicamos unas mínimas configuraciones adicionales, que son: `sudo gedit /usr/local/share/moin/wikiconfig.py`, quitamos el comentario de `page_front_page = u"FrontPage"` e indicamos el nombre

del administrador, por ejemplo, `superuser = [u"WikiAdmin",]`. Por último, hacemos `sudo /etc/init.d/apache2 restart`. Para configurar el lenguaje, debemos entrar como administrador (WikiAdmin) (si no tenemos un usuario seleccionamos *login* y lo creamos) e ir a la siguiente dirección: `http://localhost/LanguageSetup?action=language_setup`.

1.14. Gestión de copias de respaldo (*backups*)

Existen diversas opciones para hacer copias de respaldo de un conjunto de ordenadores. Una de las más potentes es **Bacula***, que es una colección de herramientas para realizar copias de seguridad en una red. Bacula se basa en una arquitectura cliente/servidor que resulta muy eficaz y fácil de manejar ya que presenta un conjunto muy amplio de características y es eficiente tanto para un conjunto de ordenadores personales como para grandes instalaciones. El paquete está formado por diferentes componentes, entre los más importantes se pueden encontrar:

- **Bacula-director**, *daemon* que gestiona la lógica de los procesos de *backup*;
- **Bacula-storage**, *daemon* encargado de manejar los dispositivos de almacenamiento;
- **Bacula-file**, *daemon* por medio del cual Bacula obtiene los ficheros que necesita para hacer el respaldo y que se deberán instalar en las máquinas fuente de los ficheros a respaldar, y
- **Bacula-console**, que permite interactuar con el servicio de *backup*.

Bacula soporta discos duros, cintas, DVD, USB y también diferentes bases de datos (MySQL, PostgreSQL y SQLite) pero como contrapartida, es necesario disponer de todos los paquetes instalados y su instalación y puesta a punto pueden resultar complejas.

Otro paquete interesante es **BackupPC***, que permite hacer copias de seguridad de disco a disco con una interfaz basada en la web. El servidor se ejecuta en cualquier sistema Linux y admite diferentes protocolos para que los clientes puedan escoger la forma de conectarse al servidor. Este programa no es adecuado como sistema de copia de seguridad de imágenes de disco o particiones ya que no soporta copias de seguridad a nivel de bloque de disco; sin embargo, es muy simple de configurar y la posible intrusión sobre la red de ordenadores en la cual se desea hacer el respaldo es mínima. Este servidor incorpora un cliente *Server Message Block* (SMB) que se puede utilizar para hacer copia de seguridad de recursos compartidos de red de equipos que ejecutan Windows.

Instalación de servidor

Para instalar el servidor [3], primero, ejecutad `apt-get install backuppc smbfs libfile-rsyncp-perl`, que instalará también otros paquetes nece-

*<http://www.bacula.org>

*<http://backuppc.sourceforge.net/info.html>

sarios de Perl y preguntará qué servidor de Apache tenemos instalado y si queremos configurarlo, así como el grupo de SMB. La instalación generará una contraseña para el acceso web*. Si se desea cambiar la contraseña, se puede hacer con `htpasswd /etc/backuppc/htpasswd backuppc`. A continuación, añadid a `/etc/samba/smb.conf` la opción `unix charset = ISO8859-1` y quitad el comentario del parámetro `'- -checksum-seed=32761'`, de las opciones `RsyncArgs` y `RsyncRestoreArgs` del fichero principal de configuración, `/etc/backuppc/config.pl`. Con esto ya podremos acceder a la página web del servidor: `http://localhost/backuppc` y se nos pedirán el usuario y la contraseña antes indicados.

*Recordad apuntar bien el usuario-contraseña, que en nuestro caso es `backupPC` y `Bg0BRd90` como contraseña para la instalación de prueba.

Cliente en las estaciones de trabajo

Las copias de seguridad de las estaciones Windows se realizarán (de acuerdo a la indicaciones de la documentación de Backuppc) a través de las `rsyncd`. Para ello, se deberá instalar el paquete `rsyncd`, que se puede descargar de la misma página de proyecto y seguir las indicaciones de `README.txt`. Se recomienda utilizar los mismos nombre de usuario y contraseña para los `rsyncd`, así como configurar el fichero `c:\rsyncd\rsyncd.secrets` para cada estación de trabajo y, en las opciones `RsyncdPasswd` y `RsyncdUserName`, en el fichero de configuración del servidor. Además, para aplicar las copias de seguridad, se deberá configurar para cada estación el fichero `c:\rsyncd\rsyncd.conf` además de la instalación del servidor `rsync`. Para las estaciones de trabajo Linux, se utilizará también `rsyncd` (otra opción es utilizar `rsync` sobre `ssh`).

Configuración adicional del servidor

En el servidor se debe añadir una línea al fichero `/etc/backuppc/hosts` y un fichero de configuración para cada estación de trabajo en el directorio `/etc/backuppc`. Por ejemplo, para la anterior configuración (nuestro *host* de prueba se llama **nteum**), los ficheros resultantes son:

```
# File: /etc/backuppc/hosts
# ...
#
host dhcp user moreUsers # ← do not edit this line
#farside 0 craig jill,jeff # ← example static IP host entry
#larson 1 bill # ← example DHCP host entry
localhost 0 backuppc
nteum 0 nteum_user

# File: nteum.pl
#
$ConfXferMethod = 'rsyncd';
$ConfRsyncShareName = ['docs','soft','tmp','data'];
```

Estaciones Linux

Para las estaciones Linux solo se tiene que instalar el paquete `rsync` en el cliente y crear los ficheros `/etc/rsyncd.conf` y `/etc/rsyncd.secrets`, como por ejemplo:

```
# File: rsyncd.conf
# rsyncd para el host debianSYS
#
```

```
pid file=/var/run/rsyncd.pid
lock file = /var/lock/rsyncd
read only = yes
list = false
auth users = bkpuser
strict modes = true
secrets file = /etc/rsyncd.secrets
hosts allow = 192.168.1.1
```

```
[root]
comment = home de root
path = /root
```

```
[home]
comment = homes
path = /home
```

```
[etc]
comment = configuraciones
path = /etc
```

```
[var]
comment = datos de programas
path = /var
```

```
# File: rsyncd.secrets
# The format of this file is user:password. You can have as many entries
# as you wish. These accounts are sepecific to the rsync daemon and share
# no relation to Windows local/domain accounts, nor Cywin entries in the
# passwd file.
#
# SECURITY WARNING: Don't use these defaults of UUU for the user name
# and PPP for the password! Change them!!
#
bkpuser:xxxxxxxxx
```

La configuración adicional para el servidor consistiría por un lado en añadir una línea al fichero `/etc/backuppc/hosts` y por otro en crear el fichero `/etc/backuppc/debianSYS.pl`:

```
# File debianSYS.pl
$ConfXferMethod = 'rsyncd';
$ConfRsyncShareName = ['etc','home','var','root'];
```

Es importante que el servidor pueda encontrar el nombre de la máquina y es por ello que se recomienda poner una entrada `ip host` en `/etc/hosts`. Por último, se debe ejecutar el *daemon* `rsync` en el cliente y para ello existen dos opciones: *standalone* o a través de `inetd`. En este caso, se escogerá la primera opción y se deberá cambiar la configuración en `/etc/default/rsync` para que el *daemon* esté activo por defecto al ejecutar `dpkg-reconfigure rsync` (hay que reconfigurar el servicio).

Creación de archivos

Si bien este servicio creará copias de seguridad, solo es un servidor y en entornos críticos es aconsejable hacer copias de las copias externas al servidor en soporte DVD, USB, etc. BackupPC proporciona un mecanismo para generar archivos bajo demanda y su mecanismo es similar a configurar un nuevo *host*, pero indicando como método la palabra *archive*. Por ejemplo, en nuestro caso tenemos el fichero `/etc/backuppc/archive.pl` para preparar copias de seguridad externas en DVD:

```
# File: etc/backuppc/archive.pl
#
# archivo de backups externo (en DVD)
#
$ConfXferMethod = 'archive';
```

Como se puede observar, es simple, ya que solo se deben modificar las opciones generales del archivo que están en `/etc/backuppc/config.pl`, sección *Archive* (solo se muestran las partes más interesantes):

```
# ...
# Archive Destination. The Destination of the archive
# e.g. /tmp for file archive or /dev/nst0 for device archive
$ConfArchiveDest = '/var/lib/backuppc/archives';

# Archive Compression type
$ConfArchiveComp = 'gzip';

# Archive Parity Files
$ConfArchivePar = 0;

# Archive Size Split
$ConfArchiveSplit = 4500;

# Archive Command
$ConfArchiveClientCmd = '$Installdir/bin/BackupPC_archiveHost'
. ' $tarCreatePath $splitpath $parpath $host $backupnumber'
. ' $compression $compext $splitsize $archiveloc $parfile *';
```

Actividades

1. Configurad un servidor DNS como caché y con un dominio propio.
2. Configurad un servidor/cliente NIS con dos máquinas exportando los directorios de usuario del servidor por NFS.
3. Configurad un servidor SSH para acceder desde otra máquina sin contraseña.
4. Configurad un servidor Apache+ SSL+ PHP+ MySQL+ PHPAdmin para visualizar las hojas personales de los usuarios.
5. Cread y configurad un sistema de correo electrónico a través de Exim, Fetchmail, SpamAssassin y un servidor IMAP para recibir correos desde el exterior y poder leerlos desde una máquina remota con el cliente Mozilla (Thunderbird).
6. Instalad la Wiki MoinMoin y cread un conjunto de páginas para verificar su funcionamiento.
7. Instalad el servidor de *backups* BackupPC y generad una copia de respaldo desde una máquina Windows y otra desde una máquina Linux. Escoged el método de comunicación con los clientes y justificar la respuesta.

Bibliografía

- [1] *Apache2 + SSL*.
<<http://www.debian-administration.org/articles/349>>
- [2] *Apache2 + WebDav*.
<<http://www.debian-administration.org/articles/285>>
- [3] **Baila, S.** (2005). *Instalación de BackupPC*.
<<http://fitxers.sargue.net/fitxers/diarioBackupPC.html>>
- [4] *Comparison of FTP client software*
<http://en.wikipedia.org/wiki/Comparison_of_FTP_client_software>
- [5] **Debian.org**. *Debian Home*.
<<http://www.debian.org>>
- [6] *Exim*.
<<http://www.exim.org/docs.html>>
- [7] **IETF**. Repositorio de Request For Comment desarrollados por Internet Engineering Task Force (IETF) en el Network Information Center (NIC). <<http://www.ietf.org/rfc.html>>
- [8] **Instituto de Tecnologías Educativas**. *Redes de área local: Aplicaciones y Servicios Linux*.
<http://www.isftic.mepsyd.es/formacion/materiales/85/cd/REDES_LINUX/index.htm>
- [9] **Kiracofe, D.** *Transparent Proxy with Linux and Squid mini-HOWTO*.
<<http://tldp.org/HOWTO/TransparentProxy-1.html>>
- [10] **Kukuk, T.** *The Linux NIS(YP)/NYS/NIS+ HOWTO*.
<<http://tldp.org/HOWTO/NIS-HOWTO/verification.html>>
- [11] **Langfeldt, N.** *DNS HOWTO*.
<<http://tldp.org/HOWTO/DNS-HOWTO-7.html>>
- [12] *LWP*.
<http://www.lawebdelprogramador.com/temas/tema_stablephpapachemysql.php>
- [13] *MoinMoin*.
<<http://moinmo.in/>>
- [14] *MoinMoin + Ubuntu*.
<<http://moinmo.in/HowTo/UbuntuQuick>>
- [15] **Mourani, G.** (2001). *Securing and Optimizing Linux: The Ultimate Solution*. Open Network Architecture, Inc.

- [16] *Mutt*.
<<http://www.mutt.org>>
- [17] *NIS HOWTO*.
<<http://www.linux-nis.org/doc/nis.debian.howto>>
- [18] **Pinheiro Malère, L. E.** (2007). *Ldap. The Linux Documentation Project*.
<<http://tldp.org/HOWTO/LDAP-HOWTO/>>
- [19] *ProcMail*.
<<http://www.debian-administration.org/articles/242>>
- [20] *Proftpd*.
<<http://www.debian-administration.org/articles/228>>
- [21] *PureFtpd*.
<<http://www.debian-administration.org/articles/383>>
- [22] *Squid proxy server, Proxy Cache*.
<<http://www.squid-cache.org/>>

