Guía de Administración de Redes con Linux

Contenidos

Contenidos

Contenidos

Contenidos

Contenidos

Contenidos

Contenidos

Contenidos

Contenidos

Contenidos

Guía de Administración de Redes con Linux

Olaf Kirch

Terry Dawson

Editado por

Tabla de contenidos

Lista de tablas

Tabla de figuras

Tabla de ejemplos

Prefacio

En muchos países, Internet es ya un término doméstico. Mientras la gente comienza a pensar en las superautopistas de la información, los ordenadores interconectados son ya algo cotidiano, como lo son los televisores o los hornos microondas. De alguna manera se está desarrollando la Cultura de Internet: todo el mundo habla de ella.

Por supuesto, las redes han existido desde mucho antes. La conexión de ordenadores para formar una red local viene siendo habitual desde hace mucho, bien para pequeñas instalaciones, o bien para grandes empresas que intercomunican sus redes locales usando líneas de comunicación proporcionadas por compañías telefónicas. Internet se ha convertido en una opción interesante cuando una empresa pequeña desea conectar allí un nodo de correo y noticias, por ejemplo, ofreciendo acceso por RDSI o RTC. Con la popularización de las líneas DSL (Líneas Digitales de Abonado) y los módem−cable, estas situaciones se han facilitado muchísimo.

Hablar de redes de ordenadores siempre implica hablar de Unix. Por supuesto, Unix no es el único sistema operativo con capacidad para conectarse a las redes, pero ha sido la opción elegida durante años para las conexiones de empresas, y seguirá siéndolo durante mucho tiempo.

Lo que hace a Unix especialmente interesante para los usuarios particulares es que hay mucha actividad en la producción de sistemas operativos libres compatibles Unix para PC, como sucede con 386BSD, FreeBSD y GNU/Linux.

GNU/Linux es un clon de Unix libremente distribuible para computadores personales. Actualmente corre en una amplia variedad de máquinas, que incluye no solo la familia Intel, sino también las máquinas basadas en arquitecturas Motorola 680x0, como el Commodore Amiga o el Apple Macintosh; también están soportadas las máquinas Sun SPARC y Ultra−SPARC, las Compaq Alpha, MIPS, PowerPCs (como sucede con los Mac de última generación) e incluso StrongARM (usada en dispositivos de mano o *handhelds*). También tenemos reescritura de Linux en plataformas más «oscuras», como el IBM S/390 o los Fujitsu AP−1000. Y parece que esta tendencia no va a decaer.

Linux fue desarrollado por un gran equipo de voluntarios a través de Internet. Fue iniciado en 1990 por el estudiante finlandés Linus Torvalds, como un proyecto universitario. Desde entonces, ha ido creciendo hacia un sistema compatible−unix muy completo, y muy utilizado para todo tipo de aplicaciones, desde simples procesadores de texto hasta avanzados sistemas de reconocimiento de voz, sin contar con que es la plataforma perfecta para acceder a Internet. Además soporta una gran cantidad de hardware, y tiene una pila TCP/IP completa, con adición de software cortafuegos, implementaciones PPP, SLIP, IPX; e incluso protocolos que otros sistemas no suelen incluir. Linux es potente, rápido y libre. Y su popularidad sigue aumentando.

El sistema operativo GNU/Linux está cubierto por la Licencia Pública General GNU, que es la misma que rige el software desarrollado por la Free Software Foundation. Esta licencia permite a cualquiera modificar y/o redistribuir el software (gratuitamente o a cambio de dinero), «contagiándose» de dicha licencia cualquier modificación posterior. El término inglés «free» se refiere a la libertad, no a su precio, no necesariamente gratuito.

1. Propósito y Audiencia de este libro

Este libro lo he escrito para proporcionar un manual de referencia para administrar redes en entornos Linux. Será útil tanto a los principiantes como a los usuarios avanzados, que encontrarán en él lo que necesitan para realizar muchas de sus labores de administración y configuración en red. No obstante, este libro no lo incluye todo, ya que era imposible. Pero hemos intentado cubrir lo más importante. La gente nueva en Linux lo encontrará útil para conocer los pasos para montar y poner en marcha una red basada en sistemas Linux.

Hay otros muchos libros o fuentes de información desde donde podemos aprender cualquiera de los temas cubiertos aquí (salvo, quizás, algunos de los temas más específicos de Linux, como el soporte de tarjetas de red). Hemos preparado una lista bibliográfica para cuando se desee profundizar más en cualquiera de las áreas tratadas.

2. Fuentes de información

Si somos nuevos en el mundo de Linux, tenemos algunos recursos que explorar para familiarizarnos con este sistema. El contar para ello con una conexión a Internet será útil, pero no imprescindible.

Guías del Proyecto de Documentación de Linux

El Proyecto de Documentación de Linux es un grupo de voluntarios que han trabajado en preparar libros (guías), documentos cortos (HOWTO), páginas de manual, etc; abarcando temas desde la instalación hasta la programación del núcleo. En España, el grupo LuCAS[1] está traduciendo estas guías y publicando muchos otros documentos. Algunos trabajos del LDP son:

Linux Installation and Getting Started

Por Matt Welsh, y otros. Este libro cuenta cómo obtener, instalar y utilizar Linux. Incluye un tutorial de introducción a Unix e información sobre administracion, X−Window y redes.

El grupo LuCAS tradujo este manual hace algunos años, con el nombre de «Linux Instalación y Primeros Pasos».

Linux System Administrators Guide

De Lars Wirzenius y Joanna Oja. Este libro es un manual general de administración de Linux, cubriendo cuestiones como creación de usuarios, realización de copias de respaldo, configuración de paquetes de aplicación, actualización del sistema, etc.

Linux System Administration Made Easy

De Steve Frampton. Es un libro que cuenta de forma sencilla las tareas más habituales de administración que deben realizar todos los poseedores de una estación de trabajo Linux.

Linux Programmers Guide

De B. Scott Burkett, Sven Goldt, John D. Harper, Sven van der Meer, y Matt Welsh. Este libro cubre temas de interés para aquellos que quieren desarrollar aplicaciones con Linux.

El grupo LuCAS tradujo este manual hace algún tiempo, con el nombre de «Guía Linux de Programación».

The Linux Kernel

De David A. Rusling. Es un manual que introduce al mundo del núcleo Linux. Cómo se estructura, el proceso de arranque del sistema operativo, etc.

Este manual se encuentra parcialmente traducido al español por el grupo LuCAS, con el nombre «Guía del Núcleo».

The Linux Kernel Module Programming Guide

De Ori Pomerantz. Esta guía explica cómo escribir módulos para el núcleo Linux.

Hay más manuales en desarrollo. Para más información acerca del LDP lo más conveniente es consultar su servidor web en http://www.linuxdoc.org/ o una de sus muchas réplicas.

Documentos HOWTO

Los HOWTOs de Linux (CÓMOs, en español) son artículos más o menos breves que detallan de manera específica alguna cosa en Linux, como la configuración de X−Window, cómo configurar la red, etc. Son documentos localizados en el directorio `HOWTO` de los sitios FTP listados más adelante, y disponibles por web en las réplicas del LDP. Muchos HOWTOs están traducidos al español, y están también disponibles a través del Proyecto LuCAS y la página web INSFLUG[2]. Véase la bibliografía del final o el fichero `HOWTO−INDEX` para conocer una lista de HOWTOs disponibles en inglés.

Será interesante que obtengamos el *Installation HOWTO*, que describe cómo instalar Linux en una máquina; el *Hardware Compatibility HOWTO*, que contiene una lista de hardware que se sabe que funciona correctamente en Linux; y el *Distribution HOWTO*, que lista empresas que empaquetan y venden Linux en disquetes o CD−ROM.

La bibliografía de este libro incluye referencias a los CÓMOs relacionados con el uso de redes en Linux.

Preguntas comunes en Linux

Las *Preguntas comunes con respuestas* (comúnmente conocidas como FAQs) contienen colecciones amplias de típicas preguntas que se hacen los usuarios, junto a las correspondientes respuestas. Son documentos de lectura obligada para todos los nuevos usuarios.

2.1. Documentación disponible por FTP

Si tenemos acceso al FTP anónimo, podemos obtener toda la documentación de Linux listada desde varios sitios, incluyendo metalab.unc.edu:/pub/Linux/docs y tsx−11.mit.edu:/pub/linux/docs. Por supuesto, el Grupo LuCAS tiene su servidor FTP con documentación en español, en la dirección lucas.hispalinux.es:/pub/LuCAS.

Estos sitios tienen también réplicas por todo el mundo. Concretamente, las réplicas oficiales del grupo LuCAS están listadas en su página web, http://lucas.hispalinux.es/.

2.2. Documentación disponible por web

Hay muchos sitios web con cosas para Linux. El sitio principal del LDP está en http://www.linuxdoc.org/.

The Open Source Writers Guild (OSWG) (en español se llamaría Gremio de Autores de Fuente Abierta) es un proyecto que tiene un alcance que excede a este libro. El OSWG, como este libro, está destinado a promover y facilitar la producción de documentación para software libre y «OpenSource». Su sitio principal está en http://www.oswg.org:8080/oswg.

Todos los sitios contienen documentos relacionados con Linux en hipertexto y otros formatos.

2.3. Documentación comercial

Algunas editoriales y productoras de software han publicado en papel los trabajos del LDP. Dos de estas empresas son:

Specialized Systems Consultants, Inc. (SSC)
http://www.ssc.com/
P.O. Box 55549 Seattle, WA 98155−0549
1−206−782−7733
1−206−782−7191 (FAX)
sales@ssc.com

y:

Linux Systems Labs
http://www.lsl.com/
18300 Tara Drive
Clinton Township, MI 48036
1−810−987−8807
1−810−987−3562 (FAX)
sales@lsl.com

Ambas empresas venden compendios de CÓMOs de Linux (en inglés) y otros libros impresos.

O'Reilly & Associates publica regularmente libros sobre Linux. A veces son libros del LDP mejorados e impresos en papel. Algunos de los manuales publicados son:

Running Linux
: Guía de instalación y de usuario, cómo hacer computación personal con Linux.

Learning Debian GNU/Linux, Learning Red Hat Linux
: Más básicos que *Running Linux*, estos libros contienen las populares distribuciones en CD−ROM e incluyen instrucciones precisas para instalarlas correctamente.

Linux in a Nutshell
: Es otro libro de la exitosa serie "in a Nutshell". En este caso se trata de una referencia amplia sobre Linux.

2.4. Linux Journal y Linux Magazine

Linux Journal y *Linux Magazine* son revistas mensuales para la comunidad Linux, escritas y publicadas por varios activistas de Linux. Contienen artículos que van desde las preguntas de novatos hasta avanzados artículos técnicos. Aun teniendo cosas como Usenet, suscribirse a una de estas revistas es muy conveniente para mantenerse al día en lo que se cuece en la comunidad Linux.

Linux Journal es la revista más veterana y está siendo publicada por S.S.C. Incorporated, cuyas señas ya hemos escrito. También podemos consultarla en su página web, en la dirección http://www.linuxjournal.com/.

Linux Magazine es una publicación independiente de más reciente creación. Su sitio web está en http://www.linuxmagazine.com/.

2.5. Grupos en Usenet sobre Linux

Si tenemos acceso a Usenet, nos interesarán los siguientes grupos relacionados con Linux:

comp.os.linux.announce
: Es un grupo moderado que contiene anuncios de software nuevo, distribuciones, informes de errores y otros. Todos los usuarios de Linux deben leer este grupo. Las nuevas noticias deben enviarse a linux−announce@news.ornl.gov.

comp.os.linux.help
: Preguntas generales y sus respuestas acerca de la instalación o el uso de Linux.

comp.os.linux.admin
: Discusiones de interés para los administradores de sistemas Linux.

comp.os.linux.networking
: Discusiones relativas a las redes con Linux.

comp.os.linux.development
: Discusiones acerca del desarrollo del núcleo Linux o el sistema.

comp.os.linux.misc
: Grupo donde caben discusiones no relacionadas con los grupos anteriores.

También hay distintos grupos de noticias sobre Linux en otros idiomas, como ocurre con fr.comp.os.linux en francés, de.comp.os.linux en alemán y la jerarquía es.comp.os.linux.* en español.

2.6. Listas de correo sobre linux

Hay un gran número de listas especializadas en las que encontraremos gente con inquietudes similares a las nuestras y que con frecuencia podrán ayudarnos a resolver nuestros problemas.

Las más conocidas son las que se alojan en la Universidad Rutgers. Podemos suscribirnos a ellas enviando un mensaje con un formato similar al siguiente:

```
To: majordomo@vger.rutgers.edu
Subject: cualquier cosa
Body:

subscribe lista
```

Alguna de las listas relacionadas con redes son:

linux−net
: Discusiones relativas a las redes con Linux

linux−ppp
: Discusiones relativas al soporte PPP en Linux

linux−kernel
: Discusiones relativas al desarrollo del núcleo Linux.

También hay listas en español sobre Linux. Que sepamos ninguna está orientada específicamente a redes. Una de las listas más populares es l−linux@calvo.teleco.ulpgc.es. Para suscribirse a ella hay que enviar un mensaje similar al ejemplo anterior, pero a la dirección majordomo@calvo.teleco.ulpgc.es.

2.7. Soporte en línea

Hay diversas formas de obtener ayuda en línea, de voluntarios de todo el mundo que ofrecen sus servicios para asistir a los usuarios con problemas o preguntas.

La red IRC OpenProjects está enteramente dedicada a proyectos abiertos (como el software libre o el hardware abierto). Algunos de sus canales están pensados para soporte en línea a los usuarios de Linux. IRC significa "Internet Relay Chat"[3] y es un servicio de la red que permite hablar interactivamente con varias personas a la vez. Las redes IRC soportan los canales para agrupar las charlas de cada grupo. Todo lo que tecleemos en un canal será visible a todos los usuarios del mismo.

Hay un número de canales que están activos siempre y en ellos encontraremos cualquier día y a cualquier hora alguien que nos podrá ayudar a resolver un determinado problema. Podemos usar este servicio instalando un cliente IRC como *irc−II*, teniéndonos que conectar al servidor irc.openprojects.org:6667, y luego entrar en el canal `#linpeople`.

En general en todos los canales se habla inglés, no obstante miembros de la comunidad Linux hispanohablante están migrando desde otras redes a OpenProjects y están comenzando a ayudar a otros usuarios hispanohablantes.

2.8. Grupos de usuarios

Hay por todo el mundo muchos grupos de usuarios que ofrecen ayuda desinteresada a los usuarios nuevos. Muchos de estos grupos organizan actividades como «jornadas de instalaciones», conferencias, demostraciones, «parties» y otros eventos sociales. Encontrar un grupo local es una forma de conocer gente con inquietudes parecidas a las nuestras. Hay listas de grupos de usuarios publicadas, algunas de las más importantes son:

Groups of Linux Users Everywhere
: http://www.ssc.com/glue/groups/

LUG list project
: http://www.nllgg.nl/lugww/

LUG registry
: http://www.linux.org/users/

En el mundo hispanohablante también hay muchos grupos de usuarios. La asociación Hispalinux[4] es el grupo principal en España. Dicha asociación mantiene una página en Internet que contiene una lista actualizada de grupos locales, diseminados por todo el país. La página se llama http://grupos−locales.hispalinux.es.

2.9. Obtención de Linux

El software de Linux se puede conseguir con muchas distribuciones, como Debian, Redhat, Caldera, Corel, SuSE y Slackware. Cada distribución contiene todo lo que necesitamos para instalar un sistema Linux completo: el núcleo, utilidades básicas, bibliotecas, ficheros de soporte y aplicaciones.

Las distribuciones de Linux pueden obtenerse a través de diferentes fuentes en línea, como Internet. Cada una de ellas suele tener su sitio de distribución por FTP y un sitio web. Algunos son:

Caldera
: http://www.caldera.com/ftp://ftp.caldera.com/

Corel
: http://www.corel.com/ftp://ftp.corel.com/

Debian
: http://www.debian.org/ftp://ftp.debian.org/

RedHat

http://www.redhat.com/ftp://ftp.redhat.com/

Slackware

http://www.slackware.com/ftp://ftp.slackware.com/

SuSE

http://www.suse.com/ftp://ftp.suse.com/

Hispafuentes (totalmente en español)

http://www.hispafuentes.com/

ESWare (totalmente en español)

http://www.esware.com/ftp://ftp.esware.com/

Muchos sitios FTP populares también incluyen réplicas de las distribuciones más populares. Los sitios FTP más conocidos son:

metalab.unc.edu:/pub/Linux/distributions/
ftp.funet.fi:/pub/Linux/mirrors/
tsx−11.mit.edu:/pub/linux/distributions/
mirror.aarnet.edu.au:/pub/linux/distributions/

Muchas distribuciones modernas pueden instalarse directamente desde Internet. Eso implica bajarse gran cantidad de programas, luego no nos interesará si no tenemos una conexión permanente o de alta velocidad, salvo que queramos simplemente actualizar nuestra instalación actual[5]

Linux puede ser adquirido en CD−ROM de cualquiera de los cada vez más frecuentes distribuidores. Si nuestra tienda más cercana no tiene CD−ROMs con Linux, lo mejor es pedirles que los traigan. En todo caso siempre podemos pedirlos por correo. Algunos fabricantes hacen lotes que traen varias distribuciones, quizás nos interese para probarlas y poder elegir con cuál vamos a trabajar.

3. Estándares de Sistemas de Ficheros

En el pasado, uno de los problemas que aquejaban las distribuciones de Linux, así como los paquetes separados, era que no había un único sistema de ficheros aceptado. Esto generaba incompatibilidades entre paquetes diferentes, y enfrentaba a usuarios y administradores con la tarea de localizar varios programas y ficheros.

Para mejorar esta situación, en Agosto de 1993, varias personas formaron el Grupo del Estándar de Sistema de Ficheros de Linux, o Grupo FSSTND para abreviar, coordinado por Daniel Quinlan. Después de seis meses de discusión, el grupo presentó un diseño que muestra una estructura de sistema de ficheros coherente y define la localización de los programas más esenciales y ficheros de configuración.

Este estándar se supone que se ha implementado en la gran mayoría de distribuciones y paquetes de Linux. A lo largo de este libro, además, asumiremos que todos los ficheros que se traten residen en el lugar especificado por este estándar; sólo donde haya una larga tradición que choque con esta especificación se mencionarán emplazamientos alternativos.

El estándar FSSTND ha sido reemplazado en 1997 por el estándar FHS de jerarquía de ficheros. El estándar FHS considera cuestiones propias de arquitecturas heterogéneas que FSSTND no tiene en cuenta. El texto de este estándar puede obtenerse en el directorio de documentación de Linux de casi cualquier servidor FTP dedicado a Linux o sus réplicas; y su sitio principal es http://www.pathname.com/fhs/. Daniel Quinlan, coordinador del grupo FHS, puede ser localizado en quinlan@transmeta.com.

4. Estándar del Sistema Básico Linux

El amplio repertorio de distribuciones Linux, cada una proporcionando diversas ventajas sobre otras, han contribuido a crear problemas nuevos a los desarrolladores de softwareparticularmente a los desarrolladores de software propietario.

Cada distribución empaqueta y proporciona ciertas bibliotecas básicas, utilidades de configuración, aplicaciones del sistema y ficheros de configuración. Sin embargo, hay diferencias entre versiones, nombres y localizaciones que hace difícil saber cuáles tenemos en cada distribución. Esto hace complicado incluir aplicaciones precompiladas que funcionen bien en todas las distribuciones.

Para intentar solucionar este problema, se ha creado un nuevo proyecto llamado «Linux Standard Base» (Estándar de Sistema Linux Básico). Intenta describir un estándar de ficheros básicos que debe cumplir toda distribución. Así, si un fabricante desea desarrollar una aplicación que funcione en muchas distribuciones sin problemas, basta tener en cuenta la ubicación y versiones de ficheros que el estándar propone.

Podemos encontrar más información sobre el estado de este proyecto en su servidor principal, en http://www.linuxbase.org/.

Si nos preocupa la interoperatividad, en especial si somos fabricantes de software comercial, debemos asegurarnos de que la distribución en la que desarrollemos esté haciendo un esfuerzo para estandarizarla.

5. Acerca de este Libro

Cuando Olaf se unió al Proyecto de Documentación de Linux en 1992, escribió dos pequeños capitulos, sobre UUCP y **smail**, con el objeto de contribuir a la incipiente guía de administración del sistema. El desarrollo de la pila TCP/IP acababa de comenzar, y aquellos «pequeños capítulos» empezaban a pedir que se completaran en una más amplia Guía de Redes. «¡Perfecto!», pensaron muchos. «¡Hagámoslo!». De este modo, Olaf preparó la primera versión de la Guía de Administración de Redes Linux y la presentó en septiembre de 1993.

Olaf continuó trabajando en la Guía de Redes y alguna vez sacó versiones mejoradas. Vince Skahan contribuyó con el capítulo sobre **sendmail**, que fue reemplazado debido a la nueva interfaz de configuración desarrollada.

La edición que está usted leyendo es una revisión y actualización propuesta por O'Reilly & Associates y supervisada por Terry Dawson.[6] Terry ha sido radioaficionado durante más de 20 años y ha trabajado en la industria de telecomunicaciones durante más de 15. Fue coautor de la FAQ de redes y desde entonces ha mantenido varios documentos cortos (HOWTOs) sobre diversos temas de redes. Terry siempre ha sido un entusiasta participante en el proyecto de la Guía de Administración de Redes, y ha añadido algunos capítulos más a esta versión para describir características de trabajo en red de Linux que se han ido creando con el tiempo. También ha escrito actualizaciones para el resto de los capítulos.

El capítulo sobre **exim** fue escrito por Philip Hazel,[7] quien además es el desarrollador principal y responsable del mantenimiento de este paquete.

El libro está organizado intentando reproducir la secuencia de pasos que hay que seguir para configurar la red. Se inicia mostrando conceptos básicos de redes, particularmente las basadas en TCP/IP. A continuación se trata la configuración de bajo nivel, es decir, desde configuración de tarjetas de red hasta software de bajo nivel como cortafuegos o NAT (enmascaramiento). Después se trata la configuración de alto nivel, como servicios **rlogin**, sistemas de ficheros en red (NFS) y servicio de información NIS. Seguidamente se cuenta cómo configurar un nodo UUCP. El resto de capítulos se dedica a tratar aplicaciones que corren por encima, como correo electrónico o las noticias. También hay un capítulo especial para el protocolo IPX y el sistema de ficheros NCP, debido a que se usan en muchos entornos corporativos.

La parte de correo electrónico incluye una introducción a partes internas del transporte y encaminamiento de correo, y la variedad de formatos de dirección existentes. Se cuenta cómo se configura **exim**, un agente de transporte ideal para utilizar en casi todos los sitios, y también hay un capítulo dedicado a **sendmail**, otro agente de transporte más adecuado para redes que usen TCP/IP y UUCP.

En los capítulos dedicados a noticias nos encontraremos explicaciones para el uso de Usenet. Se incluyen capítulos dedicados a INN y C−News, los servidores de noticias más usados, y el protocolo NNTP para dar acceso de lectura a clientes en la red local. Se finaliza con un capítulo dedicado a los programas de lectura de noticias más populares.

Por supuesto, en un libro nunca podemos contestar todas las preguntas que podamos tener. Por tanto, si se siguen las instrucciones del libro y algo falla, es mejor tener algo de paciencia. Algunos problemas pueden deberse a errores por nuestra culpa (véase la sección Sección 9, al final de este prólogo) aunque también pueden deberse a cambios en el software de red. Por tanto, deben verificarse las fuentes de información mostradas. Además, no estamos solos por lo que la solución a nuestros problemas muchas veces se encuentra preguntando. Si tenemos la oportunidad, tal vez debamos obtener la última versión del núcleo o del software que nos interesa para ver si funciona. Muchos problemas proceden de haber usado software en desarrollo, después de todo Linux es un sistema en continua mejora.

6. La Versión Oficial Impresa

En el otoño de 1993, Andy Oram, quien ha estado en la lista de correo de LDP desde casi el principio de todo, me pidió publicar mi libro en O'Reilly and Associates. Me puse nervioso; jamás había imaginado que mi libro tuviese tanto éxito. Finalmente acordamos que O'Reilly produjese una Versión Impresa Oficial mejorada de la Guía de Red, mientras yo conservaba el copyright original de forma que las fuentes del libro pudieran ser distribuidas libremente. Esto significa que Ud. puede elegir libremente: puede conseguir las fuentes distribuidas en la red (o las versiones preformateadas en DVI o PostScript, para cada caso), e imprimirlas. O puede comprar la versión impresa oficial de O'Reilly, que ya está disponible.

Entonces, ¿por qué íbamos a pagar dinero por algo que puede conseguir gratis? ¿Está loco Tim O'Reilly por publicar algo que todos pueden imprimir e incluso vender por sí mismos?[8] ¿O hay alguna diferencia entre estas versiones?

Las respuestas son «depende», «no, definitivamente no» y «sí y no». O'Reilly & Associates asume un riesgo al publicar la Guía de Red, pero espero que finalmente merezca la pena. Si es así, creo que este proyecto puede servir como ejemplo de cómo el mundo del software libre y las compañías pueden cooperar para producir algo que beneficia a los dos. Desde mi punto de vista, el gran servicio que está dando a la comunidad Linux (aparte del libro que tiene disponible en su librería) es que puede ayudar a que Linux sea reconocido como algo que puede ser tomado en serio: una alternativa útil y viable a los sistemas operativos UNIX de PC comerciales.

¿Por qué lo publican? La razón es que lo ven como su tipo de libro. Es lo que esperarían producir si acordasen con un autor escribir sobre Linux. El desarrollo, el nivel de detalle y el estilo concuerdan con sus otras ofertas.

El detalle de la licencia LDP es para asegurarse de que nadie se queda fuera. Otra gente puede imprimir copias de este libro, y nadie le perseguirá si Ud. se hace con una de esas copias. Pero si no ha tenido oportunidad de ver la versión de O'Reilly, intente conseguirlo en una librería o vea la copia de un amigo. Pensamos que le gustará lo que vea, y querrá comprárselo para Ud. mismo.

Entonces, ¿qué hay de las diferencias entre la versión impresa y la versión en línea? Andy Oram se ha esforzado enormemente en transformar mis primeros pasos en algo que merezca la pena imprimirse (él también ha estado revisando los otros libros salidos del Proyecto de Documentación de Linux, intentando contribuir con sus habilidades profesionales a la comunidad Linux).

Desde que Andy empezó a revisar la Guía de Red y a editar las copias que yo le mandaba, el libro ha mejorado enormemente frente a lo que era hace medio año. No estaría tan cerca de lo que es ahora sin su contribución.

Por tanto, la versión O'Reilly *se mantendrá diferente*. Será profesionalmente maquetada, y aunque en general tendremos buena calidad con la versión libre, probablemente nos merecerá la pena adquirir el libro a pesar del precio. Además, las figuras de la versión impresión impresa tienen mejor aspecto que las que mostramos aquí. Los índices están también mejorados.

7. Revisión del libro

Capítulo 1 trata de la historia de Linux y cubre información básica sobre UUCP, TCP/IP y nociones básicas sobre sus protocolos. Los siguientes capítulos tratan de la configuración de Linux para algunas aplicaciones de red. Examinaremos IP un poco más de cerca en Capítulo 2, antes de entrar en la edición de los ficheros de configuración. Si sabemos cómo funciona el encaminado de IP o la resolución de direcciones, podemos saltarnos este capítulo.

Capítulo 3 trata de aspectos muy básicos de configuración, como la construcción del núcleo y la configuración de la tarjeta de red. La configuración de los puertos serie se trata aparte en Capítulo 4, ya que estos conceptos son de utilidad cuando hablamos de UUCP.

Capítulo 5, nos ayudará a preparar la máquina para redes TCP/IP. Contiene trucos de instalación para nodos aislados con red en «loopback», y también para nodos conectados a una ethernet. También nos introduce a algunas utilidades que podemos usar para comprobar y depurar la instalación. Capítulo 6, muestra cómo configurar la resolución de nombres y explica cómo poner en marcha un servidor de nombres.

Capítulo 7, muestra cómo establecer conexiones SLIP y proporciona una referencia detallada para **dip**, una utilidad que permite automatizar buena parte de los pasos necesarios. Capítulo 8, cubre PPP y **pppd**, el demonio de PPP.

Capítulo 9, extiende nuestra discusión a la seguridad de la red, y describe cómo es el cortafuegos IP de Linux y sus utilidades de configuración: **ipfwadm**, **ipchains**, e **iptables**. El cortafuegos IP proporciona un mecanismo para controlar quién tiene acceso a nuestra red y a cada uno de nuestros nodos.

Capítulo 10, muestra cómo configurar el sistema de contabilidad IP de Linux, que permite monitorizar el tráfico entrante y saliente de la red.

Capítulo 11, cubre una característica del sistema de red de Linux que es el enmascaramiento IP, que permite a toda una red local acceder a través de una sola dirección IP pública a Internet, ocultando los sistemas internos a la gran red. Enmascarar es una forma de NAT (Network Address Translation).

Capítulo 12, nos da una breve introducción a la puesta en marcha de algunas aplicaciones de red, como los comandos **rlogin** y **rsh**. Este capítulo también cuenta cómo los servicios se gestionan mediante el *superservidor* **inetd**; y cómo puede limitarse su acceso a ciertos nodos.

Capítulo 13, y Capítulo 14, hablan acerca de NIS y NFS. NIS es una utilidad que permite distribuir información administrativa, como contraseñas de los usuarios de una red local. NFS permite compartir sistemas de ficheros entre diferentes nodos de la red.

En Capítulo 15, veremos el protocolo IPX y el sistema de ficheros NCP. Estos dos permiten a Linux integrarse en un entorno compatible con Novell Netware, compartiendo ficheros e impresoras con sistemas sin Linux.

Capítulo 16, nos da una amplia introducción a la administración de Taylor UUCP, una implementación libre del conjunto UUCP.

El libro finaliza con un completo viaje por el correo electrónico y las noticias Usenet. Capítulo 17, nos introduce los conceptos principales de correo electrónico, cómo son los mensajes y cómo funciona el transporte de correo.

Capítulo 18, y Capítulo 19, cubren la configuración de **sendmail** y **exim**, dos agentes de transporte de correo que podemos utilizar en Linux. Este libro explica ambos, ya que mientras que **exim** es la opción fácil de instalar, **sendmail** será el que elijamos para configuraciones que combinen correo UUCP con correo de Internet.

Desde Capítulo 20 hasta Capítulo 23, contaremos cómo funcionan las noticias de Usenet y cómo instalar y usar C News, **nntpd** e INN, tres paquetes habituales para transportar y servir noticias. Tras una breve introducción en Capítulo 20, podemos leer Capítulo 21, si pensamos transportar las noticias con C News, que tradicionalmente se usa en redes UUCP. Los siguientes capítulos tratan alternativas más modernas como el protocolo para Internet, NNTP (Network News Transfer Protocol). Capítulo 22 trata el funcionamiento de un sencillo demonio servidor **nntpd**, que da acceso de noticias a una red local, mientras que en Capítulo 23 veremos un servidor más robusto para transporte de noticias, el INN (InterNet News daemon). Finalmente, Capítulo 24 nos muestra cómo configurar y mantener diversos programas de lectura de noticias.

8. Convenciones en Este Libro

Todos los ejemplos del libro asumen que estamos usando una shell compatible con **sh**, como **bash**. Si somos usuarios de una shell tipo **csh** habrá que realizar los correspondientes cambios a los ejemplos.

A continuación mostramos las convenciones tipográficas utilizadas en este libro:

Cursiva
: La usamos para nombres de fichero y directorio, programas o comandos, opciones de línea de comandos, direcciones de correo, URLs y para enfatizar términos.

Negrita
: La usamos para nombres de nodo, sitios, nombres de usuario, identificadores y a veces para enfatizado.

`Ancho fijo`
: Se usa en los ejemplos para mostrar trozos de ficheros o salida de comandos. También para variables de entorno y palabras clave que aparecen en el código.

`Ancho fijo cursiva`
: Se usa para indicar opciones variables, palabras clave o texto que el usuario debe reemplazar por un valor concreto.

`Ancho fijo negrita`
: Se usa en ejemplos para mostrar el texto o comandos que el usuario teclea en la máquina.

Aviso
El texto que aparece así resaltado es un aviso. No seguir su recomendación puede producir un error en el

sistema difícil de recuperar.

9. Envío de Cambios

Hemos comprobado y verificado que la información de este libro es bastante correcta. Sin embargo pueden encontrarse cosas que han cambiado (¡o simplemente errores!). Por favor, háganos llegar cualquier error que descubra, así como sugerencias para futuras ediciones, escribiendo a:

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472
1−800−998−9938 (in the U.S. or Canada)
1−707−829−0515 (international or local)
1−707−829−0104 (FAX)

También puede enviarnos mensajes electrónicos. Para inscribirse en la lista de correo o pedir un catálogo, envíe un mensaje a:

info@oreilly.com

Para plantear cuestiones técnicas o comentarios sobre el libro, envíe mensajes a:

bookquestions@oreilly.com

Tenemos una página dedicada a este libro, donde ponemos ejemplos, fe de erratas y otras posibles ampliaciones. La página está disponible en:

http://www.oreilly.com/catalog/linag2

Para más información sobre este y otros libros, consúltese el sitio web de O'Reilly:

http://www.oreilly.com

10. Agradecimientos

Esta edición de la Guía de Redes es fruto del trabajo de Olaf y Vince. Es difícil apreciar el trabajo que conlleva la investigación y tarea posterior de escritura de un libro de esta naturaleza, hasta que nos vemos de lleno en un trabajo de este tipo. Actualizarlo es un reto adicional, pero con una buena base de la que partir, es también una tarea muy interesante y amena.

En este libro ha colaborado mucha gente leyéndolo para encontrar errores, tanto técnicos como ortográficos o gramaticales. Agradecemos específicamente la participación de Phil Hughes, John Macdonald y Erik Ratcliffe.

También agradecemos la ayuda al personal de O'Reilly con quienes hemos tenido el placer de trabajar: Sarah Jane Shangraw, quien le dio al libro forma, Maureen Dempsey, quien maquetó el texto; Rob Romano, Rhon Porter y Chris Reilley, quienes crearon las ilustraciones; Hanna Dyer, que diseñó la cubierta; Alicia Cech, David Futato y Jennifer Niedherst, que se ocuparon del formato; Lars Kaufman que sugirió tonos madera en el tema visual; Judy Hoer por el índice; y finalmente, Tim O'Reilly por el coraje para sacar adelante este proyecto.

Estamos en deuda con Andrés Sepúlveda, Wolfgang Michaelis, Michael K. Johnson y otros desarrolladores que gastaron su tiempo en comprobar la información proporcionada en la guía. Phil Hughes, John MacDonald y Eric Ratcliffe contribuyeron con valiosos comentarios en la segunda edición. También tenemos que agradecer a aquellos que leyeron la primera versión de la guía y nos enviaron correcciones y sugerencias. En el fichero `Thanks` de la distribución en línea podemos encontrar la lista completa de gente que ha participado. Finalmente, este libro no habría sido posible sin la ayuda de Holger Grothe, quien proporcionó a Olaf la conectividad Internet que necesitaba durante la realización de la versión inicial.

Olaf también desea agradecer a los siguientes grupos y empresas que imprimieron la primera edición y donaron beneficios bien a Olaf o bien al proyecto de documentación de Linux: Linux Support Team, Erlangen, Alemania; S.u.S.E. GmbH, Fuerth, Alemania; y Linux System Labs, Inc., Clinton Twp., Estados Unidos, RedHat Software, North Carolina, Estados Unidos.

Terry quiere dar las gracias a su mujer, Maggie, quien pacientemente soportó su participación en el proyecto independientemente de los retos que suponían el nacimiento de su primer hijo, Jack. Además, agradece a las *muchas personas* de la comunidad Linux que le han apoyado.

10.1. El Hall de la Fama

Además de los ya mencionados, otras muchas personas han participado en la Guía de Redes, revisándola y enviándonos correcciones y sugerencias. A todos les estamos muy agradecidos.

A continuación, una lista de todos aquellos de los que tenemos constancia en nuestras carpetas de correo:

Al Longyear, Alan Cox, Andres Sepúlveda, Ben Cooper, Cameron Spitzer, Colin McCormack, D.J. Roberts, Emilio Lopes, Fred N. van Kempen, Gert Doering, Greg Hankins, Heiko Eissfeldt, J.P. Szikora, Johannes Stille, Karl Eichwalder, Les Johnson, Ludger Kunz, Marc van Diest, Michael K. Johnson, Michael Nebel, Michael Wing, Mitch D'Souza, Paul Gortmaker, Peter Brouwer, Peter Eriksson, Phil Hughes, Raul Deluth Miller, Rich Braun, Rick Sladkey, Ronald Aarts, Swen Thüemmler, Terry Dawson, Thomas Quinot, y Yury Shevchuk.

11. La traducción al español

Esta traducción, coordinada desde el Proyecto LuCAS[9] no hubiera sido posible sin la desinteresada colaboración de numerosas personas. A continuación citamos todos aquellos que han colaborado en esta segunda edición de la guía.

Carlos G. Arqués
Natalia Ballesteros
Hardy Beltrán
Daniel Callejas
Gabriel Ceravolo
Francisco Javier Fernández
José M. Fernández
David Grajal
Sebastián Gurin «Cancerbero»
Eduardo Hernández
Luis Jäeger
Raul Mahiques
Luis Alberto Melgar
Max de Mendizábal
Toni Pérez

José Manuel Puerta
Gabriel Rodríguez
Domingo Sánchez
César Tapia
Javier Villa
Francisco Javier Fernández – Revisiones
Juan José Amor – Coordinador

Buena parte de esta segunda edición de la guía reutiliza textos de la primera edición. Por ello no sería justo terminar este capítulo sin recordar a quienes hicieron posible aquella edición. Muchas gracias, por tanto, a Corsino Álvarez, Iñaki Arenaza, César Ballardini, Alfonso Belloso, Javier Bravo, Santiago Crespo, David Escorial, Máximo Escobar, Manuel Jesús Garrido, Luis F. González, Eduardo Hernández, Jesús Jiménez, José Andrés Jiménez, Urko Lusa, Carlos Martínez Txakartegi, Max de Mendizábal, Francisco J. Montilla, José Manuel Puerta, Angel Luis Pinazo, Pedro Soria Rodríguez y Enrique Zanardi.

Capítulo 1. Introducción al Trabajo en Redes

1.1. Historia

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

Por supuesto, la humanidad ha avanzado un poco desde la la Edad de Piedra; ya no se usan aquellos primitivos artefactos ni tenemos los mismos inconvenientes que nuestros antepasados. En la actualidad, contamos con computadoras que hablan con otras sobre una colección de cables, fibra óptica, microondas, etc. tan grande como para llenar el estadio en el partido de fútbol de los sábados[10] A continuación, se hará referencia a los conceptos y métodos que son utilizados para llevar a cabo todo esto. Sin embargo, dejaremos de lado tanto el tema de los cables, como la parte del fútbol.

En esta guía se describirán tres tipos de redes. Sin embargo, se discutirá más profundamente TCP/IP puesto que es el protocolo más usado, ya sea en Redes Locales (Local Area Networks, LANs), o en Redes de Área Amplia (Wide Area Networks, WANs), como por ejemplo, Internet. También se echará un vistazo a UUCP e IPX. UUCP fue antiguamente el medio general para transportar las noticias y los mensajes de correo, mediante una conexión telefónica. Es menos usado en estos días, pero sigue siendo útil en muchas situaciones. El protocolo IPX es usado más frecuentemente en los entornos Novell NetWare, y se detallará cómo usarlo para conectar una máquina GNU/Linux a una red Novell. Cada uno de estos protocolos de red son usados para transportar datos entre computadoras. Se discutirá aquí cómo son usados y se hará una introducción a sus principios fundamentales.

Se define una red, como una colección de *nodos* (del inglés hosts), capaces de comunicarse entre sí, a veces confiando en los servicios de un número determinado de máquinas que se encargan de transmitir datos entre quienes que lo demanden. Los nodos son casi siempre computadoras, pero no necesariamente; se puede pensar, sin equivocación, en terminales X o impresoras inteligentes como nodos. Por otro lado, a las pequeñas aglomeraciones de éstos, se las denomina *sitios*, (sites).

La comunicación, sería imposible sin algún tipo de lenguaje o código. En la jerga de las redes de computadoras, estos lenguajes se denominan conjuntamente como *protocolos*. No obstante, no se debería pensar aquí en lenguajes ya escritos y definidos, sino más bien en el código de comportamiento altamente formalizado, que se observa en una población cuando se reúnen jefes de estado, por citar un ejemplo. Así, los protocolos usados en las redes de computadoras no son más que reglas muy estrictas de intercambio de mensajes entre dos o más servidores.

1.2. Redes TCP/IP

Las aplicaciones modernas para trabajo en redes requieren de un sofisticado método de transporte desde una máquina a otra. Si usted administra una máquina GNU/Linux que posea muchos usuarios, los cuales desean estar conectados simultáneamente a un servidor remoto o a una red, necesitará un modo de acceso para que puedan compartir la conexión a la red, sin que las acciones de cada uno interfieran con las de los demás. La estrategia que un gran número de protocolos de red utilizan hoy día se llama *conmutación de paquetes*, (packet−switching). Un *paquete* es nada más que un pequeño trozo de datos que se transfiere de una máquina a otra a través de una red. Esta transferencia ocurre a medida que el datagrama es transmitido a través de cada enlace en la red. Una red de conmutación de paquetes comparte un único enlace con muchos usuarios, enviando los paquetes alternadamente, desde un usuario a otro, a través de ese enlace.

La solución que muchos sistemas Unix, (y posteriormente muchas otras plataformas), han adoptado, se conoce como TCP/IP. Cuando se habla de redes TCP/IP, siempre estará presente el término *datagrama*. Técnicamente, este término tiene un significado especial, pero es a menudo usado de forma intercambiable con *paquete*. En la siguiente sección, se echará un vistazo a los conceptos fundamentales de los protocolos TCP/IP.

1.2.1. Introducción a las Redes TCP/IP

El origen del protocolo TCP/IP, se debe a un proyecto de investigación, financiado por la DARPA, (Defense Advanced Research Projects Agency, o Agencia de Proyectos Avanzados de Investigación en Defensa), en 1969. La ARPANET, fue una red experimental que se convirtió en funcional a mediados de 1975, tras haber sido admitida su funcionalidad.

En 1983, el nuevo conjunto de protocolos TCP/IP, fue adoptado como estándar y todas las máquinas de la red tuvieron la necesidad de él. Cuando, finalmente, ARPANET creció y se convirtió en Internet, (integrándose luego ella misma a Internet, en 1990), el uso de TCP/IP se propagó incluso a redes ajenas a ella. Ahora, muchas compañías empresariales construyen redes TCP/IP, e Internet ha crecido hasta tal punto, que se la puede considerar como la corriente principal de consumo tecnológico. Actualmente, es difícil leer un periódico sin ver referencias sobre Internet; casi todo el mundo ya puede usarla.

Para apreciar algo palpable sobre lo que hemos discutido anteriormente, supongamos como ejemplo, la Universidad Groucho Marx, (GMU), la cual se encuentra en algún lugar de Federilandia. La mayoría de las divisiones de la universidad tienen su propia red local, mientras que algunas comparten una sola y otras poseen muchas de ellas. Todas se encuentran interconectadas, y están enlazadas a Internet por un simple enlace de alta velocidad.

Supóngase que se tiene una máquina GNU/Linux conectada a una LAN de servidores Unix en la división de Matemáticas, y su nombre es erdos. Para acceder a un servidor que se encuentra en la división de Física, cuyo nombre es, por ejemplo quark, se deberá introducir la siguiente orden:

```
$ rlogin quark.physics
Welcome to the Physics Department at GMU
(ttyq2) login:
```

Ante este indicador se podrá introducir un nombre de usuario, por ejemplo sebastian, y una contraseña. Luego, si todo es correcto, nos encontraremos frente a un intérprete de órdenes (shell)[11] de quark, en la cual, se podrá escribir como si se estuviera sentado frente a la misma consola del sistema. Tras salir del intérprete, se nos presentará nuevamente el antiguo indicador de órdenes de nuestra máquina. Se ha usado aquí, tan sólo una de las muchas aplicaciones instantáneas e interactivas que TCP/IP proporciona: remote login (registro remoto).

Mientras se trabaja en quark, puede que se desee ejecutar una aplicación de interfaz gráfica, como por ejemplo un procesador de textos, un programa de diseño gráfico, o hasta un navegador de Internet. El sistema de ventanas X es un entorno gráfico para el usuario, totalmente funcional bajo redes y está disponible para muchos tipos de sistemas informáticos. Para hacerle saber a la aplicación que se desea tener interfaz gráfica en la pantalla de nuestro nodo, se necesitará determinar la variable de entorno DISPLAY:

```
$ DISPLAY=erdos.maths:0.0
$ export DISPLAY
```

Si ahora se ejecuta la aplicación gráfica, ésta se comunicará con el servidor X de nuestro nodo en lugar de hacerlo con el de quark, y como consecuencia las ventanas aparecerán en nuestra la pantalla y no en la de nuestro servidor. Por supuesto, esto requiere que se esté ejecutando X11 en erdos. Lo más importante aquí es que TCP/IP permite el envío y reenvío de paquetes X11 entre quark y erdos, haciendo que el usuario tenga la ilusión de que trabaja en una única máquina. Trabajando de este modo, la red será bastante transparente.

Otra aplicación muy importante en una red TCP/IP es NFS, que significa *Network File System* (Sistema de Ficheros de Redes). Es otra forma de hacer de la red un sistema transparente, ya que, básicamente, permite al usuario trabajar con los ficheros y directorios de otros nodos como si fueran locales. Por ejemplo, todos los directorios \home de cada usuario pueden alojarse en un servidor central. Desde éste, los demás nodos de la LAN pueden montarlos cuando sea necesario. El resultado es que los usuarios pueden registrarse en el sistema y encontrarse siempre en el mismo directorio \home. De modo similar, es posible compartir grandes cantidades de datos, (como una base de datos, documentación o programas ejecutables), entre muchos nodos, almacenando físicamente una sola copia de dichos datos en un servidor, y permitiendo a los nodos en cuestión acceso a él. Se volverá a hablar de NFS en Capítulo 14.

Por supuesto, estos son sólo ejemplos de lo que se puede hacer en redes TCP/IP. Las posibilidades son casi infinitas, y el lector irá conociéndolas a medida que avance en el libro.

En las siguientes secciones, se estudiará más detenidamente, de qué manera funciona una red TCP/IP. Esta información ayudará a entender cómo y por qué se debe configurar una máquina. Se empezará examinando el hardware, y desde allí con las demás cuestiones.

1.2.2. Ethernets

El tipo de hardware más utilizado en LANs es lo que comúnmente conocemos como Ethernet. Descrito de una forma simple, consta de un solo cable con los nodos unidos a él a través de conectores, clavijas o transceptores. Los adaptadores Ethernet simples, son relativamente baratos de instalar, lo que unido a un flujo de transferencia neto de 10, 100 o hasta 1,000 Mega bits por segundo, avala gran parte de su popularidad.

Las redes Ethernet se pueden clasificar en tres tipos atendiendo al grosor del cable: *gruesos*,*finos*, y *de par trenzado*. Los dos primeros pueden usar cable coaxial, diferiendo en el grosor y el modo de conectar este cable a los nodos. El cable Ethernet fino emplea conectores BNC con forma de T, que se pinchan en el cable y se enganchan a los conectores de la parte trasera del ordenador. El cable Ethernet grueso requiere que se realice un pequeño agujero en el cable, y se conecte un transceptor utilizando un conector vampiro Luego, se podrán conectar uno o más nodos al transceptor. Los cables Ethernet fino y grueso pueden alcanzar una distancia de 200 y 500 metros, respectivamente, y es por ello que se les llama también 10base−2 y 10base−5. La palabra base hace referencia a modulación de banda base y significa, simplemente, que los datos que alimentan al cable, fluyen directamente sin pasar por un módem. El número que se encuentra delante de la palabra alude a la velocidad de transmisión, en Mega bits por segundo, mientras que el número al final indica la máxima longitud que se le puede dar al cable, en cientos de metros. El par trenzado usa un cable hecho de dos hilos de cobre. Por lo común necesitan, además, hardware adicional que se conoce como *Núcleo Activo*. A este Ethernet se le conoce también como 10base−T, en donde T significa de par trenzado. Los pares trenzados con velocidad de 100 Mega bits por segundo son conocidos como 100base−T.

Para agregar un nodo a una instalación Ethernet fina se deberá suspender el servicio de la red por al menos unos minutos, ya que se deberá cortar el cable para insertar un conector. A pesar de que, por otro lado, agregar un nodo a un sistema Ethernet grueso es un poco complicado no hará, por lo general, que el servicio de la red se interrumpa. Un Ethernet de par trenzado es aún más simple. Usa un dispositivo denominado hub, que trabaja como un punto de interconexión. Se pueden insertar y quitar nodos de un núcleo sin interrumpir en absoluto, a ninguno de los demás usuarios.

La mayoría de gente prefiere el Ethernet fino porque es barato: las tarjetas de PC pueden encontrarse por unos 30 americanos (algunas compañías están literalmente, regalándolas), y el cable por pocos centavos el metro. Sin embargo, para instalaciones de gran escala, son más apropiados el Ethernet grueso o el de par trenzado. Por ejemplo, en un principio, el Departamento de Matemáticas de la GMU decidió utilizar el cableado Ethernet grueso, ya que el gran tráfico que posee toda la red a lo largo de su gran recorrido, no se interrumpe cada vez que se añade un nodo. Actualmente, son muy comunes los cables Ethernet de par trenzado en una gran variedad de instalaciones. Los hubs son ahora más accesibles, y pequeñas unidades están disponibles a precios que son atractivos, incluso para pequeñas redes domésticas. El cable de par trenzado puede ser significativamente más barato para grandes instalaciones. Además, el mismo cable de par trenzado es mucho más flexible que los coaxiales usados por otros sistemas Ethernet. Los administradores de la red en la división de matemáticas de GMU, están planeando reemplazar su sistema por uno de par trenzado el año que viene, ya que, además de ahorrar tiempo a la hora de agregar nuevos nodos, y cambiar de lugar los viejos, también podrán ponerse al día con la tecnología actual.

Uno de los inconvenientes de la tecnología Ethernet es su limitada longitud de cable, que imposibilita cualquier uso fuera de las LANs. Sin embargo, pueden enlazarse varios segmentos de red Ethernet entre sí utilizando repetidores, puentes o encaminadores[12]. Los repetidores simplemente copian las señales entre dos o más segmentos, de forma que todos los segmentos juntos actúan como si fuese una única Ethernet. Debido a requisitos de tiempo, no puede haber mas de cuatro repetidores entre cualquier par de nodos de la red. Los puentes y encaminadores son más sofisticados, analizan los datos de entrada y los reenvían sólo si el nodo receptor no está en la Ethernet local.

Ethernet funciona como un sistema de bus, donde un nodo puede mandar paquetes (o *marcos*) de hasta 1500 bytes a otro nodo de la misma Ethernet. A cada nodo se le asigna una dirección de seis bytes grabada en el firmware (memoria fija) de su tarjeta Ethernet. Estas direcciones se especifican generalmente como una secuencia de números hexadecimales de dos dígitos separados por dos puntos, como por ejemplo aa:bb:cc:dd:ee:ff.

Una trama enviada por una estación es vista por todas las demás estaciones conectadas, pero sólo el nodo destinatario la toma y la procesa. Si dos estaciones intentan emitir al mismo tiempo, se produce lo que se llama una *colisión*. Una colisión en un complejo Ethernet, es detectada electrónicamente por las tarjetas de interfaz. Se resuelve por parte de las dos estaciones abortando el envío, y reintentándolo al cabo de un intervalo de tiempo tomado al azar. Seguramente se han escuchado muchas historias que afirmen que las colisiones en un Ethernet son un problema, y que la verdadera tasa de transmisión de datos en un Ethernet, sólo ocupa un 30 por ciento del ancho de banda disponible debido a ellas. La verdad es que las colisiones en un sistema Ethernet son un fenómeno *natural*. Es más, en un sistema muy activo, no se debería sorprender al ver que las colisiones tienen un índice mayor al 30 por ciento. En la práctica, el administrador de una red Ethernet sólo debería preocuparse cuando la tasa de transmisión se vea limitada a aproximadamente un 60 por ciento del ancho de banda.[13]

1.2.3. Otro Tipo de Hardware

En instalaciones mayores, como la Universidad de Groucho Marx, Ethernet no es el único tipo de red que puede utilizarse. Hay muchos otros tipos de protocolos disponibles y usados en la actualidad, y cada uno de ellos son soportados por GNU/Linux. A continuación se hará una descripción de otros protocolos usados, aunque será algo breve debido a las restricciónes en el tamaño de este documento. La mayoría de estos protocolos poseen documentos HOWTO que los describen en detalle. Es por esto que se debería leerlos si se está interesado en explorar aquellos protocolos que no se describen aquí.

En la Universidad de Groucho Marx cada LAN de un departamento está enlazada a la espina dorsal de la red del campus. Ésta está formada por un cable de fibra óptica funcionando en *FDDI* (Fiber Distributed Data Interface). FDDI emplea un enfoque totalmente diferente para transmitir datos, que básicamente implica el envío de un número de *símbolos*, de modo que una estación sólo pueda enviar una trama si captura un símbolo. La principal ventaja de FDDI es la reducción de colisiones. Como consecuencia, el paso de datos puede utilizar en mayor proporción el ancho de banda, lo que permite una velocidad de hasta 100 Mbps. Otro beneficio de FDDI es que, al utilizar fibra óptica, la máxima longitud del cable sea mucho mayor a la que ofrecen las tecnologías basadas en cables, como Ethernet. La máxima longitud de cable usando FDDI oscila en los 200 km, lo que hace que esta tecnología sea ideal para unir las máquinas que se encuentren en distintos edificios de una ciudad. En el caso de nuestra universidad, FDDI une a los diferentes edificios en un campus.

De modo similar, si se tratase de equipos IBM, sería muy común el ver una red IBM de Token Ring. Esta tecnología es usada, en algunos entornos LAN, como alternativa a Ethernet. La ventaja esencial es que, como

FDDI, en términos de utilización de la banda, se reducen las colisiones, aunque a velocidades inferiores (de 4 a 16Mbps). Su coste es menor que el de FDDI, ya que utiliza cables en lugar de fibra óptica. En un sistema GNU/Linux una red basada en Token Ring se configura casi de la misma manera que una red Ethernet, por lo que no se cubrirá, en el libro este procedimiento específicamente.

A pesar de que otras tecnologías para LAN soportadas por GNU/Linux, como por ejemplo ArcNet o DECNet, pueden ser instaladas, no se describirán aquí. Esto es debido, principalmente a que son muy poco usadas en la actualidad.

Muchas redes nacionales, operadas por compañías de Telecomunicaciónes, soportan otros protocolos basados en la conmutación de paquetes. Probablemente, el más popular de estos es un estándar llamado X.25. Muchas Redes de Datos Públicos, como por ejemplo Tymnet en EEUU, Austpac en Australia, y Datex−P en Alemania, ofrecen este servicio. X.25 define una conjunto de protocolos que describen cómo una terminal de datos se comunicará con otros equipos de transmisión, (o sea, un interruptor X.25). X.25 requiere un enlace de datos síncrono y, por consiguiente, un puerto síncrono especial en el hardware. Se puede usar X.25 en un puerto serie normal, con ayuda de un dispositivo especial llamado PAD, (Packet Assembler Disassembler). El PAD hace que el puerto en serie trabaje de modo síncrono o asíncrono, según sean las condiciones de la tarea. El dispositivo entiende el protocolo X.25 de un modo tal, que las simples terminales pueden efectuar y/o aceptar conexiones vía X.25.

X.25 también es usado para transportar otros protocolos de redes, como TCP/IP. Dado que los datagramas IP no pueden ser fácilmente asignados a X.25, (o recíprocamente), son encapsulados en paquetes X.25, y transmitidos por la red. Existe una implementación experimental del protocolo X.25 disponible para GNU/Linux.

Un protocolo más reciente ofrecido por compañías de telecomunicaciónes es el denominado *Conmutación de Tramas*, (Frame Relay). Este protocolo tiene características técnicas similares a las del X.25, aunque su comportamiento es mucho más parecido al IP. Al igual que el protocolo X.25, el de Conmutación de Tramas requiere un tipo de hardware síncrono especial. Debido a la similitud que existe entre estos dos protocolos, muchas tarjetas soportan ambos. Existe una alternativa, la cual no requiere de hardware interno y que consiste en un componente externo de hardware, denominado Dispositivo de Acceso a Conmutación de Tramas, (FRAD)[14], el cual administra la encapsulación de los paquetes Ethernet en paquetes de Conmutación de Tramas para ser transmitidos a través de la red. El protocolo de Conmutación de Tramas es ideal para transportar al TCP/IP de un sitio a otro. GNU/Linux provee de controladores que soportan algunos tipos de dispositivos internos para el protocolo de Conmutación de Tramas.

Si se necesita trabajar en una red de alta velocidad, la cual sea capaz de transportar muchos tipos de datos, como por ejemplo sonido o vídeo digitalizado, al mismo tiempo que los datos usuales, ATM (Modo de Transferencia Asíncrona)[15] es, con seguridad, lo que se está buscando. ATM es una nueva tecnología de redes, la cual fue específicamente desarrollada para suministrar control sobre la Calidad del Servicio (Quality of Service, Q.S, en inglés). Muchas compañías de telecomunicaciones han destacado la infraestructura de la tecnología ATM, ya que permite integrar diferentes tipos de servicios en una sola plataforma, todo esto aspirando al ahorro en cuanto a la administración y a los costos de mantenimiento. También ATM se usa para transportar al protocolo TCP/IP. En el documento Networking−HOWTO se puede encontrar información sobre el soporte brindado por GNU/Linux para ATM.

A menudo, los radio−aficionados usan sus propios equipos de radio para conectar sus ordenadores en red; comúnmente, a esto se le llama *radio paquetes* (packet radio). Uno de los protocolos usados por los operadores radio−aficionados es llamado AX.25, que deriva del X.25. También, los operadores radio−aficionados usan al AX.25 para transmitir otros protocolos, como por ejemplo el TCP/IP. AX.25, al igual que X.25, requiere de hardware especial que le permita realizar operaciones síncronas, o un dispositivo externo llamado Controlador de Nodo Terminal[16], el cual convierta los paquetes transmitidos por un enlace en serie asíncrono, en paquetes transmitidos síncronamente. Existen muchas clases diferentes de interfaces de tarjetas disponibles que soporten la operación de radio paquetes. Normalmente, estas tarjetas son aludidas según la base Z8530 SCC y su nombre va tras el controlador de comunicación más popular usado en el diseño. Dos de los protocolos que son transportados comúnmente por el AX.25 son NetRom y Rose, los cuales se denominan protocolos de red en capas (Network layer protocols). Puesto que estos últimos se ejecutan sobre AX.25, tienen los mismos requerimientos de hardware que este último. GNU/Linux soporta ampliamente todas las características de los protocolos AX.25, NetRom y Rose. Una buena fuente de información, sobre la implementación para GNU/Linux de éstos, es el HOWTO AX25−Como.

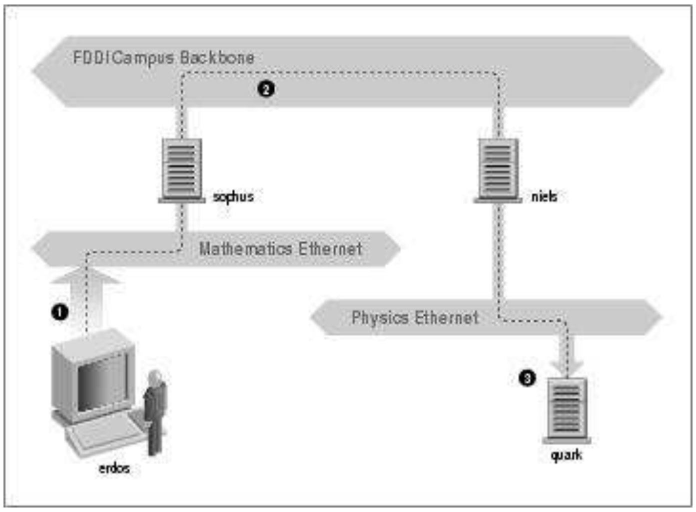
Otro tipo de acceso a Internet implicaría la utilización de marcación telefónica a una central, sobre una línea serie. Esto involucraría una conexión más lenta, aunque más barata, (usando el teléfono, RDSI, u otros servicios). Esto requiere todavía de la ayuda de otro protocolo para la transmisión de los paquetes, como por ejemplo SLIP o PPP, los que serán descritos más adelante.

1.2.4. El Protocolo IP (Internet Protocol)

Por supuesto, el administrador puede no querer que su red esté limitada solamente a una Ethernet, o a un sólo enlace de datos punto−a−punto. Seguramente la idea original consistirá en poder acceder a un servidor sin importar el hardware del que dispone. Por ejemplo, en instalaciones grandes como la Universidad de Groucho Marx, se encontrará muy a menudo con varias redes distanciadas unas de otras, pero conectadas entre ellas de alguna manera. En la GMU, el departamento de matemáticas tiene dos Ethernets: una red de máquinas rápidas para profesores y graduados, y otra con máquinas más lentas para estudiantes. Ambas redes están enlazadas de la red troncal FDDI del campus.

Esta conexión se gestiona con un nodo dedicado, denominado *pasarela*, o gateway, que maneja los paquetes entrantes y salientes copiándolos entre las dos Ethernets y el cable de fibra óptica. Por ejemplo, si se encuentra en el Departamento de Matemáticas, y quiere acceder a quark situada en la LAN del Departamento de Físicas, desde su máquina GNU/Linux, el software de red no puede mandar paquetes a quark directamente, porque no esta en la misma Ethernet. Por tanto, tiene que confiar en la pasarela para que actúe como retransmisor. La pasarela (llamémosla sophus) reenvía entonces estos paquetes a su pasarela homóloga niels del Departamento de Física, usando la red troncal, y por fin niels los entrega a la máquina destino. El flujo de datos entre erdos y quark se muestra en Figura 1−1.

Figura 1−1. Los tres pasos del enviado de un datagrama desde erdos a quark



Este esquema de envío de datos al nodo remoto se llama *encaminamiento*, y en este contexto a los paquetes se les denomina *datagramas*. Para facilitar las cosas, el intercambio de datagramas esta gobernado por un único protocolo que es independiente del hardware utilizado: IP, o *Internet Protocol* (Protocolo de Internet). En Capítulo 2, trataremos con más detalle al IP y al encaminamiento.

El principal beneficio del IP es su cualidad de convertir a redes físicamente diferentes en una red aparentemente homogénea. A esto se le llama interconexión de redes, y a la resultante meta−red se la denomina *internet*. Obsérvese aquí la sutil diferencia entre *una* internet y *la* Internet. El último es el nombre oficial de una internet global en particular.

Claro que el IP también necesita un esquema de direccionamiento independiente del hardware. Esto se consigue asignando a cada nodo un número único de 32 bits, denominado *dirección IP*. Una dirección IP está definida normalmente, por 4 números en decimal, uno por cada división de 8 bits, y separados por puntos. Por ejemplo, quark podría tener una dirección IP 0x954C0C04, que se escribiría como 149.76.12.4. Este formato de dirección, es comúnmente llamado *notación decimal de puntos*, aunque también puede hacerse referencia a él como *notación cuadrangular de puntos*[17]. Sin embargo la denominación de IP, está cambiando del nombre de IPv4, (por Internet Protocol, Version 4), a un nuevo estándar llamado IPv6 que ofrece mucha más flexibilidad a la hora de direccionar y otras mejoras modernas. Pasará por lo menos un año tras esta edición, antes de que IPv6 empiece a ser usado.

Se dará cuenta de que ahora tenemos tres tipos distintos de direcciones: primero, tenemos el nombre del nodo, como por ejemplo quark, después tenemos las direcciones IP, y por fin están las direcciones hardware, como la dirección Ethernet de 6 bytes. De alguna forma todas ellas deben relacionarse, de modo que cuando se escriba **rlogin quark**, se le pueda pasar la dirección IP de quark al software de red; y cuando el nivel IP envíe datos a la Ethernet del Departamento de Físicas, de algún modo tenga cómo encontrar a que dirección Ethernet corresponde la dirección IP.

Se hará un repaso de todo esto, con más profundidad en Capítulo 2. De momento, es suficiente con indicar que estos pasos para encontrar las direcciones se llaman: *resolución de nombres*al trazar un mapa de nombres de nodo con direcciones IP, y *resolución de direcciones*, al hacer corresponder estas últimas con direcciones hardware.

1.2.5. IP en Líneas Serie

Para líneas serie se usa frecuentemente el estándar de facto conocido como SLIP o *Serial Line IP* (IP sobre línea en serie). Una modificación del SLIP es el CSLIP, o *SLIP Comprimido*, que realiza compresión de las cabeceras IP para aprovechar el bajo ancho de banda que proporcionan los enlaces serie. Otro protocolo serie es el PPP, o *Point−to−Point Protocol* (Protocolo Punto a Punto). PPP dispone de muchas más características que SLIP, lo que lo hace mucho más atractivo. Su principal ventaja sobre SLIP es, sin embargo, que no se limita a transportar datagramas IP, sino que se diseñó para que la transmisión de cualquier tipo de protocolo pueda realizarse sobre él.

1.2.6. El Protocolo de Control de Transmisión, TCP

Pero la historia no se acaba con el envío de datagramas de un nodo a otro. Si se registra en quark, necesita disponer de una conexión fiable entre su proceso **rlogin** en erdos y el proceso del intérprete de órdenes en quark. Así, la información enviada en uno u otro sentido debe dividirse por paquetes en el origen, y ser reensamblada en un flujo de caracteres por el receptor. Esto que parece trivial, implica varias tareas complejas.

Una cosa importante a saber sobre IP es que, por sí sólo, no es fiable. Suponga que diez personas de su Ethernet comienzan a transferirse la última versión del código fuente del Navegador web Netscape, usando el servidor FTP de GMU. La cantidad de tráfico generada por esto podría ser excesiva para la pasarela, por ser demasiado lenta, o tener poca memoria. Si en ese momento Ud. enviara un paquete a quark, sophus podría tener agotado el espacio del búfer durante un instante y por tanto no seria capaz de reenviarlo. IP resuelve este problema simplemente descartando el paquete el cual se pierde irrevocablemente. Esto traslada, por consiguiente, la responsabilidad de comprobar la integridad y exactitud de los datos a los nodos extremos, y su retransmisión en caso de error.

De este proceso se encarga otro protocolo: el *Protocolo de Control de Transmisión*, (TCP, Transmission Control Protocol), que construye un servicio fiable por encima de IP. La propiedad esencial de TCP es que usa IP para dar al usuario la impresión de una conexión simple entre los procesos en su equipo y la máquina remota, de modo que no tiene que preocuparse de cómo y sobre el recorrido de los datos a través de la ruta por la que viajan. Una conexión TCP funciona básicamente como una tubería de doble sentido, en la que ambos procesos pueden escribir y leer; Se puede usar la analogía de una conversación telefónica para comprender el funcionamiento de este protocolo.

TCP identifica los extremos de una conexión específica por las direcciones IP de los dos nodos implicados, y el número de los *puertos* de cada nodo. Los puertos se pueden ver como puntos de enganche para conexiones de red. Para seguir utilizando el ejemplo del teléfono un poco más, si pensamos en una analogía entre las ciudades como nodos, se puede comparar las direcciones IP con los prefijos de área (los números representarían ciudades), y los números de puerto con los códigos locales (números que representan teléfonos de personas concretas). Un nodo en particular puede soportar diferentes servicios, cada uno diferenciado por su propio número de puerto.

En el ejemplo con **rlogin**, la aplicación cliente (**rlogin**) abre un puerto en erdos y se conecta al puerto 513 de quark, en el cual se sabe que el servidor **rlogind** está escuchando. Esto establece una conexión TCP. Usando esta conexión, **rlogind** desempeña el procedimiento de autorización para luego, generar un servicio de ontérprete de órdenes. La entrada y salida estándar de la shell se redirigen a la conexión TCP, de tal forma que cualquier cosa que se teclee a **rlogin** en nuestra máquina será pasada al flujo TCP para ser luego transmitida a la entrada estándar del intérprete de órdenes.

1.2.7. El Protocolo de Datagramas de Usuario

Sin embargo, TCP no es el único protocolo de usuario en redes TCP/IP. Aunque adecuado para aplicaciones como **rlogin**, la sobrecarga que impone es prohibitiva para aplicaciones como NFS, la cual utiliza un protocolo derivado de TCP llamado UDP, o *User Datagram Protocol* (Protocolo de Datagramas de Usuario). De igual modo que TCP, UDP permite que una aplicación contacte con un servicio en un puerto concreto de la máquina remota, pero no establece una conexión para ello. En cambio, se puede usar para enviar paquetes sueltos al servicio destino − de ahí su nombre.

Supóngase que se ha solicitado una pequeña cantidad de información de un servidor de base de datos. Esto tomará, al menos tres datagramas para establecer la conexión TCP, otros tres para enviar y confirmar la cantidad de datos y otros tres para cerrar la conexión. UDP nos facilita el hacer la mayor parte de todo esto, pero solamente usando dos datagramas. Este protocolo es caracterizado por tener un método de conexión y desconexión mucho más rápido, y no requiere que el usuario establezca y cierre una conexión. El mecanismo es simple: UDP coloca los datos en un datagrama y lo envía al servidor. Éste realiza un proyecto del reenvío, poniendo los datos dentro de un datagrama direccionado a nuestra máquina, y luego lo transmite. Mientras que este procedimiento es más rápido y más eficiente que el de TCP para transacciónes simples, UDP no fue construido para tratar con posibles pérdidas de datos. Lidiar con estas dificultades dependerá de la aplicación en cuestión.

1.2.8. Más sobre Puertos

Los puertos se pueden ver como puntos de anclaje para conexiones de red. Si una aplicación quiere ofrecer un cierto servicio, se engancha ella misma a un puerto y espera a los clientes (a esto también se le llama *escuchar* en el puerto). Un cliente que quiera usar este servicio se asigna un puerto libre en su nodo local, y se conecta al puerto del servidor en el nodo remoto. El puerto del servidor podrá ser abierto por diferentes máquinas, pero nunca podrán usarlo más de una al mismo tiempo.

Una propiedad importante de los puertos es que, una vez que se ha establecido una conexión entre el cliente y el servidor, otra copia del servidor puede engancharse a su mismo puerto y aguardar a otros clientes. Esto permite, por ejemplo, varios accesos remotos simultáneos al mismo nodo, usando todos ellos el mismo puerto

513. TCP es capaz de distinguir unas conexiones de otras, ya que todas ellas provienen de diferentes puertos o nodos. Por ejemplo, si accede dos veces a quark desde erdos, el primer cliente **rlogin** usará el puerto local 1023, y el segundo el 1022. Sin embargo, ambos se conectarán al mismo puerto 513 de quark. Las dos conexiones se distinguirán según el puerto que cada una use en erdos.

Este ejemplo muestra el uso de puertos como puntos de encuentro, donde un cliente se contacta con un puerto específico para obtener un servicio específico. Para que un cliente pueda conectarse al número de puerto correcto, se ha tenido que llegar a un acuerdo entre los administradores de los dos sistemas para definir la asignación de estos números. Para servicios ampliamente usados, como **rlogin**, estos números tienen que administrarse de modo universal. Esto lo realiza el IETF (o Internet Engineering Task Force), que regularmente publica un RFC (Request For Comment) denominado *Assigned Numbers* (Números Asignados, RFC−1700). Describe, entre otras cosas, los números de puerto asignados a servicios reconocidos. GNU/Linux utiliza un archivo para hacer corresponder los nombres con números, llamado `/etc/services`.

Merece la pena indicar que aunque las conexiones TCP y UDP se basan en puertos, estos números no entran en conflicto. Esto significa que el puerto TCP 513, por ejemplo, es diferente del puerto UDP 513. De hecho, estos puertos sirven como puntos de acceso para dos servicios diferentes, como **rlogin** (TCP) y **rwho** (UDP).

1.2.9. La Biblioteca de Sockets

En sistemas operativos UNIX, el software que realiza todas las tareas y protocolos descritos anteriormente es generalmente parte del núcleo, y por tanto viene incorporado dentro de GNU/Linux. La interfaz de programación más común en el mundo UNIX es la *Biblioteca de Sockets de Berkeley*[18]. Su nombre proviene de una analogía popular que ve los puertos como enchufes, y el conectarse a un puerto como enchufarse. Proporciona la llamada `bind` para especificar un nodo remoto, un protocolo de transporte, y un servicio al que un programa pueda conectarse o escuchar (usando `connect`, `listen`, o `accept`)[19]. La biblioteca de sockets, sin embargo, es algo más general, ya que proporciona no sólo una clase de sockets basados en TCP/IP (los sockets `AF_INET`), sino también una clase que maneja conexiones locales a la máquina (la clase `AF_UNIX`). Algunas implementaciones pueden manejar también otras clases, como el protocolo XNS (*Xerox Networking System*), o el X.25.

En GNU/Linux, la biblioteca de sockets forma parte de la biblioteca C estándar, `libc`. Da soporte a los sockets `AF_INET` y `AF_INET6`, para sockets de dominio Unix. También soporta `AF_IPX` para los protocolos de redes Novell; `AF_X25` para el protocolo X.25; `AF_ATMPVC` y `AF_ATMSVC` para el protocolo de redes ATM; y `AF_AX25`, `AF_NETROM`, y `AF_ROSE` para sockets que usen el protocolo de radio−aficionados[20]. En este momentos se están desarrollando otras familias de protocolos conocidos, que se agregarán sin esperar mucho tiempo.

1.3. Redes UUCP

UUCP (Unix−to−Unix Copy), empezó como un paquete de programas para transferir ficheros sobre líneas serie, planificar dichos envíos, e iniciar la ejecución de ciertos programas en sitios remotos. Ha experimentado algunos cambios importantes desde la primera implementación a finales de los setenta, aunque sigue siendo muy estricto en la clase de servicios que ofrece. La principal aplicación de esta tecnología sigue siendo en las Redes de Área Amplia, basándose en enlaces periódicos por red telefónica.

UUCP fue desarrollado por Laboratorios Bell en 1977, con el cometido de comunicar sus sitios Unix en crecimiento. A mediados de 1978, esta red ya contaba con más de 80 sitios, los cuáles tenían acceso a correo electrónico como aplicación, así como a impresión remota. Así y todo, el sistema central en sí, se usaba principalmente para distribuir software y correcciones del mismo. Hoy día, UUCP no está confinada únicamente para el ambiente Unix. Existen puertos de libre distribución y comerciales disponibles para una gran variedad de plataformas, incluyendo a AmigaOS, DOS y el TOS de Atari.

Una de las desventajas secuenciales de las redes UUCP es su operación por lotes. O sea, no cuentan con una conexión permanente establecida entre nodos, sino que usa conexiones temporales. Una máquina UUCP, quizá sólo sea capaz de conectarse una vez al día a otro nodo UUCP, y si es así, sólo podrá hacerlo por un corto período de tiempo. Mientras dure la conexión, se transferirán todas las noticias, los correos y ficheros que estén en estén en cola, y luego se desconectará. Esta cola será lo que limite la cantidad de aplicaciones que puedan hacer uso de la conexión UUCP. En el caso del correo electrónico, un usuario podrá escribir un mensaje y ponerlo en el buzón, (dejarlo en la cola). El mensaje esperará, y será enviado solamente cuando el servidor UUCP se conecte a otro nodo UUCP. Esto no está mal para servicios como el correo electrónico, pero no se usa, en lo absoluto, para otros servicios, como por ejemplo **rlogin**.

A pesar de estas limitaciones, todavía existen muchas redes UUCP funcionando en todo el mundo, operadas principalmente como pasatiempo de algunas personas que ofrecen acceso privado, a precios razonables. La razón principal por la cual UUCP ha durado tanto tiempo es el relativo bajo costo para acceder a ellas, comparado con tener la computadora conectada directamente a Internet. Para hacer de una computadora un nodo UUCP, todo lo que se necesitará será un módem, una implementación UUCP en funcionamiento, y un nodo UUCP que esté pronto para suministrarnos los servicios de correo electrónico y noticias. Mucha gente está dispuesta a levantar servicios UUCP individualmente, ya que este tipo de conexiones no requieren de muchos recursos en sus redes.

Se cubrirá la configuración de UUCP en el capítulo correspondiente, más adelante en el libro. Sin embargo no se discutirá profundamente el uso de esta tecnología, ya que actualmente está siendo reemplazada por TCP/IP. Se puede notar esto, en el hecho de que ahora, el tener acceso barato a Internet se ha convertido en algo común en casi todas las partes del mundo.

1.4. Redes con GNU/Linux

Siendo el resultado del esfuerzo concentrado de programadores de todo el mundo, la creación de GNU/Linux no habría sido posible sin la red global. Tanto así, que no sorprende que ya en los primeros pasos del desarrollo, varias personas comenzaran a trabajar para dotarlo de capacidades de red. Casi desde el principio existía ya una implementación de UUCP para GNU/Linux. En el otoño de 1992 se comenzó a desarrollar el soporte de TCP/IP, cuando Ross Biro y otros crearon lo que ahora se conoce como Net−1.

Después de que Ross dejara el desarrollo activo en Mayo de 1993, Fred van Kempen comenzó a trabajar en una nueva implementación, reescribiendo gran parte del código. Este esfuerzo continuado se conoce como Net−2. En el verano de 1993 salió la primera versión publica de Net−2d (como parte del kernel 0.99.10), y ha sido mantenida y ampliada por varias personas, muy especialmente por Alan Cox[21], dando lugar al Net−2Debugged. Tras una dura corrección y numerosas mejoras en el código, se cambió su nombre a Net−3 después de que GNU/Linux 1.0 fuera sacado al público. El Net−3 fue desarrollado exclusivamente para GNU/Linux 1.2 y GNU/Linux 2.0. Los núcleos de versiones 2.2 en adelante, utilizan el soporte para redes Net−4, el cual es la versión del código de red que se incluye actualmente en las versiónes oficiales del núcleo.

La versión del código de red de GNU/Linux, Net−4, ofrece una gran variedad de controladores para dispositivos, y muchas características avanzadas. Dentro de los protocolos estándar de Net−4 se incluyen SLIP y PPP, (para el envío de tráfico de redes sobre líneas series), PLIP (para líneas paralelas), IPX (para redes compatibles con Novell, sobre las cuáles se hablará en Capítulo 15), Appletalk (para redes Apple), y AX.25, NetRom y Rose, (para redes de radio−aficionados). Otros rasgos característicos de Net−4 son la inclusión de cortafuegos IP, contabilidad IP (tema estudiado más en detalle en Capítulo 9 y Capítulo 10), y Enmascaramiento IP (discutido más adelante, en Capítulo 11)[22]. Está soportada la encapsulación IP, en una unión de diferentes gustos y políticas avanzadas de encaminamiento (routing). También se da soporte a una gran variedad de dispositivos Ethernets, además de algunos FDDI, Token Ring, Frame Relay, y tarjetas ISDN, y ATM.

Además de lo que ya se ha citado, hay algunas otras características que acentúan de gran manera la flexibilidad de GNU/Linux. Dentro de las mismas se destaca una implementación del sistema de ficheros

SMB, el cuál interacciona con aplicaciones como *lanmanager* y Microsoft Windows. Esta implementación se llama Samba, y fue escrita por Andrew Tridgell. También se destaca una implementación de Novell NCP, o Protocolo Principal de NetWare (NetWare Core Protocol).[23]

1.4.1. Diferentes Etapas de Desarrollo

Dentro del ambiente GNU/Linux, en varias ocasiones, se han presentado varios esfuerzos de desarrollo a la vez.

Mientras tanto, al finalizar y lanzar la implementación oficial de Net−2Debugged, Fred siguió desarrollando el Net−2e, que dispone de un diseño más revisado de la capa de red. Fred trabajó para llegar a un estándar, llamado Interfaz de Controlador de Dispositivo, (Device Driver Interface, DDI). Sin embargo, hoy en día el desarrollo de Net−2e ha terminado.

Otra implementación más para redes TCP/IP, es la realizada por Matthias Urlichs, quien escribió un controlador de RDSI(ISDN) para GNU/Linux y FreeBSD. Para ello, integró algo del código de red de BSD, en el núcleo GNU/Linux. Al igual que el anterior, ya no se trabaja más en este proyecto.

Los cambios son acelerados en las implementaciones de red para el núcleo Linux, y cambio sigue siendo el lema, tanto que el desarrollo continúa. Muchas veces, esto también significa que los cambios deberán llegar a otro software, como por ejemplo, las herramientas de configuración de la red. Aunque éste ya no es un problema esencial, como antes lo era, se puede encontrar con que al actualizar el núcleo, también se deban actualizar las herramientas de configuración de la red. Afortunadamente, gracias a la gran variedad de distribuciones disponibles actualmente, esto es una tarea relativamente fácil de llevar a cabo.

La implementación para redes Net−4 está actualmente en una etapa bastante madura, y es usada en un gran número de sitios alrededor del mundo. Ha costado mucho trabajo el mejorar su rendimiento, y ahora compite con las mejores implementaciones disponibles para su misma plataforma de trabajo. GNU/Linux está extendiéndose rápidamente en el entornoe de Proveedores de Servicio de Internet, y muchas veces es elegido por pequeñas organizaciones que necesitan de servidores World Wide Web, de correo y noticias realmente baratos y fiables. Hoy en día, existe el suficiente interés en el desarrollo de GNU/Linux, como para decir que se puede mantener el ritmo de los cambios tecnológicos en cuanto a redes. Por ejemplo, las últimas liberaciones del núcleo de GNU/Linux ofrecen como estándar, la próxima generación del protocolo IP, IPv6.

1.4.2. Dónde Conseguir el Código

Parece extraño el recordar aquellos primeros días del desarrollo del código de red para GNU/Linux. El núcleo estándar requería de un gran conjunto de parches, para dar soporte a redes. En la actualidad, sin embargo, el desarrollo del soporte para redes tiene lugar como parte misma del flujo principal del proceso de desarrollo de GNU/Linux. Los núcleos GNU/Linux estables más recientes, se pueden encontrar en **ftp.kernel.org** dentro de `/pub/linux/kernel/v2.x/`, donde *x* es un número par. Análogamente, los núcleos GNU/Linux experimentales más recientes se pueden obtener en **ftp.kernel.org** dentro de `/pub/linux/kernel/v2.y/`, donde esta vez *y*, es un número impar. Existen réplicas de las distintas versiones del núcleo GNU/Linux repartidas por todo el mundo[24]. Después de esto, sería difícil de imaginar a GNU/Linux sin soporte estándar de red.

1.5. Mantenimiento del Sistema

A lo largo de este libro, se discutirá principalmente cuestiones de instalación y configuración. Sin embargo, la administración de un sistema, es mucho más que esodespués de activar un servicio, también se deberá mantenerlo en correcto funcionamiento. Para la mayoría de éstos, será suficiente con una pequeñas revisiones, pero para otros servicios, como lo son el correo o las noticias, será necesario ejecutar rutinas de verificación para mantener el sistema en óptimo estado. Se discutirán estas tareas, en los capítulos siguientes.

La tarea mínima de mantenimiento es comprobar regularmente el sistema y los ficheros de registro de cada aplicación buscando condiciones de error y eventos inusuales. Por lo general, es posible hacer esto escribiendo un par de scripts de órdenes y ejecutándolos periódicamente mediante la orden **cron**. Se podrá encontrar algunos de estos scripts en distribuciones fuente de algunas aplicaciones importantes como **inn** o C News. Tras obtenerlos, sólo se tendrá que retocarlos para adecuarlos a nuestras necesidades y preferencias.

La salida de cualquiera de los trabajos de nuestro **cron**, se debería enviar a una cuenta de administración. Por defecto, muchas aplicaciones enviarán informes de errores, estadísticas de uso, o resúmenes del fichero de registro a la cuenta de root. Ésto solo tiene sentido si se entra al sistema como root frecuentemente. Una idea mucho mejor es redirigir el correo de root a nuestra cuenta personal, estableciendo un alias de correo como se describe en Capítulo 19 y en Capítulo 18.

De todos modos, por muy cuidadoso que sea configurando su máquina, la ley de Murphy garantiza que *surgirá* algún problema en el futuro. Por lo tanto, el mantenimiento de un sistema implica también estar disponible para quejas. Generalmente la gente espera que se pueda contactar con el administrador del sistema al menos por correo electrónico, como root. Sin embargo, existen otras denominaciones para direcciones de correo usadas comúnmente para contactar a los posibles encargados de la administración de respectivos servicios del sistema. Por ejemplo, las quejas sobre el mal funcionamiento del correo se dirigirán generalmente al postmaster (encargado del correo). Del mismo modo, los problemas con el sistema de noticias pueden ser comunicados a newsmaster o al usenet. El correo al hostmaster se debería redirigir a la persona encargada de los servicios básicos de red del nodo, y del servicio de nombres DNS si esta funcionando un servidor de nombres.

1.5.1. Seguridad del Sistema

Otro aspecto muy importante de la administración de sistemas en un entorno de red es proteger al sistema y a sus usuarios, de intrusos. Los sistemas que son administrados descuidadamente ofrecen muchos huecos a los malintencionados: los ataques van desde averiguar las claves hasta acceder a nivel de Ethernet, y el daño causado puede ser desde mensajes de correo falsos hasta pérdida de datos o violación de la privacidad de los usuarios. Mencionaremos algunos problemas concretos cuando discutamos el contexto en el que pueden ocurrir, y algunas defensas comunes contra ellos.

En esta sección se comentarán algunos ejemplos y técnicas básicas para poder lidiar con la seguridad del sistema. Por supuesto, los temas relatados aquí no pueden tratar exhaustivamente todos los aspectos de seguridad con los que uno se puede encontrar; sirven meramente para ilustrar los problemas que pueden surgir. Por tanto, la lectura de un buen libro sobre seguridad es absolutamente obligada, especialmente en un sistema en red.

La seguridad del sistema comienza con una buena administración del mismo. Esto incluye comprobar la propiedad y permisos de todos los ficheros y directorios vitales, monitorizar el uso de cuentas privilegiadas, etc. El programa COPS, por ejemplo, sirve para comprobar nuestro sistema de ficheros y ficheros de configuración generales, en busca de permisos inusuales u otras anomalías. También es conveniente usar un sistema de claves que fuerce ciertas reglas en las claves de los usuarios que las hagan difíciles de adivinar. El sistema de claves ocultas (shadow password), por ejemplo, requiere que una clave tenga al menos cinco letras, entre las cuales se encuentren tanto mayúsculas como minúsculas, números y caracteres no−alfabéticos.

Cuando un servicio se hace accesible a la red, asegúrese de darle el menor privilegio. Esto significa, en una palabra que no se deberán permitir acciones que no son imprescindibles, para que se trabaje como se diseñó el servicio originalmente. Por ejemplo, el usuario debería hacer sus programas con setuid root, o alguna otra cuenta privilegiada, sólo si realmente se necesitara. También, si se quiere usar un servicio sólo para una aplicación muy limitada, el administrador del sistema no debe vacilar en configurar el servicio tan restrictivamente como la aplicación especial lo permita. Por ejemplo, si se quiere permitir a máquinas sin disco arrancar desde un nodo en especial, se debe facilitar el servicio TFTP (*Trivial File Transfer Protocol* de modo que se puedan obtener los ficheros de configuración básicos del directorio `/boot`. Sin embargo, cuando se usa sin restringir, TFTP permite a cualquier usuario de cualquier lugar del mundo leer cualquier fichero de su sistema. Si esto no es lo que desea, luego se debe restringir el servicio TFTP solamente al directorio `/boot`[25]

Pensando en la misma línea, se podría restringir ciertos servicios a usuarios que acceden desde ciertos nodos, digamos desde nuestra red local. En Capítulo 12, presentaremos **tcpd**, que hace esto para una variedad de aplicaciones de red. Se explorarán otros métodos más sofisticados para restringir el acceso a nodos o servicios particulares en Capítulo 9.

Otro punto importante a tener en cuenta es evitar software peligroso. Claro que cualquier software que se utilice puede resultar peligroso, dado que el software puede tener fallos que gente astuta pueda explotar para acceder a nuestro sistema. Cosas como ésta ocurren, y no hay protección segura contra ello. Este problema afecta al software libre y a productos comerciales por igual[26]. De cualquier modo programas que requieran privilegio especial son inherentemente más peligrosos que otros, ya que cualquier fallo aprovechable en éstos puede tener consecuencias desastrosas.[27] Si instala un programa setuid con propósitos de red, sea muy cuidadoso y no deje de leerse toda la documentación, de manera tal de no crear una brecha en la seguridad del sistema por accidente.

Otra fuente a considerar deberían ser aquellos programas que permiten registrarse en el sistema, o la ejecución de órdenes con autentificación limitada. Las órdenes **rlogin**, **rsh** y **rexec**, son muy útiles pero ofrecen un muy ligero método de autentificación para aquellos que hagan uso de ellas. Un método de autentificación se basa en la confianza del nombre del nodo llamado, el cual fue obtenido de un servidor de nombres, (se hablará de estos más adelante), que pudo haber sido falseado. Hoy en día, debería ser una práctica común el reemplazar completamente los comandos **r** con la colección de herramientas **ssh**. Las herramientas **ssh** usan un método de autentificación mucho más confiable, además de proporcionar otros servicios como encriptación y compresión.

Nunca se debería de olvidar que nuestras precauciónes pueden fallar, por muy cuidadosas que estas sean. Por eso se debería asegurar de que la detección de los posibles intrusos es relativamente rápida. Comprobar los ficheros de actividad es un buen comienzo, pero el intruso probablemente sea bastante listo, y borrará cualquier huella que haya dejado. Sin embargo, hay herramientas como **tripwire**, (escrito por Gene Kim y Gene Spafford), [28]que permite comprobar ficheros vitales del sistema para ver si sus contenidos o permisos han cambiado. **tripwire** realiza varias e intensas sumas de verificación (checksums) sobre estos ficheros y almacena los resultados en una base de datos. En las siguientes ejecuciones, se reevalúan y comparan dichas sumas de verificación con las almacenadas, detectándose así cualquier posible modificación.

Capítulo 2. Cuestiones sobre redes TCP/IP

En este capítulo volveremos a las decisiones sobre configuración que se deben tomar cuando se conecta una máquina Linux a una red TCP/IP, incluyendo el tema de las direcciones IP, los nombres de los puestos, y cuestiones sobre el encaminamiento. Este capítulo le enseñerá lo que necesita saber para entender lo que su sistema requiere, mientras que los siguientes capítulos cubren las herramientas que serán necesarias.

Para aprender más acerca de TCP/IP y las razones de su uso, diríjase al tercer volumen del libro *Internetworking with TCP/IP* de Douglas R. Comer (Prentice Hall). Si busca una guía más detallada del control de una red TCP/IP, vea *TCP/IP Network Administration* de Craig Hunt (O'Reilly).

2.1. Interfaces de red

Para ocultar la diversidad de hardware que puede usarse en un entorno de red, TCP/IP define una *interfaz* abstracta a través de la cual se accede a dicho hardware. Esta interfaz ofrece un conjunto de operaciones que son las mismas para todos los tipos de hardware y básicamente trata con el envío y la recepción de paquetes.

Tiene que estar presente en el núcleo la correspondiente interfaz para cada dispositivo periférico de red. Por ejemplo, las interfaces Ethernet se llaman en GNU/Linux con nombres como `eth0` y `eth1`; las interfaces PPP (discutido en Capítulo 8) se denominan `ppp0` y `ppp1`; y a las interfaces FDDI se le da nombres como `fddi0` y `fddi1`. Estos nombres de interfaz se usan para propósitos de configuración cuando se quiere especificar un dispositivo físico determinado en una orden de configuración, y no tienen significado más allá de este uso.

Antes de ser usada en una red TCP/IP, a una interfaz se le debe asignar una dirección IP que sirve como su identificador cuando se comunica con el resto del mundo. Esta dirección es distinta del nombre de interfaz mencionado anteriormente; si se compara una interfaz con una puerta, la dirección es como el número de la puerta.

Se pueden seleccióńar otros parámetros de dispositivo como el tamaño máximo de los datagramas que pueden ser procesados por una parte del hardware determinada, a lo que se le denomina *Maximum Transfer Unit* (MTU). Hay otros atributos que se introducirán más tarde. Afortunadamente, la mayoría de esos atributos tienen valores por defecto muy acertados.

2.2. Direcciones IP

Como se menciona en Capítulo 1, el protocolo de red IP utiliza direcciones formadas por números de 32 bits. Se le debe asignar un número único a cada máquina del entorno de red.[29] Si está haciendo funcionar una red local que no tiene tráfico TCP/IP con otras redes, puede asignar estos números de acuerdo con sus preferencias personales. Hay algunos rangos de direcciones IP que han sido reservadas para redes privadas. Estos rangos se listan en Tabla 2−1. De cualquier modo, los números para los sitios en Internet los asigna una autoridad central, el *Network Information Center* (NIC).[30]

Para facilitar la lectura, las direcciones IP se separan en cuatro números de ocho bits llamados *octetos*. Por ejemplo, quark.physics.groucho.edu tiene una dirección IP 0x954C0C04, que se escribe como 149.76.12.4. Este formato se denomina normalmente *notación de puntos divisorios*.

Otra razón para usar esta notación es que las direcciones IP se dividen en un número de *red*, que es contenido en el octeto principal, y un número de *puesto*, que es contenido en el resto. Cuando se solicita al NIC una dirección IP, no se le asignará una dirección para cada puesto individual que pretenda usar. En cambio, se le otorgará un número de red y se le permitirá asignar todas la direcciones IP válidas dentro de ese rango para albergar puestos en su red de acuerdo con sus preferencias.

El tamaño de la parte dedicada al puesto depende del tamaño de la red. Para complacer diferentes necesidades, se han definido varias clases de redes, fijando diferentes sitios donde dividir la dirección IP. Las clases de redes se definen en lo siguiente:

Clase A
: La clase A comprende redes desde 1.0.0.0 hasta 127.0.0.0. El número de red está contenido en el primer octeto. Esta clase ofrece una parte para el puesto de 24 bits, permitiendo aproximadamente 1,6 millones de puestos por red.

Clase B
: La clase B comprende las redes desde 128.0.0.0 hasta 191.255.0.0; el número de red está en los dos primeros octetos. Esta clase permite 16.320 redes con 65.024 puestos cada una.

Clase C
: Las redes de clase C van desde 192.0.0.0 hasta 223.255.255.0, con el número de red contenido en los tres primeros octetos. Esta clase permite cerca de 2 millones de redes con más de 254 puestos.

Clases D, E, y F
: Las direcciones que están en el rango de 224.0.0.0 hasta 254.0.0.0 son experimentales o están reservadas para uso con propósitos especiales y no especifican ninguna red. La IP Multicast, un servicio que permite trasmitir material a muchos puntos en una internet a la vez, se le ha asignado direcciones dentro de este rango.

Si volvemos al ejemplo del capítulo 1, encontraremos que 149.76.12.4, la dirección de quark, se refiere al puesto 12.4 en la red de clase B 149.76.0.0.

Habrá notado que no se permiten todos los valores posibles de la lista anterior para todos los octetos de la parte del puesto. Esto se debe a que los octetos 0 y 255 se reservan para propósitos especiales. Una dirección donde todos los bits de la parte del puesto son 0, se refiere a la red, y una dirección donde todos los bits de la parte del puesto son 1, se denomina *dirección de difusión*. Ésta se refiere simultáneamente a todos los puestos de la red específica. Así, 149.76.255.255 no es una dirección de puesto válida, pero se refiere a todos los puestos en la red 149.76.0.0.

Algunas direcciones de red se reservan para propósitos especiales. 0.0.0.0 y 127.0.0.0 son dos de estas direcciones. La primera se denomina *encaminamiento por defecto*, y la segunda es la *dirección loopback*. El encaminamiento por defecto tiene que ver con el camino por el que el IP encamina los datagramas.

La red 127.0.0.0 está reservada para el tráfico local IP del puesto. Normalmente, la dirección 127.0.0.1 se asignará a una interfaz especial del puesto, la *interfaz loopback*, que actúa como un circuito cerrado. Cualquier paquete IP enviado a esta interfaz por TCP o UDP le será devuelto a cualquiera de ellos como si simplemente hubiese llegado desde alguna red. Esto permite desarrollar y probar software de red aunque no se esté usando una red real. La red loopback también permite usar software de red en un puesto solitario. Puede que esto no sea tan infrecuente como parece; por ejemplo, muchos sitios UUCP no tienen conectividad con IP en absoluto, pero aún pueden querer ejecutar un sistema de noticias INN. Para un funcionamiento adecuado en GNU/Linux, INN requiere la interfaz loopback.

Algunos rangos de direcciones de cada una de las clases de red han sido reservados y designados como rangos de direcciones reservadas o privadas. Estas direcciones están reservadas para el uso de redes privadas y no son encaminadas en Internet. Son usadas normalmente por organizaciones con su propia intranet, pero incluso las redes pequeñas suelen encontrarlas útiles. Las direcciones de red reservadas se muestran en Tabla 2−1.

Tabla 2−1. Rangos de direcciones IP reservados para uso público

Clase	Redes
A	10.0.0.0 hasta 10.255.255.255
B	172.16.0.0 hasta 172.31.0.0
C	192.168.0.0 hasta 192.168.255.0

2.3. Resolución de direcciones

Ahora que sabe como se componen las direcciones IP, se estará preguntando cómo se usan en una red Ethernet o Token Ring para identificar los diferentes puestos. Después de todo, dichos protocolos tienen sus propias direcciones para identificar los puestos y éstas no tienen absolutamente nada en común con una dirección IP, ¿verdad? De acuerdo.

Se necesita un mecanismo para proyectar las direcciones IP en las direcciones de la red subyacente. Este mecanismo es el *Address Resolution Protocol* (ARP). De hecho, ARP no se limita a Ethernet o Token Ring, sino que también se usa en otros tipos de redes, tales como el protocolo de radio amateur AX.25. La idea básica del ARP es exactamente lo que la mayor parte de la gente haría si tuviese que encontrar al señor X en una multitud de 150 personas: la persona que le busca le llamaría lo suficientemente fuerte para que todo el mundo en la habitación pueda oírle, esperando que el señor X responda si está allí. Cuando él responda, sabremos qué persona es.

Cuando ARP quiere encontrar la dirección Ethernet correspondiente a una dirección IP dada, usa una característica Ethernet denominada *difusión*, en la cual un datagrama se envía simultáneamente a todas las estaciones de la red. El datagrama de difusión enviado por el ARP contiene la dirección IP en cuestión. Cada puesto receptor compara esta dirección con la suya propia y si coinciden, devuelve una respuesta ARP al puesto emisor. El puesto emisor puede entonces obtener la dirección Ethernet del remitente de la respuesta.

Se preguntará como un puesto puede localizar una dirección de Internet que puede estar en una red diferente al otro lado del mundo. La respuesta a esta pregunta tiene que ver con el *encaminamiento*, esto es, encontrar la ubicación física de un puesto en una red. Discutiremos esta cuestión más profundamente en la próxima sección.

Hablemos un poco más sobre ARP. Una vez que un puesto ha descubierto una dirección Ethernet, la guarda en su caché ARP de forma que no tiene que preguntar por ella de nuevo la próxima vez que quiera enviar un datagrama al puesto en cuestión. De cualquier modo, es poco aconsejable mantener esta información para siempre; la tarjeta Ethernet del puesto remoto puede ser reemplazada a causa de problemas técnicos, así la entrada ARP sería inválida. Por tanto, las entradas en la caché ARP son desechadas cada cierto tiempo para forzar otra búsqueda de la dirección IP.

A veces también es necesario encontrar la dirección IP asociada a una dirección Ethernet dada. Esto ocurre cuando una máquina sin disco necesita arrancar desde un servidor de la red, lo que es una situación común en redes de área local. Un cliente sin disco, de todos modos, virtualmente no tiene información de sí mismo¡excepto de su dirección Ethernet! De modo que difunde un mensaje que contiene una petición para que un servidor de arranque le otorgue una dirección IP. Hay otro protocolo para esta situación denominado *Reverse Address Resolution Protocol* (RARP). Junto al protocolo BOOTP, sirve para definir el proceso de arranque de clientes sin disco a través de la red.

2.4. Encaminamiento IP

Ahora nos ocuparemos del problema de encontrar el puesto al que se envían los datagramas basándose en la dirección IP. La diferentes partes de las direcciones se manejan de forma distinta; es su trabajo configurar los ficheros que indican como se trata cada parte.

2.4.1. Redes IP

Cuando escribe una carta a alguien, normalmente pone una dirección completa en el sobre especificando el país, la provincia y el código postal. Después la echa al buzón, la oficina de correos la hará llegar a su destino: se enviará al país indicado, donde el servicio nacional la enviará a la provincia y la región adecuada. La ventaja de este esquema jerarquizado es obvia: mande donde mande la carta, la oficina de correos local apenas debe conocer a qué dirección remitir la carta, es más, a la oficina no le importa por dónde viajará esta siempre que llegue al país de destino.

Las redes IP están estructuradas de forma similar. Toda Internet consiste en varias redes, denominadas *sistemas autónomos*. Cada sistema realiza un encaminamiento interno entre los puestos que lo forman, por lo que la tarea de remitir un datagrama se reduce a encontrar un camino a la red del puesto de destino. Tan pronto como el datagrama se entrega a *cualquier* puesto en esa red particular, el resto del proceso se realiza exclusivamente en la misma red.

2.4.2. Subredes

Esta estructura se refleja dividiendo la dirección IP en la parte del puesto y la de red, como se explicó anteriormente. Por defecto, la red de destino se obtiene a partir de la parte de red de la dirección IP. De este modo, los puestos con números idénticos de *red* IP deben encontrarse en la misma red.[31]

También tiene sentido proporcionar un esquema similar *dentro* de la red, ya que ésta puede constar de un grupo de cientos de redes más reducidas, con las unidades más pequeñas haciendo de redes físicas como Ethernets. Por lo tanto, IP permite subdividir una red IP en varias *subredes*.

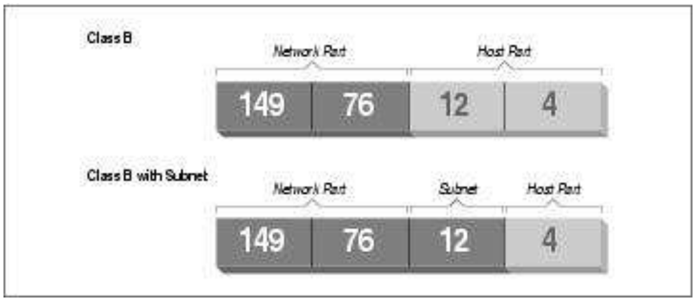
Una subred se responsabiliza de enviar datagramas a un cierto rango de direcciones IP. Esto es una extensión del concepto de dividir campos de bits, como en las clases A, B, y C. De cualquier forma, la parte de red se extiende ahora para incluir algunos bits de la parte del puesto. El número de bits que se interpreta como el número de subred viene dado por la llamada *máscara de subred* o *máscara de red*. Este es también un número de 32 bits, que especifica la máscara de bit para la parte de red de la dirección IP.

La red del campus de la Groucho Marx University es un ejemplo de este tipo de redes. Tiene un número de red de clase B 149.76.0.0, y su máscara de red es 255.255.0.0.

Internamente, la red del campus de la GMU consta de varias redes más pequeñas, como son las LANs de varios departamentos. De modo que el rango de direcciones IP se divide en 254 subredes: desde 149.76.1.0 hasta 149.76.254.0. Por ejemplo, al departamento de Física Teórica se le ha asignado 149.76.12.0. La dorsal del campus es una red por derecho propio, y se le ha asignado 149.76.1.0. Estas subredes comparten el mismo número de red, mientras que el tercer octeto se usa para distinguirlas entre sí. Utilizarán así una máscara de subred de 255.255.255.0.

Figura 2−1 muestra como 149.76.12.4, la dirección de quark, se interpreta de forma distinta cuando la dirección viene dada como una red de clase B ordinaria y cuando se usa como subred. >>>>>>> 1.7

Figura 2−1. División de una red de clase B en subredes



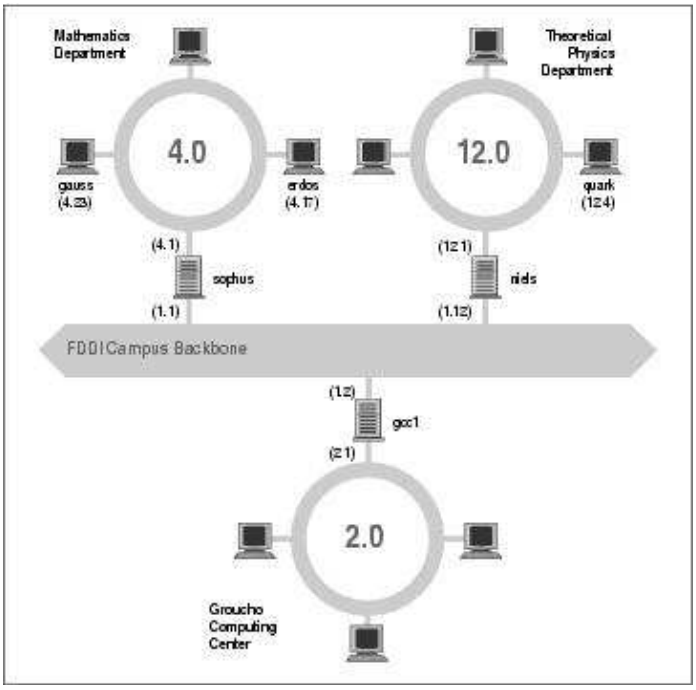
Es dificil notar que la técnica de generar subredes es únicamente una *división interna* de la red. Las subredes se generan por el propietario de la red (o el administrador). Frecuentemente, las subredes se crean para reflejar límites determinados, ya sean físicos (entre dos Ethernets), administrativos (entre dos departamentos), o geográficos (entre dos ubicaciones distintas), y la autoridad de cada subred se delega a alguna persona de contacto. De todos modos, la estructura afecta solo al funcionamiento interno de la red y es completamente invisible para el mundo exterior.

2.4.3. Pasarelas

La división en subredes no solo es un beneficio para la administración; es frecuentemente una consecuencia natural de límites de hardware. El alcance de un puesto en una red física determinada, tal como una Ethernet, es muy limitado: solo se puede comunicar con puestos de la red donde él mismo se encuentra. Solo se puede acceder a los demás puestos a través de máquinas con una utilidad especial denominadas *pasarelas*. Una pasarela es un puesto que está conectado simultáneamente a dos o más redes físicas y está configurado para intercambiar paquetes entre ellas.

Figura 2−2 muestra parte de la topología de red en la Groucho Marx University (GMU). Los puestos que están en dos subredes al mismo tiempo se muestran con ambas direcciones.

Figura 2−2. Parte de la topología de red de la Groucho Marx University



Redes físicas diferentes deben pertenecer a redes IP distintas para que IP sea capaz de reconocer si un puesto está en una red local o no. Por ejemplo, el número de red 149.76.4.0 se reserva para los puestos en la LAN de matemáticas. Cuando se manda un datagrama a quark, el software de red en erdos entiende inmediatamente por la dirección IP 149.76.12.4 que el puesto de destino está en una red física diferente, y por lo tanto sólo puede ser alcanzado a través de una pasarela (sophus por defecto).

sophus está conectada a dos subredes distintas: el departamento de Matemáticas y la dorsal del campus. Accede a cada una a través de una interfaz diferente, `eth0` y `fddi0`, respectivamente. Ahora bien, ¿que dirección IP se le debe asignar?. ¿Debemos darle una en la subred 149.76.1.0, o en la 149.76.4.0?

La respuesta es: ambas. sophus tiene asignadas la dirección 149.76.1.1 para su uso en la red 149.76.1.0 y la dirección 149.76.4.1 para la red 149.76.4.0. Una pasarela debe tener asignada una dirección IP para cada red a la que pertenezca. Estas direccionesjunto con la correspondiente máscara de redestán vinculadas a la interfaz por la que se accede a la subred. De modo que el esquema de interfaces y direcciones de sophus sería este:

Interfaz	**Dirección**	**Máscara de red**
`eth0`	149.76.4.1	255.255.255.0
`fddi0`	149.76.1.1	255.255.255.0
`lo`	127.0.0.1	255.0.0.0

La última entrada define la interfaz loopback `lo`, de la que hablamos anteriormente.

Generalmente, se puede ignorar la sutil diferencia entre destinar una dirección a un puesto o a su interfaz. En el caso de puestos que están solos en una red, como erdos, normalmente nos referiremos al puesto con "esta o aquella" dirección IP, aunque estrictamente hablando, sea la interfaz Ethernet la que tenga esa dirección IP. La distinción sólo es realmente importante en el caso de referirse a una pasarela.

2.4.4. Tabla de encaminamiento

Ahora nos centraremos en como IP elige qué pasarela usar para enviar un datagrama a una red remota.

Hemos visto que erdos, cuando envía un datagrama a quark, comprueba la dirección de destino y encuentra que ésta no está en la red local. erdos por lo tanto envía el datagrama a la pasarela por defecto sophus, que se enfrenta ahora al mismo problema. sophus reconoce que quark no está en ninguna de las redes a las que está conectada directamente, de modo que todavía tiene que encontrar otra pasarela a través de la cual remitirlo. La elección correcta debería ser niels, la pasarela del departamento de Físicas. Por lo tanto sophus necesita información para asociar una red de destino con una pasarela adecuada.

Para esta tarea, IP usa una tabla que asocia redes con las pasarelas por las que estas pueden ser alcanzadas. Generalmente, debe incluirse también una entrada que abarque todo (el *encaminamiento por defecto*); esta es la pasarela asociada a la red 0.0.0.0. Todas las direcciones de destino se corresponden con este encaminamiento, ya que no se requiere ninguno de los 32 bits para ajustarse a él, y por tanto los paquetes dirigidos a una red desconocida se envíarán al encaminamiento por defecto. En sophus, la tabla podría ser algo como esto:

Red	**Mascara de red**	**Pasarela**	**Interfaz**
149.76.1.0	255.255.255.0	–	`fddi0`
149.76.2.0	255.255.255.0	149.76.1.2	`fddi0`
149.76.3.0	255.255.255.0	149.76.1.3	`fddi0`
149.76.4.0	255.255.255.0	–	`eth0`
149.76.5.0	255.255.255.0	149.76.1.5	`fddi0`

0.0.0.0	0.0.0.0	149.76.1.2	`fddi0`

Si es necesario usar un encaminamiento a una red a la que sophus está conectada directamente, no se necesita una pasarela; en ese caso la columna de la pasarela contiene un guión.

El proceso que se sigue para identificar si una dirección de destino determinada corresponde con un encaminamiento es una operación matemática. Es bastante simple, pero requiere conocimientos de aritmética binaria y lógica: Un encaminamiento corresponde a un destino si la dirección de red operada lógicamente por medio de AND con la máscara de red es precisamente la dirección de destino operada lógicamente por medio de AND con la máscara de red.

Traducción: un encaminamiento corresponde si el número de bits de la dirección de red especificada por la máscara de red (empezando por el bit más a la izquierda, el orden más alto del byte uno de la dirección) corresponde al mismo número de bits en la dirección de destino.

Cuando la implementación de IP busca el mejor encaminamiento hasta un destino, puede que encuentre varias entradas que correspondan a la dirección del objetivo. Por ejemplo, sabemos que el encaminamiento por defecto corresponde a todos los destinos, pero los datagramas destinados a redes unidas localmente también corresponderán a su encaminamiento local. ¿Cómo IP decide que encaminamiento usar? Es aquí donde la máscara de red representa un papel importante. Mientras que los dos enrutamientos corresponden al destino, uno de ellos tiene una máscara de red mayor que la del otro. Se dijo anteriormente que la máscara de red se usa para dividir los espacios de las direcciones en redes más pequeñas. Cuanto mayor es una máscara de red, mejor especifica la correspondencia a la dirección de un objetivo; cuando se envían datagramas, siempre se debería elegir el enrutamiento que tenga la mayor máscara de red. El encaminamiento por defecto tiene una máscara de red de cero bits, y en la configuración mostrada anteriormente, las redes enlazadas localmente tienen una máscara de red de 24 bits. Si un datagrama corresponde a una de estas redes, será enrutado al dispositivo apropiado en vez de seguir el encaminamiento por defecto porque el enrutamiento de la red local corresponde a un mayor número de bits. Los únicos datagramas que se encaminan a través del encaminamiento por defecto son aquellos que no corresponden a ningún otro enrutamiento.

Se puede construir tablas de encaminamiento siguiendo distintos métodos. En el caso de LANs pequeñas, normalmente lo más eficiente es construirlas a mano y nutrirlas de IP usando el comando **route** en el momento del arranque (vease Capítulo 5). Para redes mayores, se construyen y ajustan en tiempo de ejecución por los *demonios de encaminamiento*; estos demonios corren en puestos centrales de la red e intercambian información de enrutamiento para calcular caminos óptimos entre los miembros de la red.

Dependiendo del tamaño de la red, se necesitará usar diferentes protocolos de encaminamiento. Para enrutar dentro de sistemas autónomos (tales como el campus de Groucho Marx) se usan los *protocolos de encaminamiento interno*. El más importante de éstos es el *Routing Information Protocol* (RIP), que es implementado por el demonio BSD **routed**. Para enrutar entre sistemas autónomos se tienen que usar *protocolos de encaminamiento externo* como *External Gateway Protocol* (EGP) o *Border Gateway Protocol* (BGP); estos protocolos, incluido RIP, han sido implementados en el demonio **gated** de la Universidad de Cornell.

2.4.5. Métrica de encaminamiento

Hay que contar con los encaminamientos dinámicos para elegir la mejor ruta hasta nuestro puesto o red de destino basándonos en el número de *saltos*. Los saltos son las pasarelas que un datagrama debe atravesar antes de llegar al puesto o la red. Cuanto más corta sea una ruta en mejor consideración la tendrá RIP. Las rutas muy largas con 16 saltos o más son consideradas como inusable y son descartadas.

RIP controla la información interna de encaminamiento de su red local, pero tiene que ejecutar **gated** en todos los puestos. En el momento del arranque, **gated** comprueba todas las interfaces de red activas. Si hay más de una interfaz activa (sin contar la interfaz loopback), asume que el puesto está intercambiando paquetes entre varias redes e intercambia y emite activamente información de encaminamiento. De cualquier forma, las actualizaciones de RIP solo se recibirán pasivamente y se pondrá al día la tabla de encaminamiento local.

Cuando se transmite información según la tabla de encaminamiento local, **gated** calcula la longitud de la ruta atendiendo al llamado *valor métrico* asociado a la entrada de la tabla de encaminamiento. Este valor métrico lo decide el administrador del sistema cuando configura el encaminamiento, y debe reflejar el valor actual de la ruta.[32] Por tanto, la métrica de la ruta a una subred a la que el puesto está directamente conectada debe ser siempre cero, mientras que una ruta que vaya a través de dos pasarelas debe tener una métrica de dos. De todos modos, no deberá preocuparse por las métricas si no usa **RIP** o **gated**.

2.5. El Internet Control Message Protocol

IP tiene otro protocolo complementario del que no hemos hablado todavía. Este es el *Internet Control Message Protocol* (ICMP), usado por el código de redes del núcleo para comunicar mensajes de error a otros puestos. Por ejemplo, asumiremos que nos encontramos en erdos otra vez y queremos hacer **telnet** al puerto 12345 en quark, pero no hay procesos escuchando en ese puerto. Cuando el primer paquete TCP para este puerto llegue a quark, la capa de red reconocerá esta llegada e inmediatamente enviará un mensaje ICMP a erdos empezando con Port Unreachable.[33]

El protocolo ICMP ofrece varios mensajes diferentes, muchos de ellos tratan con condiciones de error. De todas maneras, hay un mensaje muy interesante denominado mensaje Redirect[34]. Lo genera el módulo de encaminamiento cuando detecta que otro puesto está usándolo como pasarela, aunque exista una ruta más corta. Por ejemplo, después del arranque, la tabla de encaminamiento de sophus puede estar incompleta. Puede que contenga las rutas a la red de Matemáticas, a la dorsal FDDI, y el encaminamiento por defecto apuntando a la pasarela del Groucho Computing Center (gcc1). De este modo, los paquetes para quark serán enviados a gcc1 en vez de a niels, la pasarela del departamento de Fisicas. Cuando recibe un datagrama como éste, gcc1 notará que esa es una mala elección como ruta y reenviará el paquete a niels, mientras tanto envía un mensaje Redirect ICMP a sophus diciéndole la ruta superior.

Esta parece ser una forma muy inteligente de evitar la configuración manual de las rutas excepto las más básicas. De cualquier forma, hay que decir que depender de esquemas de encaminamiento dinámicos, sean RIP o mensajes Redirect ICMP, no es siempre una buena idea. Redirect ICMP y RIP ofrecen muy poca o ninguna capacidad de verificar si alguna información de encaminamiento es efectivamente auténtica. Esta situación permite a malévolos inútiles perturbar el desarrollo del tráfico de la red completa, o incluso algo peor. Consecuentemente, el código de red de GNU/Linux trata los mensaje Network Redirect como si fuesen Host Redirects. Esto minimiza los daños de un ataque restringiéndolos a sólo un puesto, en vez de la red completa. Por otro lado, esto significa que se generá un poco más de tráfico en las mismas condiciones, ya que cada puesto hace que se genere un mensaje Redirect ICMP. En la actualidad, se considera generalmente una mala costumbre depender de las redirecciones ICMP para algo.

2.6. Resolución de nombres de puesto

Como se describió anteriormente, las direcciones en una red TCP/IP, al menos en IP Version 4, giran alrededor de números de 32 bits. De modo que, tendrá que sufrir recordando más que unos pocos números de este tipo. Aunque los puestos se reconocen generalmente por nombres ordinarios tales como gauss o strange. Esto se convierte en el cometido de una aplicación que encuentra la dirección IP correspondiente a un nombre. A este proceso se le denomina *resolución de nombres de puesto*.

Cuando una aplicación necesita encontrar la dirección IP de un puesto dado, ésta delega en las funciones de biblioteca `gethostbyname(3)` y `gethostbyaddr(3)`. Tradicionalmente, estos y otros procedimientos relacionados están agrupados en una biblioteca separada denominada en GNU/Linux *resolverlibrary*, estas funciones son parte de la `libc` estándar. Coloquialmente, nos referimos por tanto a este conjunto de funciones como el sistema de resolución. La configuración del sistema de resolución de nombre se detalla en

Capítulo 6.

En una red pequeña como una Ethernet o incluso un grupo de Ethernets, no es muy difícil mantener tablas de asignación de nombres de puesto a direcciones. Esta información la mantiene normalmente un fichero llamado `/etc/hosts`. Cuando se añaden o se eliminan puestos, o se reasignan direcciones, lo único que se debe hacer es actualizar el fichero `hosts` en todos los puestos. Obviamente, esto puede resultar arduo en redes que cuenten con más de un puñado de equipos.

Una solución a este problema es el *Network Information System* (NIS), desarrollado por Sun Microsystems, coloquialmente llamado YP o Yellow Pages. NIS almacena el fichero `hosts` (y otra información) en una base de datos en un puesto principal de donde los puestos cliente pueden obtenerla según se necesite. Sin embargo, esta aproximación es adecuada sólo para redes de tamaño medio tales como LANs, ya que esto implica mantener centralmente la base de datos `hosts` al completo y distribuirla a los servidores. La instalación y la configuración del NIS se discute en detalle en Capítulo 13.

En Internet, la información de direcciones también fue inicialmente almacenada en una base de datos simple `HOSTS.TXT`. Este fichero se mantenía en el *Network Information Center* (NIC), y tenía que ser descargado e instalado por todos los sitios participantes. Cuando la red creció, surgieron varios problemas con el esquema. Además del gasto administrativo que suponía la instalación regular de `HOSTS.TXT`, la carga de los servidores que distribuía llegó a ser demasiedo grande. Más grave aún, todos los nombres tenían que estar registrado en el NIC, lo que aseguraría que ningún nombre se distribuyese dos veces.

Esta es la razón por la que se aprobó un nuevo esquema de resolución de nombres en 1994: el *Domain Name System*. DNS fue diseñado por Paul Mockapetris y aborda los dos problemas a la vez. El Domain Name System se discute en detalle en Capítulo 6.

Capítulo 3. Configuración del hardware de red

Hasta ahora, hemos estado hablando bastante sobre las interfaces de red pero sin explicar realmente qué es lo que pasa cuando el código de red del núcleo accede a una parte del hardware. Para ello, y antes que nada, tenemos que hablar un poco sobre los conceptos de interfaz y controladores.

Primero, evidentemente, está el hardware por sí mismo; por ejemplo, una tarjeta Ethernet, FDDI o Token Ring: es una oblea de silicio, atiborrada de montones de pequeños chips con extraños números encima e insertada en una ranura de su PC. Esto es lo que por lo general denominamos un dispositivo físico.

Para poder utilizar una tarjeta de red son necesarias una serie de funciones especiales definidas en el núcleo de Linux que seran capaces de entender la forma particular de acceso al dispositivo. Al software que implementa estas funciones se le llama *controlador*(N. del T.: Con frecuencia, la bibliografía especializada en español también los llama *manejadores* o *drivers*). Linux tiene controladores para muchos tipos de tarjetas de red: ISA, PCI, MCA, EISA, puerto paralelo, PCMCIA, y más recientemente, USB.

¿Pero qué es lo que queremos decir con que un controlador gestione un dispositivo? Vamos a tratar sobre esto con una tarjeta Ethernet. El controlador tiene que ser capaz de comunicarse de alguna forma con la lógica interna de la tarjeta: tiene que enviar órdenes y datos a la tarjeta, mientras que la tarjeta debe transmitir al controlador cualquier dato recibido.

En un PC compatible, esta comunicación se establece por medio de una serie de direcciones de E/S que son mapeadas a los registros de la tarjeta y/o a través de transferencias directas o compartidas a memoria. Todos las órdenes y datos que el núcleo envía a la tarjeta tienen que ir a estas direcciones. Las direcciones de memoria y E/S son obtenidas generalmente por medio del arranque o de las *direcciones base*. Las direcciones base típicas para las tarjetas Ethernet por bus ISA son `0x280` o `0x300`. Las tarjetas de red por bus PCI generalmente ya tienen asignada automáticamente su dirección de E/S.

Normalmente no hay que preocuparse por asuntos de hardware como las direcciones base porque al arrancar el núcleo intenta detectar la localización de la tarjeta. Esto es llamado *autoverificación* (N. del T.: Del inglés autoprobe), que significa que el núcleo lee varias posiciones de memoria y compara los datos que ha encontrado con los que esperaría ver si una tarjeta de red en concreto estuviese instalada en esa posición. De todas maneras, puede haber tarjetas de red que no puedan ser detectadas automáticamente; esto ocurre a veces con tarjetas de red baratas que no son réplicas exactas de tarjetas estándar de otros fabricantes. Por otro lado, el núcleo intentará detectar solamente un único dispositivo de red al arrancar. Si está usando más de una tarjeta, tendrá que informar al núcleo de las otras tarjetas explícitamente.

[35] Otro de los parámetros del que puede tener que informar al núcleo es la línea de petición de interrupción. Los componentes hardware normalmente interrumpen al núcleo cuando tienen la necesidad de que éste se ocupe de ellos, por ejemplo, cuando han llegado datos o se presenta una condición especial. En un bus ISA, las interrupciones pueden ocurrir en uno de los 15 canales de interrupción numerados asi: 0, 1, y del 3 al 15. Al número de interrupción asignado a un componente hardware se le denomina número de *petición de interrupción* (IRQ)..[36]

Como se describe en Capítulo 2, el núcleo accede a un dispositivo mediante lo que llamamos un *interfaz*. Los interfaces ofrecen un conjunto abstracto de funciones que es el mismo para todo tipo de hardware. Por ejemplo, las funciones para enviar o recibir datagramas.

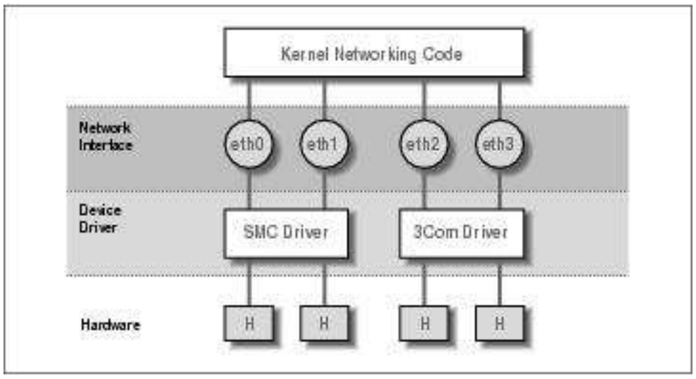
Los interfaces se identifican por medio de nombres. En muchos sistemas operativos tipo Unix, el interfaz de red se implementa como un fichero de dispositivo especial en el directorio `/dev/`. Si usted teclea la orden `ls -las /dev/`, verá como aparecen sus ficheros de dispositivos. En la columna de permisos de los ficheros (segunda) verá que los ficheros de dispositivos comienzan con una letra en vez del guión visto con los ficheros normales. Este carácter indica el tipo de dispositivo. Los tipos de dispositivos más comunes son los `b`, que indica que es un dispositivo de *bloque* y maneja grandes bloques de datos cada vez que lee y escribe, y `c`, que indica que el dispositivo es un dispositvo de *carácter* y maneja datos de un solo carácter cada vez. Donde normalmente desearía ver el tamaño del fichero en la salida de **ls**, en vez de eso verá dos números, llamados los números de dispositivo "major" y "minor" (primario y secundario). Estos números indican el dispositivo actual al que está asociado el fichero de dispositivo.

Cada controlador de dispositivo registra un unico número primario para el núcleo. En cada *caso* los registros de dispositivos tienen un único número secundario para dicho dispositivo primario. Los interfaces `tty`,`/dev/tty*`, son unos dispositivos de modo carácter por lo que indica la `c`, y tienen un maximo número de `4`, pero `/dev/tty1` tiene un número menor de `1`, y `/dev/tty2` tiene un número menor de `2`. Los ficheros de dispositivos son muy útiles para muchos tipos de dispositivos, pero pueden ser pesados de usar cuando intentamos encontrar un dispositivo sin usar para abrir.

Los nombres de las interfaces de Linux son definidos internamente en el núcleo y no son ficheros de dispositivos del directorio `/dev`. Algunos nombres de dispositivos típicos serán listados después en Sección 3.2. La asignación de interfaces a los dispositivos depende normalmente del orden en que los dispositivos son configurados. Por ejemplo, la primera tarjeta Ethernet instalada será `eth0`, la siguiente `eth1`, y así sucesivamente. Las interfaces SLIP son manejadas de forma diferente a otras porque éstas son asignadas dinámicamente. Cuando se establece una conexion SLIP, una interfaz es asignada al puerto serie.

Figura 3−1 Ilustra la relación entre el hardware, los controladores de dispositivos, y las interfaces.

Figura 3−1. Relación entre controladores, interfaces, y hardware



Al arrancar, el núcleo muestra los dispositivos detectados y las interfaces que instala. Lo siguiente es un extracto de la pantalla de arranque:

```
  .
  .  This processor honors the WP bit even when in supervisor mode./
     Good.
Swansea University Computer Society NET3.035 for Linux 2.0
NET3: Unix domain sockets 0.13 for Linux NET3.035.
Swansea University Computer Society TCP/IP for NET3.034
IP Protocols: IGMP,ICMP, UDP, TCP
Swansea University Computer Society IPX 0.34 for NET3.035
IPX Portions Copyright (c) 1995 Caldera, Inc.
Serial driver version 4.13 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16550A
tty01 at 0x02f8 (irq = 3) is a 16550A
CSLIP: code copyright 1989 Regents of the University of California
PPP: Version 2.2.0 (dynamic channel allocation)
PPP Dynamic channel allocation code copyright 1995 Caldera, Inc.
PPP line discipline registered.
eth0: 3c509 at 0x300 tag 1, 10baseT port, address 00 a0 24 0e e4 e0,/
    IRQ 10.
3c509.c:1.12 6/4/97 becker@cesdis.gsfc.nasa.gov
```

```
    Linux Version 2.0.32 (root@perf) (gcc Version 2.7.2.1)
    #1 Tue Oct 21 15:30:44 EST 1997
     .
     .
```

Este ejemplo muestra que el núcleo ha sido compilado con el TCP/IP activado e incluyendo controladores para SLIP, CSLIP, y PPP. La tercera línea empezando desde abajo muestra que una tarjeta Ethernet 3C509 ha sido detectada e instalada como la interfaz `eth0`. Si tiene algún otro tipo de tarjeta de red; quizás un adaptador de bolsillo D−Link, por ejemploel núcleo normalmente mostrará una línea que empieza con el nombre del dispositivo`dl0` en el caso del ejemplo del D−Linkseguido por el tipo de tarjeta detectada. Si tiene una tarjeta de red instalada pero no aparece ningún mensaje similar significa que el núcleo es incapaz de detectar su tarjeta correctamente. Esta situación será tratada más adelante en la secciónEthernet Autoprobing.

3.1. Configuracion del núcleo

Muchas distribuciones de GNU/Linux vienen con discos de arranque que funcionan con el hardware para PC más común. Normalmente, el núcleo suministrado es altamente modulable e incluye casi cualquier controlador que pueda necesitar. Esta es una gran idea para los discos de arranque, pero no es lo que usted probablemente quiera para un uso a largo plazo. No es un buen sistema tener almacenados controladores en su disco que nunca va a usar. Por lo tanto, será conveniente crear su propio núcleo e incluir sólo aquellos controladores que realmente necesite o desee; de esta forma ahorrará un poco de espacio en disco y reduce el tiempo que lleva compilar un núcleo nuevo.

En cualquier caso, al trabajar con un sistema GNU/Linux, le deberá ser familiar la compilación de un núcleo. Piense en esto como si fuera un tránsito, una afirmación de una de las cosas que hace al software libre más poderoso de lo que ya es usted tiene las fuentes. Este no es un caso de, tengo que compilar un núcleo, más bien es el caso de, *puedo* compilar un núcleo. Los conceptos básicos de la compilacion de un núcleo Linux se explican en la Guía de Matt Welsh: *"instalación y primeros pasos"*, que también forma parte de la serie del Proyecto de Documentación de GNU/Linux. Por tanto, en esta sección solo trataremos las opciones de configuración que afectan a la red.

Un punto importante que vamos a repetir aquí es la forma en que funciona el esquema de numeración de la versión del núcleo. Los núcleos Linux son numerados en el siguiente formato: `2.2.14`. El primer dígito indica el número de versión *primario*. Este dígito cambia cuando hay cambios numerosos y significativos en el diseño del núcleo. Por ejemplo, el núcleo cambio del 1 al 2 cuando obtuvo soporte para máquinas de diferente arquitectura a la Intel x86 (la del PC). El segundo número es el número de versión *secundario*. En muchos aspectos, este número es el más importante a tener en cuenta. La comunidad de desarrolladores de Linux ha adoptado un estándar en el cual un número de versión secundario *par* indica que el núcleo está en *producción*, o es *estable*, y un número de versión secundario *impar* indica que el núcleo esta en *desarrollo*, o es *inestable*. Debe usar los núcleos estables para los equipos importantes, ya que han sido comprobados más a fondo. Los núcleos en desarrollo son los que debe de usar si está interesado en experimentar con las últimas características de Linux., pero éstos pueden tener muchos problemas que todavía no han sido corregidos. El tercer número es simplemente un incremento por cada liberación de una versión secundaria.[37]

Al ejecutar **make menuconfig**, aparecerá un menú de texto que le mostrará una lista de cuestiones sobre la configuración, como por ejemplo, si desea usar la emulación del coprocesador matemático en el núcleo. Una de esas cuestiones pregunta si desea soporte para redes TCP/IP. Debe contestar con `y` para que el núcleo sea capaz de trabajar con redes TCP/IP.

3.1.1. Opciones del Núcleo Linux 2.0 y superiores

Después de completar la sección de opciones generales, se le preguntará si quiere incluir soporte para varios tipos de dispositivos, como controladoras SCSI o tarjetas de sonido. El cursor le indicará qué opciones están disponibles. Puede pulsar `?` para obtener una descripción de la opción en la que se encuentre. Siempre tiene la opción de sí (`y`) para incluir dicho componente de forma estática en el núcleo, o no (`n`) para excluir el componente completamente. Aparte puede ver la opción de módulo (m) para que dicho componente sea compilado como un módulo cargable. Los módulos necesitan ser cargados antes para que puedan ser usados, ésto es útil para controladores de componentes que no usa muy a menudo.

La siguiente lista de preguntas trata sobre el soporte de red. El juego exacto de opciones de configuración cambia constantemente debido al continuo desarrollo. Una lista típica de las opciones ofrecidas por la mayoría de las versiones del núcleo en torno a las 2.0 y 2.1 puede ser ésta:

```
*
* Network device support
*
Network device support (CONFIG_NETDEVICES) [Y/n/?]
```

Debe responder a esta cuestión con y si quiere usar *cualquier* tipo de dispositivo de red, ya sea Ethernet, SLIP, PPP, o el que sea. Cuando conteste a la pregunta cony, el soporte para los dispositivos Ethernet será activado automáticamente. Deberá responer a otras preguntas si quiere habilitar el soporte de otros tipos de controladores de red:

```
PLIP (parallel port) support (CONFIG_PLIP) [N/y/m/?] y
PPP (point-to-point) support (CONFIG_PPP) [N/y/m/?] y
*
* CCP compressors for PPP are only built as modules.
*
SLIP (serial line) support (CONFIG_SLIP) [N/y/m/?] m
CSLIP compressed headers (CONFIG_SLIP_COMPRESSED) [N/y/?] (NEW) y
Keepalive and linefill (CONFIG_SLIP_SMART) [N/y/?] (NEW) y
Six bit SLIP encapsulation (CONFIG_SLIP_MODE_SLIP6) [N/y/?] (NEW) y
```

Estas cuestiones conciernen a varios protocolos de la capa de enlace que Linux soporta. PPP y SLIP le permiten transportar datagramas IP a través de líneas serie. PPP es usado actualmente por un grupo de protocolos para enviar el tráfico de la red a través de líneas serie. Algunos de los protocolos que forman PPP gestionan la manera de poderse autentificar en el servidor, mientras otros gestionan el modo en que ciertos protocolos son transportados por el enlacePPP no esta limitado a transportar solo datagrama TCP/IP; además de este protocolo también puede transportar otros como el IPX.

Si responde y o m al soporte para SLIP, le serán preguntadas tres cuestiones que trataremos más abajo. La opción de comprimir las cabeceras permite el soporte para CSLIP, una técnica que comprime las cabeceras TCP/IP a solo 3 pequeños bytes. Recuerde que esta opción del núcleo no activa automáticmente el CSLIP; simplemente provee las funciones necesarias al núcleo para ello. La opción `Keepalive and linefill` causa que el soporte de SLIP genere periódicamente actividad en la línea para prevenir que ésta sea desconectada por inactividad. La opción `Six bit SLIP encapsulation` le permite ejecutar SLIP sobre líneas y circuitos que no son capaces de transmitir el grupo de 8−bit de datos correctamente. Esto es similar al uuencoding o la técnica binhex usada para enviar ficheros binarios por e−mail.

PLIP proporciona una forma de enviar datagramas IP a través de una conexión por puerto paralelo. Esto es usado comúnmente para comunicarse con PCs que usan DOS. En el hardware tíipico de PC, PLIP puede ser más rápido que PPP o SLIP, pero requiere mucha más CPU para funcionar, además cuando la tasa de transferencia sea buena, otras tareas en la máquina podrían volverse más lentas.

A medida que se desarrollan más controladores, la lista de preguntas en esta sección se hace mayor. Si desea compilar un núcleo que se pueda usar en varias máquinas, o si su máquina tiene más de un tipo de tarjeta de red instalada, puede activar más de un controlador:

```
 .
 .
Ethernet (10 or 100Mbit) (CONFIG_NET_ETHERNET) [Y/n/?]
3COM cards (CONFIG_NET_VENDOR_3COM) [Y/n/?]
3c501 support (CONFIG_EL1) [N/y/m/?]
3c503 support (CONFIG_EL2) [N/y/m/?]
```

```
    3c509/3c579 support (CONFIG_EL3) [Y/m/n/?]
    3c590/3c900 series (592/595/597/900/905) "Vortex/Boomerang" support/
        (CONFIG_VORTEX) [N/y/m/?]
    AMD LANCE and PCnet (AT1500 and NE2100) support (CONFIG_LANCE) [N/y/?]
    AMD PCInet32 (VLB and PCI) support (CONFIG_LANCE32) [N/y/?] (NEW)
    Western Digital/SMC cards (CONFIG_NET_VENDOR_SMC) [N/y/?]
    WD80*3 support (CONFIG_WD80x3) [N/y/m/?] (NEW)
    SMC Ultra support (CONFIG_ULTRA) [N/y/m/?] (NEW)
    SMC Ultra32 support (CONFIG_ULTRA32) [N/y/m/?] (NEW)
    SMC 9194 support (CONFIG_SMC9194) [N/y/m/?] (NEW)
    Other ISA cards (CONFIG_NET_ISA) [N/y/?]
    Cabletron E21xx support (CONFIG_E2100) [N/y/m/?] (NEW)
    DEPCA, DE10x, DE200, DE201, DE202, DE422 support (CONFIG_DEPCA) [N/y/m/?]/
        (NEW)
    EtherWORKS 3 (DE203, DE204, DE205) support (CONFIG_EWRK3) [N/y/m/?] (NEW)
    EtherExpress 16 support (CONFIG_EEXPRESS) [N/y/m/?] (NEW)
    HP PCLAN+ (27247B and 27252A) support (CONFIG_HPLAN_PLUS) [N/y/m/?] (NEW)
    HP PCLAN (27245 and other 27xxx series) support (CONFIG_HPLAN) [N/y/m/?]/
        (NEW)
    HP 10/100VG PCLAN (ISA, EISA, PCI) support (CONFIG_HP100) [N/y/m/?] (NEW)
    NE2000/NE1000 support (CONFIG_NE2000) [N/y/m/?] (NEW)
    SK_G16 support (CONFIG_SK_G16) [N/y/?] (NEW)
    EISA, VLB, PCI and on card controllers (CONFIG_NET_EISA) [N/y/?]
    Apricot Xen-II on card ethernet (CONFIG_APRICOT) [N/y/m/?] (NEW)
    Intel EtherExpress/Pro 100B support (CONFIG_EEXPRESS_PRO100B) [N/y/m/?]/
        (NEW)
    DE425, DE434, DE435, DE450, DE500 support (CONFIG_DE4X5) [N/y/m/?] (NEW)
    DECchip Tulip (dc21x4x) PCI support (CONFIG_DEC_ELCP) [N/y/m/?] (NEW)
    Digi Intl. RightSwitch SE-X support (CONFIG_DGRS) [N/y/m/?] (NEW)
    Pocket and portable adaptors (CONFIG_NET_POCKET) [N/y/?]
    AT-LAN-TEC/RealTek pocket adaptor support (CONFIG_ATP) [N/y/?] (NEW)
    D-Link DE600 pocket adaptor support (CONFIG_DE600) [N/y/m/?] (NEW)
    D-Link DE620 pocket adaptor support (CONFIG_DE620) [N/y/m/?] (NEW)
    Token Ring controlador support (CONFIG_TR) [N/y/?]
    IBM Tropic chipset based adaptor support (CONFIG_IBMTR) [N/y/m/?] (NEW)
    FDDI controlador support (CONFIG_FDDI) [N/y/?]
    Digital DEFEA and DEFPA adapter support (CONFIG_DEFXX) [N/y/?] (NEW)
    ARCnet support (CONFIG_ARCNET) [N/y/m/?]
      Enable arc0e (ARCnet "Ether-Encap" packet format) (CONFIG_ARCNET_ETH)/
          [N/y/?] (NEW)
      Enable arc0s (ARCnet RFC1051 packet format) (CONFIG_ARCNET_1051)/
          [N/y/?] (NEW)
      .
      .
```

Finalmente, en la sección de los sistemas de ficheros, el script de configuración le preguntará si desea soporte para NFS (networking file system), el sistema de ficheros en red. NFS le permite exportar sistemas de ficheros en varios hosts, haciendo que los ficheros aparezcan como si estuvieran en un disco duro normal y corriente conectado al host.

```
    NFS file system support (CONFIG_NFS_FS) [y]
```

Describiremos el NFS con detalle en Capítulo 14.

3.1.2. Opciónes de red del núcleo de GNU/Linux 2.0.0 and Higher

Linux 2.0.0 marcó un cambio significativo en el trabajo en red de GNU/Linux. Muchas características formaron parte estándar del núcleo, como el soporte para IPX. También fueron añadidas y hechas configurables un buen número de opciones. Muchas de esas opciones son usadas sólo en circunstancias muy especiales y no vamos a tratarlas en detalle. El Redes COMO tratará probablemente lo que no está tratado aquí. Vamos a listar unas cuantas opciones útiles en esta sección, y explicaremos cuando debe usar cada una.

Basics

Para trabajar en redes TCP/IP, debe contestar a esta cuestión con `y`. Aun contestando con `n` podra compilar el núcleo con soporte para IPX.

```
    Networking options  --->
        [*] TCP/IP networking
```

Pasarelas

Tiene que activar esta opción si su sistema va a actuar como una pasarela entre dos redes, entre una red y un enlace SLIP, etc.. Aunque no cuesta nada activarla, así lo está por omisión, podría querer desactivarla para configurar el nodo como un *cortafuegos*. Los cortafuegos(o firewalls) son nodos que están conectados a dos o más redes, pero que no enrutan el tráfico entre ellas. Son comúnmente usados para proporcionar a los usuarios acceso a Internet con un riesgo mínimo para la red interna. Los usuarios tienen permitido acceder al cortafuegos y usar los servicios de Internet, pero las máquinas de la compañía estan protegidas frente ataques del exterior porque las conexiones entrantes no pueden atravesar el cortafuegos (los cortafuegoss son tratados con más detalle en Capítulo 9):

```
        [*] IP: forwarding/gatewaying
```

Virtual hosting

Estas opciones permiten configurar más de una dirección IP para una única interfaz. Esto es útil si quiere hacer virtual hosting, (alojamiento virtual), con una sola máquina que puede ser configurada para escuchar y actuar como si esta fuera varias máquinas separadas entre sí, cada una con su propia configuración de red. Más adelante hablaremos acerca del IP aliasing:

```
        [*] Network aliasing
        <*> IP: aliasing support
```

Accounting

Esta opción le permite recolectar los datos en el caudal del tráfico IP enviados o recibidos en su máquina (trataremos esto con más detalle en Capítulo 10):

```
        [*] IP: accounting
```

PC hug

Esta opción evita incompatibilidades con algunas versiones de PC/TCP, una implementación comercial de TCP/IP basada en DOS para PCs. Si activa esta opción, todavía será capaz de comunicarse con máquinas UNIX normales, pero bajará el rendimiento cuando el enlace sea lento.

```
        --- (it is safe to leave these untouched)
        [*] IP: PC/TCP compatibility mode
```

Diskless booting

Esta función activa el *Protocolo de Resolución de Direcciones Inverso* (RARP). RARP se utiliza en clientes sin disco y terminales X para pedir su dirección IP al arrancar. Deberá activar RARP si planea ofrecer este tipo de servicios. Un pequeño programa llamado **rarp**, incluido con las utilidades de red estándares, se usa para añadir entradas a la tabla RARP del núcleo:

```
        <*> IP: Reverse ARP
```

MTU

Cuando enviamos datos sobre TCP, el núcleo tiene que dividir éstos en varios bloques de datos para pasarlos al nivel IP. El tamaño de estos bloques es llamado la *Unidad Maxima de Transmisión* (Maximum Transmission Unit), o MTU. Para los nodos accesibles a través de una red local como una Ethernet, se usa un MTU tan alto como la máxima longitud permitida para los paquetes Ethernet 1,500 bytes. Cuando enrutamos IP sobre una WAN como Internet, es preferible usar datagramas de menor tamaño para asegurarnos de que no necesitan ser partidos de nuevo a lo largo de la ruta mediante el proceso llamado *fragmentación IP* .[38] El núcleo es capaz de determinar automáticamente el MTU más bajo de una ruta IP y configurar automáticamente una conexión TCP para usar éste. Este comportamiento es activado por defecto. Si contesta con `y` a esta opción, esta característica será deshabilitada.

Si desea usar un tamaño de paquete menor para enviar datos a nodos específicos (porque, por ejemplo, los datos irán a través de un enlace SLIP), puede hacer ésto usando la opción **mss** de la orden **route**, que está descrita brevemente al final del capítulo:

```
    [ ] IP: Disable Path MTU Discovery (normally enabled)
```

Security feature

El protocolo IP soporta una característica llamada *Source Routing* Source Routing le permite especificar la ruta que un datagrama debe seguir mediante la grabación por usted mismo de la ruta dentro del datagrama. Esto fue alguna vez útil antes de que los protocolos de enrutamiento como RIP y OSPF se hicieran usuales. Pero hoy día es considerado una amenaza de seguridad debido a que puede facilitar a los atacantes inteligentes una forma de rodear ciertos tipos de cortafuegos evitando la tabla de enrutamiento de un enrutador. Normalmente deseará filtrar la procedencia de los datagramas externos enrutados, por lo que esta opción está activada normalmente.

```
    [*] IP: Drop source routed frames
```

Novell support

Esta opción activa el soporte para IPX, el protoco de transmisión para redes que usa Novell. GNU/Linux podría funcionar fácilmente como un enrutador IPX y su soporte es útil en entornos donde tiene servidores de ficheros Novell. El sistema de ficheros NCP también requiere tener el soporte de IPX activado en el núcleo; si desea añadir o montar sus sistemas de ficheros Novell deberá activar esta opción (hablaremos más sobre IPX y el sistema de ficheros NCP en Capítulo 15):

```
    <*> The IPX protocol
```

Amateur radio

Estas tres opciones seleccionadas dan soporte para los tres protocolos de Radio soportados por Linux: AX.25, NetRom y Rose (No vamos a describirlos en este libro, pero puede encontrar más informacion en el AX25 HOWTO):

```
        <*> Amateur Radio AX.25 Level 2
        <*> Amateur Radio NET/ROM
        <*> Amateur Radio X.25 PLP (Rose)
```

Linux soporta otro tipo de controlador: el controlador vacío (dummy). la siguiente pregunta aparece hacia el comienzo de la sección de controladores de dispositivos:

```
        <*> Dummy net controller support
```

El controlador vacío no hace realmente gran cosa, pero es bastante útil en nodos aislados o conectados mediante PPP/SLIP. Es básicamente una interfaz enmascarada del bucle local. En nodos que tienen PPP/SLIP pero no otras interfaces de red, es necesario tener una interfaz que contínuamente gestione las direcciones IP. Ésto se tratará con más detalle en Sección 5.7.7 en Capítulo 5. Recuerde que actualmente puede obtener el mismo resultado usando la característica IP alias y configurando sus direcciones IP como alias en la interfaz de bucle local.

3.2. Un vistazo a los dispositivos de red de Linux

El núcleo Linux soporta multitud de controladores para varios tipos de hardware. En esta sección daremos un breve repaso de las familias de controladores disponibles y los nombres de interfaz que usan.

Hay un conjunto de nombres estándares para los interfaces en GNU/Linux que se enumeraran a continuación. La mayoría de los controladores soportan más de un interfaz, en cuyo caso los interfaces son numeradas, como en `eth0` y `eth1`:

`lo`
: Éste es el interfaz de bucle local (loopback). Es usado para realizar tests, y para un par de aplicaciones de red. Funciona como un circuito cerrado que devuelve cualquier datagrama recibido a la capa de red del host. Siempre hay un dispositibo loopback presente en el núcleo y no tiene mucho sentido tener más.

`eth0`, `eth1`,
: Éstos son los interfaces de las tarjetas Ethernet. Son unsados para la mayoría de las tarjetas Ethernet, incluyendo algunas de las tarjetas Ethernet por puerto paralelo.

`tr0`, `tr1`,
: Éstos son las interfaces de las tarjetas Token Ring. Son usados para la mayoría de las tarjetas Token Ring, incluyendo tarjetas que no han sido fabricadas por IBM.

`sl0`, `sl1`,
: Éstos son los interfaces SLIP. Los interfaces SLIP se asocian a líneas serie en el orden en el que son instaladas

`ppp0`, `ppp1`,
: Éstos son los interfaces PPP. Como ocurre con los interfaces SLIP, cada interfaz PPP se asocia a una línea serie una vez convertida al modo PPP.

`plip0`, `plip1`,
: Éstas son los interfaces PLIP. PLIP transporta datagramas IP sobre líneas paralelas. Los interfaces son asignados por el controlador PLIP al arrancar el sistema y son mapeadas a los puertos paralelos. En los núcleos *2.0.x* hay una relación directa entre el nombre del dispositivo y el puerto E/S del puerto paralelo, pero en los núcleos posteriores los nombres de dispositivos son asignados secuencialmente, como para los dispositivos SLIP y PPP.

`ax0`, `ax1`,
: Estas son las interfaces AX.25. AX.25 es el principal protocolo usado por los radioaficionados. Las interfaces AX.25 son asignadas y mapeadas en modo similar a los dispositivos SLIP.

Hay muchos otros tipos de interfaces disponibles para otros controladores de red. Nosotros solo mostramos los más comunes.

En las siguientes secciones, trataremos en detalle el uso de los controladores descritos anteriormente. El Networking HOWTO ofrece más detalles sobre como configurar la mayoría de los que faltan aquí y el AX25 HOWTO explica cómo configurar los dispositivos de red para radioaficionados.

3.3. Instalación de una Ethernet

Las versiones actuales de GNU/Linux soportan una gran variedad de tarjetas Ethernet. La mayoria de los controladores fueron escritos por Donald Becker, que es el autor de los controladores para una familia de tarjetas basadas en el chip de National Semiconductor 8390; éstos son conocidos como las Series de Controladores de Becker. Aunque muchos desarrolladores también han contribuido, actualmente hay algunas tarjetas Ethernet comunes que no están soportadas por GNU/Linux. La lista de las tarjetas Ethernet soportadas crece continuamente, así que si su tarjeta de red no esta soportada ahora, pronto lo estará.

Algunas veces en la temprana historia de GNU/Linux hemos intentado hacer un listado de todas las tarjetas Ethernet soportadas, pero esto podría llevarnos mucho tiempo y espacio. Afortunadamente, Paul Gortmaker mantiene una lista con cada una de las tarjetas soportadas y el método para hacerlas funcionar bajo GNU/Linux, [39] Éste es enviado mensualmente al grupo de noticias comp.os.linux.answers, y también está disponible en cualquiera de las réplicas de la web del Linux Documentation Project.

Aun si está seguro de saber como instalar una tarjeta Ethernet particular en su máquina, a menudo merece la pena echar un vistazo a lo que pone en el Ethernet HOWTO. Podrá encontrar mucha información a parte de los simples asuntos de configuración. Por ejemplo, puede ahorrarle un montón de dolores de cabeza conocer el comportamiento de muchas tarjetas Ethernet basadas en DMA que usan el mismo canal DMA que la controladora SCSI Adaptec 1542 por omisión. Si no cambia una de ellas a un canal DMA diferente puede terminar con la tarjeta Ethernet escribiendo paketes de datos en lugares al azar de su disco duro.

Para usar cualquiera de las tarjetas Ethernet soportadas con GNU/Linux, debe usar un núcleo precompilado procedente de alguna de las principales distribuciones de GNU/Linux. Estos generalmente tienen modulos disponibles para todos los controladores soportados, y el proceso de instalación normalmente permite seleccionar qué controladores quiere cargar. A largo plazo, sin embargo, es mejor contruirse su propio núcleo y compilarlo sólo con los controladores que necesita; esto ahorra espacio y memoria.

3.3.1. Ethernet Autoprobing

Muchos de los controladores Ethernet de GNU/Linux son lo suficientemente listos para saber como encontrar a su tarjeta Ethernet. Esto le ahorra tener que decirle al núcleo dónde está. El Ethernet HOWTO tiene un listado donde pone qué controladores usan autoverificación y en qué orden buscan por las direcciones E/S a la tarjeta.

Hay tres limitaciones en el código de autoverificación. Primero, éste no reconoce bien todas las tarjetas. Esto es especialmente cierto para algunos clones de tarjetas habituales. Segundo, el núcleo no autocomprobará para buscar más de una tarjeta a no ser que se le ordene. Esto fue una concisa decisión de diseño, asumiendo que se quería tener el control sobre qué tarjeta es asignada a cada interfaz. La mejor manera de hacer esto con seguridad es configurar manualmente cada tarjeta Ethernet en su máquina. Tercero, el controlador puede que no busque en las direcciones en que su tarjeta está configurada. Generalmente hablando, los controladores autocomprobarán en las direcciones en que el dispositivo en particular es capaz de ser configurado, pero algunas veces ciertas direcciones son ignoradas para evitar conflictos de hardware con otros tipos de tarjetas que usan normalmente la misma dirección.

Las tarjetas de red PCI suelen ser detectadas correctamente. Pero si está usando más de una tarjeta, o si la autodetección falla, tiene una forma de decirle al núcleo la dirección base y el nombre de la tarjeta.

En el arranque puede dar al núcleo información y mandatos que cualquiera de los componentes de éste leerán. Este mecanismo le permite enviar información al núcleo que el controlador Ethernet pueda usar para localizar a su tarjeta Ethernet o hacer que la detecte.

Si usa lilo para arrancar, puede enviarle al núcleo parámetros especificándolos a través de la opción append en el fichero `lilo.conf`. Para informar al núcleo acerca de un dispositivo Ethernet puede escribir los siguientes parámetros:

```
ether=irq,base_addr,[param1,][param2,]name
```

Los primeros cuatro parámetros son numéricos, mientras que el último es el nombre del dispositivo. Los parámetros *`irq`*, *`base_addr`*, y *`name`* son necesarios, pero los dos parámetros *`param`* son opcionales. Si cualquiera de los valores numéricos es puesto a cero, el núcleo determinará el valor por medio de la autoverificación.

El primer parámetro especifica el IRQ asignado al dispositivo. Por defecto, el núcleo intentará autocomprobar el canal IRQ del dispositivo. El controlador 3c503, por ejemplo, tiene una característica especial que selecciona un IRQ libre de entre el 5, 9, 3, 4 y configura la tarjeta para usar uno. El párametro *base_addr* proporciona la dirección base de E/S de la tarjeta, un valor de 0 le dirá al núcleo que pruebe las direcciones listadas arriba.

Varios controladores usan los dos parámetros siguentes de forma diferentes. Para tarjetas de memoria−compartida, como la WD80x3, éstos especificarán las direcciones de principio y final del área de memoria compartida. Otras tarjetas normalmente usan el *param1* para especificar el nivel de información para debuging que es mostrada. Con valores del 1 al 7 variará la cantidad de información mostrada, con 8 no se mostrará nada; con 0 se usará el valor por omisión. El controlador 3c503 usa el *param2* para elegir entre el transceptor (transceiver) interno (por defecto) o el transceptor externo (el valor de 1). El anterior usa el conector BNC de la tarjeta, el posterior usa el puerto AUI. Los argumentos de *param* no son necesarios en todo si no tiene nada especial que configurar.

El primer argumento no numerico es interpretado por el núcleo como el nombre del dispositivo. Debe especificar un nombre de dispositivo para cada tarjeta Ethernet.

Si tiene dos tarjetas Ethernet, puede dejar que Linux autodetecte una e indicarle los parámetros de la segunda con **lilo**, pero probablemente querrá configurar manualmente las dos. Si decide que el núcleo busque la primera y configurar manualmente la segunda, tendrá que asegurarse de que el núcleo no ha encontrado accidentalmente la segunda tarjeta primero, o cualquiera que no quiera que sea registrada. Haga esto introduciendo en **lilo** la opción reserve, con esto le dice al núcleo que evite la comprobacion de la direccion base E/S que usa la segunda tarjeta. Para hacer que Linux instale una segunda tarjeta Ethernet en `0x300` como `eth1`, tendrá que usar los siguientes argumentos en el núcleo:

```
reserve=0x300,32 ether=0,0x300,eth1
```

La opción *reserve* asegura que ningún controlador accederá a la dirección E/S de la segunda tarjeta cuando compruebe algún dispositivo. También puede usar los parámetros del núcleo para evitar la autoverificación para `eth0`:

```
reserve=0x340,32 ether=0,0x340,eth0
```

También puede desactivar la autoverificación. Debe de hacer esto, por ejemlo, para detener la búsqueda de una tarjeta Ethernet que ha quitado temporalmente. Deshabilitar la autoverificación es tan simple como especificar el *base_addr* con un 1:

```
ether=0,-1,eth0
```

Para indicar al núcleo éstos parámetros antes de arrancar, introducirá los parámetros en el indicador "boot:" del lilo. Para que **lilo** muestre el indicador **`"boot:"`**, tiene que pulsar una de las siguientes teclas: Control, Alt o Shift, mientras **lilo** esté arrancando. Si pulsa la tecla del tabulador en el indicador, le aparecerá la lista de núcleos que puede arrancar. Para arrancar un núcleo con los parámetos suministrados escriba el nombre del núcleo que desea que arranque, seguido de un espacio, acompañándolo con el parámetro que desea. Cuando pulse la tecla Enter, **lilo** cargará ese núcleo y lo iniciará con el parámetro que ha escrito.

Para que este cambio ocurra automáticamente en cada arranque, introduzca los parámetros en el fichero `/etc/lilo.conf` usando la palabra append=. Quedando algo parecido a esto por ejemplo:

```
boot=/dev/hda
root=/dev/hda2
install=/boot/boot.b
map=/boot/map
vga=normal
delay=20
append="ether=10,300,eth0"

image=/boot/vmlinuz-2.2.14
label=2.2.14
```

```
    read-only
```

Después de que haya editado `lilo.conf`, debe ejecutar **lilo** para activar el cambio.

3.4. El controlador PLIP

Parallel Line IP (PLIP) es una forma barata de trabajar en red cuando sólo quiere conectar dos máquinas. Esta usa un puerto paralelo y un cable especial y llega a alcanzar velocidades desde los 10KB/s hasta los 20KB/s

PLIP fue desarrollada originalmente por Crynwr, Inc. Este diseño fue muy ingenioso en su tiempo (o, si lo prefiere, un hack), porque el puerto paralelo original de los IBM PCs fue diseñado para perder su tiempo sólo con puertos de impresoras unidireccionales; las ocho líneas de datos pueden ser usadas sólo para enviar datos desde el PC al periférico, pero no al contrario. [40] El diseño de Cyrnwr del PLIP trabajaba evitando esta limitación mediante el uso de las 5 líneas de estado del puerto para la entrada, lo cual limitaba a transferir todos los datos sólo como nibbles (medios bytes), pero permitía la transferencia bidireccional. Este modo de operación fue llamado PLIP mode 0. Actualmente, el puerto paralelo de los PCs permite la transmision de 8−bits de datos bidireccionalmente, y PLIP ha sido ampliado para acomodarse a esta situacion con la adición del PLIP mode 1.

Los núcleos Linux version 2.0 e inferiores sólo soportan PLIP mode 0, y existe un controlador para puerto paralelo mejorado en forma de parche para el núcleo 2.0, y como parte estándar del codigo del núcleo 2.2, para realizar operaciones en PLIP mode 1. [41] A pesar de las últimas versiones del código de PLIP, el controlador continúa siendo compatible con la implementacion de Crynwr del PLIP, como por ejemplo el controlador PLIP del NCSA **telnet** .[42] Para conectar dos máquinas usando PLIP, necesitas un cable especial que se vende en las tiendas como Cable de Impresora Nulor o cable Laplink Turbo. Puede, además, hacer usted mismo uno fácilmente; Apéndice B aquí le muestran cómo.

El controlador PLIP para Linux es el resultado del trabajo de muchas personas. Actualmente esta mantenido por Niibe Yutaka.[43] Si es compilado dentro del núcleo, éste creara una interfaz de red para cada uno de los posibles puertos de impresora, `plip0` corresponderá al puerto paralelo `lp0`, `plip1`, corresponderá al `lp1`, etc. El mapeado de interfaces a puertos cambia de los núcleos 2.0 a los núcleos 2.2. En los 2.0, el mapeado estaba especificado en el fichero `drivers/net/Spacd.c` del código fuente del núcleo. El mapeado por defecto en este fichero es:

Interface	I/O Port	IRQ
`plip0`	`0x3BC`	7
`plip1`	`0x378`	7
`plip2`	`0x278`	5

Si configuró su puerto de impresora de una forma diferente, tiene que cambiar estos valores en `drivers/net/Space.c` dentro del código fuente del núcleo Linux y compilar un núcleo nuevo.

En los núcleos 2.2, el controlador PLIP usa el controlador compartido de puerto paralelo parport desarrollado por Philip Blundell.[44] El controlador nuevo asigna en serie los nombres a los dispostivos de red PLIP, como pasa con los controladores Ethernet o PPP, por lo tanto el primer dispositivo PLIP creado es `plip0`, el segundo es `plip1`, etc... Los puertos paralelos hardware son también asignados en serie. Por omisión, el controlador de puerto paralelo intentará detectarlos con una rutina de autoverificación, guardando la información física del dispositivo en el orden en que la encuentra. Ésto es mejor que decirle con parámetros al núcleo la E/S. Puede hacer ésto introduciendo la información en el módulo `parport_pc.o` cuando lo carga, o si tiene compilado el controlador dentro del núcleo, usando lilo como hemos explicado antes La IRQ de cada dispositivo puede cambiarse después escribiendo un nuevo valor en el fichero que le corresponda:

/proc/parport/*/irq.

Configurar los parámetros de E/S en un núcleo 2.2 cuando cargamos un módulo es sencillo. Por ejemplo, para decirle al controlador que tienes dos puertos paralelos de PC en las direcciones E/S 0x278 y 0c378 y los IRQs 5 y 7, respectivamente, tendrá que cargar el módulo con los siguientes argumentos:

```
    modprobe parport_pc io=0x278,0x378 irq=5,7
```

Los argumentos correspondientes para un controlador ya compilado en el núcleo son:

```
    parport=0x278,5 parport=0x378,7
```

Puede usar la orden append de lilo para mandar esos argumentos al núcleo cada vez que arranque.

Cuando el controlador PLIP es inicializado, cada arranque, si éste esta dentro del núcleo, o cuando el módulo plip.o es cargado, cada puerto paralelo tendráa un dispositivo de red plip asociado. El plip0 será asignado al primer dispositivo de puerto paralelo, plip1 al segundo, etc... Puede evitar manualmente la asignacióon automática enviando otro parámetro al núcleo. Por ejemplo, para asignar el parport0 al dispositivo de red plip0, y el parport1 al dispositivo de red plip1, usará los argumentos del núcleo:

```
    plip=parport1 plip=parport0
```

Este mapeado no significa que aparte no pueda usar esos puertos paralelos para imprimir u otros propóositos. El controlador PLIP sólo usa el dispositivo de puerto paralelo cuando la correspondiente interfaz esté activada.

3.5. Los controladores PPP y SLIP

El Point−to−Point Protocol (PPP) y Serial Line IP (SLIP) son protocolos muy usados para transportar paquetes IP sobre un enlace serie. Un gran número de instituciones ofrecen acceso a Internet por conexiones PPP y SLIP proporcionando conectividad IP a personas privadas.

No son necesarias modificaciones de hardware para ejecutar PPP o SLIP, puede usar cualquier puerto serie. Desde que la configuración del puerto serie no es especifica para el trabajo en red con TCP/IP, hemos dedicado un capítulo a parte para esto. Acuda a Capítulo 4 para más información. Tratamos el PPP en detalle en Capítulo 8, y SLIP en Capítulo 7.

3.6. Otros tipos de redes

Muchos otros tipos de redes se configuran de forma similar a una Ethernet. Los argumentos que pueden llevar los módulos serán diferentes y algunos controladores puede que no soporten más de una tarjeta, pero en cuanto a lo demás son iguales. Normalmente hay documentacion disponible sobre estas tarjetas en el directorio /usr/src/linux/Documentation/networking/ del código fuente del núcleo Linux.

Capítulo 4. Configuración del Hardware Serie

Internet está creciendo en una proporción increíble. Gran parte de este crecimiento se atribuye a usuarios que no pueden permitirse conexiones permanentes a la red, y que usan protocolos como SLIP, PPP, o UUCP para conectar por teléfono con un proveedor de red y recoger su ración diaria de correo electrónico y noticias (news).

Este capítulo va dedicado a aquellos que emplean módems para mantener su vínculo con el resto del mundo. No se cubrirán los aspectos de configuración del módem (el manual que lo acompaña le proporcionará más información que nosotros), pero sí nos centraremos en la mayor parte de los aspectos específicos de GNU/Linux relativos al uso de dispositivos que emplean puertos serie. Incluiremos temas sobre aplicaciones para comunicación serie, creación de ficheros de dispositivos serie, hardware serie y configuración de dispositivos con las órdenes **setserial** y **stty**. El Serial HOWTO de David Lawyer cubre otros muchos temas.[45]

4.1. Software de Comunicaciones para Enlaces con Módem

GNU/Linux tiene disponible cierta cantidad de aplicaciones de comunicaciones. Muchas de éstas son *programas de terminal*, que permiten al usuario conectar por teléfono a otro ordenador como si estuviera sentado tras un terminal simple. El programa más tradicional para entornos Unix y similares es **kermit**. No obstante, es bastante antiguo y podría ser considerado difícil de usar. Hay otros programas más cómodos y con más características, como listines de teléfonos a los que conectar, lenguajes de script para hacer automática la llamada y acceso (login) al sistema remoto, y un rango de protocolos de intercambio. Uno de esos programas es **minicom**, creado con base en varios de los programas de terminal para DOS. También tienen representación los usuarios de X11. **seyon** es un completo programa de comunicaciones basado en X11.

Los programas de terminal no son los únicos disponibles. Otros programas le permiten conectar con un ordenador y transferir correo electrónico y noticias (news) de forma compacta, para leer y contestar en otro momento cualquiera. Esto puede suponer un buen ahorro de tiempo, y es especialmente útil si tiene usted la mala fortuna de vivir en una zona donde las llamadas locales se pagan por duración. Todo el tiempo dedicado a leer y contestar se puede estar desconectado, y, cuando esté listo, puede volver a conectar y enviar todas las respuestas en un único bloque. La contrapartida es un uso un poco mayor de disco duro, ya que los mensajes son almacenados en él antes de ser leídos; sin embargo, con los precios actuales de discos duros, esta opción puede ser interesante.

UUCP ejemplifica este estilo de programas de comunicaciones. Es un conjunto de aplicaciones que copian ficheros de una computadora a otra y ejecutan programas en computadoras remotas. Se usa frecuentemente para trasladar correo o noticias dentro de redes privadas. El paquete de UUCP de Ian Taylor, que también funciona en GNU/Linux, se describe con detalle en Capítulo 16. Otro software no interactivo de comunicaciones se usa en redes como Fidonet. Existen versiones de aplicaciones de Fidonet como **ifmail**, aunque no esperamos que haya todavía mucha gente utilizándolas.

PPP y SLIP estan a mitad de camino, puesto que permiten tanto uso interactivo como no interactivo. Mucha gente usa PPP o SLIP para conectar por teléfono a la red de su Universidad u otro Proveedor de Servicios de Internet para ejecutar FTP y leer páginas de la web. PPP y SLIP son, sin embargo, empleadas también con frecuencia para conexiones permanentes o semi−permanentes de interconexión entre LAN. De todos modos, este uso es interesante sólo con RDSI u otro tipo de conexión de alta velocidad.

4.2. Introducción a los Dispositivos Serie

El núcleo de Unix proporciona dispositivos para acceder a hardware serie, típicamente conocidos como dispositivos *tty*. Se trata de una abreviatura de dispositivos *Teletype*, que fue uno de los principales fabricantes de dispositivos de terminal en los primeros días de Unix. Esta terminología se usa actualmente para cualquier terminal basado en caracteres. En este capítulo emplearemos este término para referirnos exclusivamente a los ficheros de dispositivo de GNU/Linux, no para el teminal físico.

GNU/Linux proporciona tres tipos de dispositivos tty: dispositivos serie, terminales virtuales (accesibles por orden con las pulsaciones Alt−F1 a Alt−F*nn* en la consola local), y pseudo−terminales (similares a una tubería de dos direcciones, empleados por aplicaciones como X11). Los primeros recibieron el nombre de dispositivos tty porque originalmente los terminales basados en caracteres se conectaban a la máquina Unix mediante un cable serie o con un módem y una línea de teléfono. Los otros dos recibieron la misma denominación debido a que tratan de comportarse de manera similar, desde el punto de vista del programador.

SLIP y PPP se implementan en la mayoría de casos en el núcleo. Realmente, el núcleo no trata a un dispositivo *tty* como dispositivo de red al cual se pueda manipular, como los dispositivos Ethernet, empleando órdenes como **ifconfig**. Sin embargo, sí que trata a los dispositivos tty como lugares a los que se pueden asociar dispositivos de red. Para hacer esto, el núcleo cambia lo que se llama disciplina de línea del dispositivo tty. Tanto SLIP como PPP son disciplinas de línea que pueden ser activadas en dispositivos tty. La idea, en general, es que el manejador serie trata los datos que recibe de forma distinta según la disciplina de línea para la que está configurado. En la disciplina de línea por defecto, el manejador se limita a transmitir cada carácter que recibe por orden. Por contra, al seleccionar SLIP o PPP como disciplina de línea, el manejador lee bloques de datos, los encapsula con una cabecera especial que permite al otro extremo de la conexión identificarlos en una secuencia, y transmite todo el nuevo bloque. No es realmente importante comprender esto por ahora; trataremos SLIP y PPP en capítulos posteriores. De todos modos, esto sucede automáticamente para usted.

4.3. Acceso a Dispositivos Serie

Como todo dispositivo en un sistema Unix, los puertos serie son accesibles a través de ficheros especiales de dispositivo, localizados en el directorio `/dev`. Hay dos tipos de ficheros de dispositivo relacionados con manejadores serie, y hay un fichero de dispositivo de cada tipo para cada puerto. El dispositivo tendrá un comportamiento levemente distinto, según cuál de sus ficheros de dispositivo empleemos. Cubriremos estas diferencias porque ayudará a entender algunos aspectos de configuración y algunos consejos que puede encontrar respecto a dispositivos serie, pero en la práctica, sólo necesita utilizar uno de ellos. Quizá en un futuro desaparezca alguno de estos tipos.

La más importante de las dos clases de dispositivos serie tiene un número mayor de 4, y sus ficheros especiales de dispositivo se llaman `ttyS0`, `ttyS1`, etc. La otra variedad tiene un número mayor de 5, y fue diseñada para emplearse en llamadas salientes a través de un puerto; sus ficheros especiales de dispositivo son `cua0`, `cua1`, etc. En el mundo Unix, las cuentas comienzan generalmente en cero, mientras que los profanos tienden a comenzar por uno. Esto genera una pequeña confusión ya que `COM1:` se representa por `/dev/ttyS0`, `COM2:` por `/dev/ttyS1`, etc. Un usuario cualquiera familiarizado con hardware del estilo del IBM PC sabe que `COM3:` y mayores nunca llegaron a ser estándar, de todos modos.

Los dispositivos *cua*, o llamada salientes, fueron creados para solucionar el problema de evitar conflictos en dispositivos serie para módems que tienen que aceptar tanto conexiones entrantes como conexiones salientes. Desafortunadamente, han creado sus propios problemas y probablemente dejarán de ser utilizados. Echemos un vistazo al asunto.

GNU/Linux, igual que Unix, permite que un dispositivo, u otro fichero cualquiera, sea abierto por más de un proceso de forma simultánea. Por desgracia, esto es raramente útil para dispositivos tty, ya que ambos procesos interferirán entre sí. Pero, por suerte, se diseñó un mecanismo que permite a un proceso comprobar si un dispositivo tty está en uso por otro proceso antes de tratar de abrirlo. Este mecanismo usa lo que denominamos *ficheros de bloqueo*. La idea es que cuando un proceso trata de abrir un dispositivo tty, ha de comprobar la existencia de un fichero en un lugar especial, llamado de forma parecida al dispositivo tty. Si este fichero no existe, el proceso lo crea y abre el dispositivo tty. Si el fichero, por contra, ya existe, el proceso asume que hay otro proceso que ya ha abierto el dispositivo tty y toma la decisión adecuada. Un último truco para que este sistema de manejo funcionara adecuadamente es escribir el identificador (pid) del proceso que ha creado el fichero de bloqueo en el propio fichero de bloqueo; seguiremos con este punto un poco más abajo.

El mecanismo de ficheros de bloqueo funciona perfectamente en los casos en que tenemos una localización bien definida para estos ficheros de bloqueo, y todos los programas saben dónde buscarlos. Sin embargo, este no ha sido siempre el caso de GNU/Linux. No fue hasta que se definió el Estándar de Sistema de Ficheros de Linux (Linux Filesystem Standard) cuando comenzaron a trabajar correctamente los ficheros de bloqueo `tty`. Llegó a haber cuatro, e incluso más lugares elegidos por los programadores para almacenar los ficheros de bloqueo: `/usr/spool/locks/`, `/var/spool/locks/`, `/var/lock/` y `/usr/lock/`. La confusión trajo el caos. Los programas abrían ficheros de bloqueo en lugares distintos para controlar un mismo dispositivo tty; la situación era similar a no usar ficheros de bloqueo en absoluto.

Los dispositivos `cua` fueron creados para solventar este problema. En lugar de confiar a los ficheros de bloqueo la prevención de conflictos entre procesos que pretendían usar dispositivos serie, se decidió que el núcleo podría suministrar un método sencillo de arbitrar quién debía obtener acceso. Si el dispositivo `ttyS` estaba abierto, un intento de abrir el `cua` resultaría en un error que podría ser interpretado por el programa como que el dispositivo ya estaba en uso. Si el `cua` estaba previamente abierto y se trataba de abrir el `ttyS`, la petición crearía un bloqueo; es decir, se pondría en espera hasta que el dispositivo `cua` fuera cerrado por el otro proceso. Esta solución era adecuada para casos como un módem único configurado para recibir accesos entrantes y que, en ocasiones, se quisiera emplear para accesos salientes. Pero no era suficiente para ámbitos en los que varios programas tratan de realizar llamadas salientes desde el mismo dispositivo. La única forma de remediar este problema era ¡usar ficheros de bloqueo! De vuelta al problema inicial.

Basta decir que el Estándar de Sistema de Ficheros de Linux llegó al rescate y ahora es obligatorio almacenar los ficheros de bloqueo en el directorio `/var/lock`, y que, por acuerdo, el nombre del fichero de bloqueo correspondiente al dispositivo `ttyS1`, por ejemplo, es `LCK..ttyS1`. Los ficheros de bloqueo `cua` también deberían ir en este directorio, pero el uso de dispositivos `cua` queda desaconsejado

Los dispositivos `cua` seguirán existiendo por un tiempo, para conservar la compatibilidad con software antiguo, pero con el tiempo serán retirados. Si usted se pregunta cuáles debe usar, quédese con los dispositivos `ttyS`, y asegúrese de que su sistema cumpla con el estándar Linux FSSTND (Estándar de Sistema de Ficheros de Linux), o, como mínimo, que todos los programas que accedan a dispositivos serie estén de acuerdo en la localización de los ficheros de bloqueo. Gran parte del software que trata con dispositivos tty serie proporciona opciones de compilación para especificar la localización de ficheros de bloqueo. Es probable que aparecerá como una variable llamada `LOCKDIR` en el `Makefile` o en algún fichero de configuración de cabecera. Si usted mismo compila el software, la mejor opción es modificar esto de acuerdo al lugar definido en el FSSTND. Si usa usted binarios precompilados y no está seguro de dónde escribirá el programa sus ficheros de bloqueo, quizá esta orden pueda proporcionarle alguna pista:

```
strings binaryfile | grep lock
```

Si el lugar encontrado no es compatible con el resto de su sistema, puede tratar de crear un enlace simbólico desde el directorio de bloqueo que pretende usar el binario hacia `/var/lock`. No es una solución muy elegante, pero funcionará.

4.3.1. Los Ficheros Especiales De Dispositivos Serie

Los números menores son idénticos para ambos tipos de dispositivos serie. Si tiene usted su módem conectado en un puerto desde COM1: a COM4:, su número menor será el número de puerto COM más 63. Si emplea usted hardware serie especial, como controladores de múltiples puertos serie de gran rendimiento, probablemente necesite crear ficheros especiales de dispositivo para él; probablemente no emplee el manejador estándar de dispositivo. El Serial−HOWTO debe poder ayudarle a encontrar los detalles específicos.

Supongamos que su módem está en COM2:. Su número menor será 65, y su número mayor será 4 para usos normales. Debería existir un dispositivo llamado `ttyS1` que tiene estos números. Liste los ttys serie del directorio `/dev`. La quinta y sexta columna muestran respectivamente el número mayor y el número menor:

```
$ ls −l /dev/ttyS*
0 crw−rw−−−−   1 uucp     dialout    4,  64 Oct 13  1997 /dev/ttyS0
0 crw−rw−−−−   1 uucp     dialout    4,  65 Jan 26 21:55 /dev/ttyS1
0 crw−rw−−−−   1 uucp     dialout    4,  66 Oct 13  1997 /dev/ttyS2
0 crw−rw−−−−   1 uucp     dialout    4,  67 Oct 13  1997 /dev/ttyS3
```

Si no hay ningún dispositivo con número mayor 4 y número menor 65, necesitará crear uno. Pase a modo superusuario y escriba:

```
# mknod −m 666 /dev/ttyS1 c 4 65
# chown uucp.dialout /dev/ttyS1
```

Según la distribución de GNU/Linux, se emplean estrategias sutilmente distintas para determinar quién debe ser propietario de los dispositivos serie. A veces son propiedad de *root*, y en otros casos pertenecen a otro usuario, como **uucp** en nuestro ejemplo. Las distribuciones más modernas tienen un grupo especial para dispositivos que realizan llamadas salientes, y todo usuario que pueda emplearlos estará añadido a este grupo.

Hay quien sugiere crear `/dev/modem` como un enlace simbólico al dispositivo de módem para que los usuarios ocasionales no tengan que recordar el menos intuitivo `ttyS1`. Pero no se puede utilizar `modem` en un programa y el fichero real de dispositivo en otro. Sus ficheros de bloqueo tendrían nombres diferentes, y esto haría fallar al mecanismo de bloqueo.

4.4. Hardware Serie

RS−232 es actualmente el estándar más común para comunicaciones serie en el mundo de los PC. Emplea una serie de circuitos para transmitir bits de uno en uno, así como para sincronización. Otras líneas adicionales pueden ser empleadas para señalar la presencia de un portador (empleadas por módems) y para negociación (handshaking). GNU/Linux acepta una amplia variedad de tarjetas serie que usan el estándar RS−232.

La negociación por hardware es opcional, pero muy útil. Permite a cualquiera de las dos estaciones avisar si está preparada para recibir más datos, o si la otra estación debería esperar mientras se procesan los datos recibidos. Las líneas usadas para esto se llaman Clear to Send (CTS) y Ready to Send (RTS), respectivamente, lo que explica el nombre coloquial para negociación por hardware: RTS/CTS. El otro tipo de negociación con el que quizá esté usted familiarizado es la negociación XON/XOFF XON/XOFF utiliza dos caracteres concretos, convencionalmente Ctrl−S y Ctrl−Q, para indicar al extremo remoto que debería parar o comenzar la transmisión de datos, respectivamente. A la par que este método es simple de implementar y válido para uso con terminales sin procesamiento, también es cierto que crea una gran confusión cuando los datos son binarios, ya que probablemente se quisieran transmitir esos caracteres como parte del flujo de datos que enviamos, en lugar de que fueran interpretados como caracteres de control del flujo. Por otra parte, la rapidez con que toman efecto es menor que con negociación por hardware. La negociación hardware es limpia, rápida y se recomienda sobre XON/XOFF si existe la posibilidad de elegir.

En el IBM PC original, el interfaz RS−232 era manejado por un chip UART llamado 8250. Los PCs de la época del 486 emplearon una versión más moderna del UART, llamada 16450. Era ligeramente superior al 8250. Casi todas las máquinas basadas en procesadores Pentium contienen otra versión aún más moderna del UART, llamada 16550. Ciertas marcas (particularmente módems internos equipados con el chipset Rockwell) utilizan chips completamente distintos que emulan el comportamiento del 16550 y pueden ser tratados de forma similar. GNU/Linux acepta todos ellos con su manejador estándar de puerto serie.[46]

El 16550 fue una mejora significativa sobre el 8250 y el 16450, ya que proporciona un búfer FIFO de 16 bytes. En realidad, el 16550 es una familia de dispositivos UART, que comprende el 16550, el 16550A y el 16550AFN (posteriormente llamado PC16550DN). Las diferencias radican en si el FIFO funciona realmente o no; el 16550AFN es el único en el que es seguro que funciona. También existió un NS16550, pero su FIFO nunca funcionó.

Los UARTs 8250 y 16450 tenían un buffer simple de 1 byte. Esto significa que un 16450 genera una interrupción por cada carácter transmitido o recibido. Cada interrupción se procesa en un corto período de tiempo, y esta pequeña demora limita a los 16450 a una velocidad de bits máxima y fiable de 9.600 bps en una máquina de arquitectura ISA típica.

En su configuración por defecto, el núcleo comprueba los cuatro puertos serie estándar, desde COM1: a COM4:. El núcleo también puede detectar qué tipo de UART se emplea en cada uno de los puertos serie estándar, y hará uso del búfer FIFO mejorado del 16550, si está disponible.

4.5. Uso de las Utilidades de Configuración

Dediquemos algo de tiempo a examinar las dos utilidades más comunes: **setserial** y **stty**.

4.5.1. La orden setserial

El núcleo hará su mayor esfuerzo en determinar cómo está configurado su hardware serie, pero las diversas variantes en la configuración de dispositivos serie hace difícil obtener una determinación fiable al 100%. Un buen ejemplo de este problema son los módems internos que discutimos anteriormente. La UART que usan tiene un búfer FIFO de 16 bytes, pero parece una UART 16450 para el núcleo: a no ser que específicamente informemos al manejador que el puerto es realmente un dispositivo 16550, el núcleo no hará uso del búfer extendido. Otro ejemplo es el de las tarjetas de cuatro puertos sin procesamiento que permiten compartir una misma IRQ entre un número de dispositivos serie. Probablemente tengamos que especificarle al núcleo qué IRQ se supone que usará el puerto, y que las IRQ pueden estar compartidas.

setserial fue creado para configurar el manejador serie en tiempo de ejecución. La orden **setserial** se ejecuta por lo común al arrancar, desde un script llamado `0setserial` en unas distribuciones, y `rc.serial` en otras. Este script está al cargo de iniciar adecuadamente el manejador de puerto serie para dar cabida a cualquier hardware no estándar o inusual que haya en la máquina.

La sintaxis general para la orden **setserial** es:

```
    setserial dispositivo [parámetos]
```

donde el dispositivo es uno de los dispositivos serie, como *ttyS0*.

La orden **setserial** tiene un gran número de parámetros. Los más comunes vienen descritos en Tabla 4−1. Para más información sobre el resto de los parámetros, consulte la página de manual de **setserial**.

Tabla 4−1. Parámetros de Línea de Órdenes de setserial

Parámetro	Descripción
`port` *número_puerto*	Especifica la dirección de E/S del dispositivo serie. Los números de puerto deben ser especificados en notación hexadecimal, p.ej. `0x2f8`.
`irq` *núm*	Especifica la línea de petición de interrupción que emplea el dispositivo serie.
`uart` *tipo_uart*	Especifica el tipo de UART del dispositivo serie. Los valores más comunes son `16450`, `16550`, etc. Ajustar este valor a `none` desactivará el dispositivo serie.
`fourport`	El uso de este parámetro informará al manejador de puertos serie del núcleo de que este puerto forma parte de una tarjeta AST Fourport.
`spd_hi`	Programar la UART para emplear una velocidad de 57,6 kbps cuando un proceso solicita 38,4 kbps.
`spd_vhi`	Programar la UART para emplear una velocidad de 115 kbps cuando un proceso solicita 38,4 kbps.
`spd_normal`	Programar la UART para emplear la velocidad predefinida de 38,4 kbps cuando se le solicite. Este parámetro sirve para deshacer el efecto de un `spd_hi` o `spd_vhi` aplicado en el dispositivo serie especificado.
`auto_irq`	Este parámetro provocará que el núcleo trate de determinar automáticamente la IRQ del dispositivo especificado. Este intento puede no ser fiable completamente, así que quizá sea mejor entenderlo como una solicitud al núcleo para que adivine la IRQ. Si conoce usted la IRQ del dispositivo, debe especificar que se use el parámetro `irq` en su lugar.
`autoconfig`	Este parámetro debe ser especificado simultáneamente con el parámetro `port`. Cuando se suministra este parámetro, **setserial** ordena al núcleo que intente determinar automáticamente el tipo de UART localizada en la dirección de puerto proporcionada. Si el parámetro `auto_irq` también es suministrado, el núcleo tratará también de determinar la IRQ automáticamente.
`skip_test`	Con este parámetro se solicita que el núcleo no se ocupe de determinar el tipo de UART en la auto−configuración. Esto es necesario cuando la UART es detectada de forma incorrecta por el núcleo.

Un fichero `rc` típico y sencillo para configurar sus puertos serie al arrancar puede parecerse a lo mostrado en Ejemplo 4−1. Una mayoría de distribuciones de GNU/Linux incluirán algo más sofisticado que este ejemplo.

Ejemplo 4−1. Ejemplo de órdenes setserial en rc.serial

```
# /etc/rc.serial − script de configuración de líneas serie.
#
# Configurar dispositivos serie.
/sbin/setserial /dev/ttyS0 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS1 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS2 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS3 auto_irq skip_test autoconfig
#
# Mostrar la configuración de dispositivos serie.
/sbin/setserial −bg /dev/ttyS*
```

El argumento `−bg /dev/ttyS*` en la última orden mostrará un pulcro sumario de la configuración hardware de todos los dispositivos serie activos. Esta salida se parecerá a la mostrada en Ejemplo 4−2.

Ejemplo 4−2. Salida de la orden setserial −bg /dev/ttyS*

```
/dev/ttyS0 at 0x03f8 (irq = 4) is a 16550A
/dev/ttyS1 at 0x02f8 (irq = 3) is a 16550A
```

4.5.2. La Orden stty

El nombre **stty** probablemente signifique set tty, pero la orden **stty** también puede ser empleada para mostrar la configuración de un terminal. Quizás aún más que **setserial**, la orden **stty** proporciona un desconcertante número de características configurables. Cubriremos las más importantes en breve. Puede usted encontrar descrito el resto en la página de manual de **stty**.

La orden **stty** se utiliza principalmente para configurar parámetros del terminal, tales como qué caracteres serán mostrados, o qué tecla deberá generar una señal de parada. Explicamos anteriormente que los dispositivos serie son dispositivos tty, y por tanto la orden **stty** es igualmente aplicable a ellos.

Uno de los usos más importantes de **stty** para dispositivos serie es habilitar la negociación por hardware en ellos. Anteriormente describimos someramente la negociación por hardware. La configuración por defecto en dispositivos serie es que esta negociación hardware esté deshabilitada. Esta disposición permite que los cables serie de tres hilos funcionen; estos cables no aceptan las señales necesarias para la negociación por hardware, y si estuviera activada por defecto, serían incapaces de transmitir ningún carácter para desactivarla.

Sorprendentemente, algunos programas de comunicación serie no habilitan la negociación por hardware, así que si su módem la permite, deberá configurar el módem para emplearla (diríjase al manual de su módem para averiguar qué orden usar), y también deberá configurar el dispositivo serie para utilizar negociación por hardware. La orden **stty** tiene una bandera `crtscts` que habilita la negociación por hardware en un dispositivo; tendrá que utilizar esto. El lugar más apropiado para ejecutar esta orden probablemente sea el fichero `rc.serial` (o equivalente) al arrancar, utilizando sentencias como las mostradas en Ejemplo 4−3.

Ejemplo 4−3. Órdenes stty de Ejemplo en rc.serial

```
#
stty crtscts < /dev/ttyS0
stty crtscts < /dev/ttyS1
stty crtscts < /dev/ttyS2
stty crtscts < /dev/ttyS3
#
```

La orden **stty** trabaja en el terminal en uso de forma predefinida, pero mediante la redirección de entrada (<) facilitada por el intérprete de órdenes, podemos hacer que **stty** maneje cualquier dispositivo tty. Es un error común olvidar si se ha de emplear < o >; las versiones más modernas de **stty** tienen una sintaxis más clara de especificarlo. Con esta nueva sintaxis, nuestro ejemplo quedaría como en Ejemplo 4−4.

Ejemplo 4−4. Órdenes stty de Ejemplo en rc.serial Empleando Sintaxis Moderna

```
#
stty crtscts −F /dev/ttyS0
stty crtscts −F /dev/ttyS1
stty crtscts −F /dev/ttyS2
stty crtscts −F /dev/ttyS3
#
```

Antes mencionamos que la orden **stty** puede ser usada para mostrar los parámetros de configuración de un dispositivo tty. Para mostrar todos los parámetros activos de un dispositivo tty se hace:

```
$ stty -a -F /dev/ttyS1
```

La salida de esta orden, mostrada en Ejemplo 4−5, le muestra el estado de todas las banderas para ese dispositivo; una bandera precedida por un signo menos, como en `crtscts`, significa que la bandera ha sido desactivada.

Ejemplo 4−5. Salida de una Orden stty −a

```
speed 19200 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
         eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
         werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr -icrnl -ixon
        -ixoff -iuclc -ixany -imaxbel
-opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0
         bs0 vt0 ff0
-isig -icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop
         -echoprt echoctl echoke
```

Una descripción de las banderas más importantes se da en Tabla 4−2. Cada una de ellas se habilita suministrándola a **stty**, y se deshabilita de la misma forma, pero precedida por un carácter delante. Así, para deshabilitar la negociación hardware, se haría:

```
$ stty -crtscts -F /dev/ttyS0
```

Tabla 4−2. Banderas de stty Más Relevantes Para Configurar Dispositivos Serie

Bandera	**Descripción**
`N`	Ajustar la velocidad de línea a `N` bits por segundo.
`crtsdts`	Habilitar/Deshabilitar negociación por hardware.
`ixon`	Habilitar/Deshabilitar control de flujo mediante XON/XOFF.
`clocal`	Habilitar/Deshabilitar señales de control del módem como DTR/DTS y DCD. Esto es necesario si se usa cable serie de tres hilos ya que no proporciona estas señales.
`cs5 cs6 cs7 cs8`	Ajustar el número de bits de datos a 5, 6, 7 u 8, respectivamente.
`parodd`	Habilitar paridad impar. Desactivar esta bandera activa la paridad par.
`parenb`	Habilitar comprobación de paridad. Si esta bandera se niega, no se utiliza paridad.
`cstopb`	Ajustar los bits de parada a dos por carácter. Al negar esta bandera, se usará sólo un bit de parada por carácter.
`echo`	Habilitar/Deshabilitar el eco de caracteres recibidos al que los envía.

El siguiente ejemplo combina algunas de estas banderas y configura el dispositivo `ttyS0` a 19.200 bps, 8 bits de datos, sin paridad, con negociación por hardware y eco deshabilitado:

```
$ stty 19200 cs8 -parenb crtscts -echo -F /dev/ttyS0
```

4.6. Dispositivos Serie y el Indicador login: (ingreso)

Llegó a ser muy común que una instalación basada en Unix incluyera una máquina servidor y muchos terminales tontos (sin procesamiento) de caracteres o módems para acceso telefónico. Hoy en día este tipo de configuraciones son menos comunes, lo que supone una buena noticia para mucha gente interesada en trabajar así, ya que estos terminales sin procesamiento son muy baratos actualmente. Las configuraciones con módems de acceso telefónico no han dejado de ser comunes, pero en estos tiempos probablemente se utilicen como soporte para ingresos mediante SLIP o PPP (temas tratados en Capítulo 7 y Capítulo 8) más que para un simple ingreso. En cualquier caso, cada uno de estos métodos puede hacer uso de un programa sencillo llamado programa **getty**.

El término **getty** es probablemente una abreviatura de get tty (conseguir tty). Un programa **getty** abre un dispositivo serie, lo configura apropiadamente, configura opcionalmente un módem, y espera a que se realice una conexión. Una conexión activa en un dispositivo serie se indica normalmente mediante la patilla Data Carrier Detect (DCD) en el dispositivo serie que ha sido activado. Cuando se produce esta detección, el programa **getty** llama a un programa que muestra el punto indicativo `login:`, y es el que maneja realmente el ingreso al sistema. Cada uno de los terminales virtuales (por ejemplo, `/dev/tty1`) en GNU/Linux tiene un **getty** ejecutándose para él.

Hay varias implementaciones distintas de **getty**, cada una diseñada para adaptarse a ciertas configuraciones mejor que a otras. El **getty** que describiremos aquí se llama **mgetty**. Es bastante conocido porque proporciona todo tipo de características que lo hacen especialmente indicado para uso con módems, aceptando incluso programas automáticos para fax y módems de voz. Nos concentraremos en configurar **mgetty** para responder a las llamadas convencionales de datos, y dejaremos el resto para que explore usted en función de sus necesidades.

4.6.1. Configuración del Demonio mgetty

El demonio **mgetty** está disponible en forma de código fuente en ftp://alpha.greenie.net/pub/mgetty/source/, y prácticamente todas las distribuciones de GNU/Linux lo incluyen como paquete. El demonio **mgetty** se diferencia de la mayoría de las implementaciones de programas **getty** en que ha sido diseñado específicamente para módems compatibles con Hayes. Admite aún conexiones directas de terminales, pero se adapta de forma especialmente buena a aplicaciones de llamadas telefónicas. En lugar de emplear la señal DCD para detectar una llamada entrante, espera a la escucha del mensaje `RING` generado por la mayoría de módems modernos cuando detectan una llamada entrante y no están configurados para responder automáticamente.

El ejecutable principal es `/usr/sbin/mgetty`, y su fichero de configuración se llama `/etc/mgetty/mgetty.config`. Hay otros programas y ficheros de configuración que cubrirán otros rasgos concretos de **mgetty**.

La configuración, en gran parte de los sistemas, es cuestión únicamente de editar el fichero `/etc/mgetty/mgetty.config` y añadir las entradas apropiadas a `/etc/inittab` para que **mgetty** se ejecute automáticamente.

Ejemplo 4−6 muestra un fichero muy simple de configuración de **mgetty**. Este ejemplo configura dos dispositivos serie. El primero, `/dev/ttyS0`, se refiere a un módem compatible con Hayes a 38.400 bps. El segundo, `/dev/ttyS1`, se corresponde con un terminal VT100 conectado directamente, a 19.200 bps.

Ejemplo 4−6. Fichero /etc/mgetty/mgetty.config de ejemplo

```
#
# fichero de configuración de mgetty
#
# este es un fichero de ejemplo de configuración, vea mgetty.info para obtener detalles
#
# las líneas de comentario  comienzan con "#", las líneas vacías son deshechadas
#
# ----- sección global -----
#
# En esta sección van los valores globales por defecto, la configuración por puertos va debajo
#
```

```
# acceder al módem (o módems) a 38400 bps
speed 38400
#
# ajustar el nivel global de depuración a "4" (valor por defecto de policy.h)
debug 4
#

# ----- sección específica de puerto -----
#
# Aquí se pondran las cosas que sean válidas sólo para una línea, no las demás
#
#
# Módem Hayes conectado a ttyS0: no usar como fax, menos registro de actividad
#
port ttyS0
  debug 3
  data-only y
#
# conexión directa de un terminal VT100 que no gusta de bajadas en la señal DTR
#
port ttyS1
  direct y
  speed 19200
  toggle-dtr n
#
```

El fichero de configuración acepta opciones globales y específicas de cada puerto. En nuesto ejemplo, empleamos una opción global para fijar la velocidad a 38.400 bps. Este valor es heredado por el puerto `ttyS0`. Los puertos a los que aplicamos **mgetty** emplean este ajuste de velocidad a no ser que sea reemplazado por un ajuste de velocidad explícito para el puerto, tal y como hemos hecho en la configuración de `ttyS1`.

La palabra clave `debug` controla la cantidad de texto del registro de actividad de **mgetty**. La palabra clave `data-only` en la configuración de `ttyS0` hace que **mgetty** desprecie las características de fax del módem, para que funcione únicamente como módem de datos. La palabra `direct` en la configuración de `ttyS1` avisa a **mgetty** para que no lleve a cabo ninguna inicialización para módem en ese puerto. Finalmente, con `toggle-dtr` se consigue que **mgetty** no trate de colgar la línea bajando la patilla DTR (Data Terminal Ready) en el interfaz serie; algunos terminales no reaccionan bien ante esto.

También puede usted elegir dejar vacío el fichero `mgetty.config` y emplear argumentos en la línea de órdenes para especificar la mayoría de esos mismos parámetros. La documentación que acompaña a la aplicación incluye una completa descripción de los parámetros del fichero de configuración de **mgetty** y los argumentos de línea de órdenes. Observe el siguiente ejemplo.

Hemos de añadir dos entradas al fichero `/etc/inittab` para activar esta configuración. El fichero `inittab` es el fichero de configuración de la orden **init** de Unix System V. Esta orden es la encargada de la iniciación del sistema; proporciona un medio para ejecutar programas automáticamente al iniciar la máquina y volver a ejecutarlos cuando terminan. Es muy apropiada para los objetivos de ejecutar un programa **getty**.

```
T0:23:respawn:/sbin/mgetty ttyS0
T1:23:respawn:/sbin/mgetty ttyS1
```

Cada línea del fichero `/etc/inittab` contiene cuatro campos, separados por dos puntos. El primero es un identificador que etiqueta de forma única cada entrada del fichero; tradicionalmente se utilizan dos caracteres, pero las versiones más modernas permiten cuatro. El segundo campo es la lista de niveles de ejecución en los que deberá estar activa. Un nivel de ejecución es un mecanismo para proporcionar distintas configuraciones del equipo y se implementa mediante el uso de árboles de scripts de inicio, almacenados en directorios llamados `/etc/rc1.d`, `/etc/rc2.d`, etc. Esta característica es implementada típicamente de forma muy simple, por lo que usted debe modelar sus entradas en el fichero basándose en otras del mismo, o bien consultar la documentacion de su sistema para obtener más información. El tercer campo describe cuándo hay que llevar a cabo la acción. Para los propósitos de ejecutar un programa **getty**, este campo debe ser ajustado a `respawn`, lo que significa que la orden se re−ejecutará automáticamente cuando muera. Hay otras opciones también, pero no son útiles para nuestros propósitos aquí. El cuarto campo es la orden real que ha de ejecutarse; aquí es donde especificamos la orden **mgetty** y cualquier argumento que queramos pasarle. En nuestro ejemplo simple, ejecutamos y reiniciamos **mgetty** siempre que el sistema está operando en los niveles de ejecución dos o tres, y le suministramos como argumento el nombre del dispositivo que queremos utilizar. La orden **mgetty** asume `/dev/`, así que no hemos de proporcionarlo.

Este capítulo ha sido una introducción rápida a **mgetty** y cómo ofrecer un punto identificativo de ingreso al sistema en dispositivos serie. Podrá encontrar información más extensiva en el Serial−HOWTO.

Una vez que haya editado los ficheros de configuración ha de recargar **init** para hacer efectivos los cambios. Simplemente envíe una señal hangup al proceso **init**; siempre tiene un identificador de proceso 1, así que puede usar sin problemas:

```
# kill −HUP 1
```

Capítulo 5. Configuración del Protocolo TCP/IP

En este capítulo recorreremos todos los pasos necesarios para configurar el protocolo TCP/IP en su máquina. Empezando en la asignación de direcciones IP, iremos describiendo la configuración de las interfaces TCP/IP e introduciremos unas cuantas herramientas que resultan bastante útiles a la hora de resolver problemas surgidos durante la instalación de la red.

La mayoría de las tareas descritas en este capítulo, generalmente, sólo habrá de ejecutarlas una única vez. Una vez hecho esto, sólo tendrá que tocar alguno de los ficheros de configuración cuando añada un nuevo sistema a su red, o si decide reconfigurar el sistema completamente. Algunas de las órdenes usados para configurar el protocolo TCP/IP, sin embargo, deben ser ejecutados cada vez que se arranca el sistema. La forma usual de llevar esto a cabo es a través de los scripts `/etc/rc*`.

Generalmente, las partes específicas de la red están contenidas en un script. El nombre de este script varía dependiendo de las distintas distribuciones de GNU/Linux. En muchas distribuciones de GNU/Linux antiguas, se llama `rc.net` o `rc.inet`. A veces también aparecerán dos scripts llamados `rc.inet1` y `rc.inet2`siendo la primera la encargada de inicializar la parte del núcleo que se ocupa de las comunicaciones, mientras que la segunda es la que se encarga de arrancar los servicios básicos y las aplicaciones. En las distribuciones modernas, los ficheros `rc` se estructuran de forma más sofisticada; en ellas encontrará scripts en los directorios `/etc/init.d/` (o `/etc/rc.d/init.d/`) que crean los dispositivos de red, y otros archivos `rc` que lanzan las aplicaciones de red. Los ejemplos de este libro están basados en esta última disposición.

Este capítulo explica partes del script que configura las interfaces de su red, mientras que las aplicaciones se verán en próximos capítulos. Al finalizar este capítulo, debería usted haber establecido la secuencia de órdenes que configuran correctamente el protocolo TCP/IP en su ordenador. Sustituya las órdenes de ejemplo en los scripts de configuración por los suyos propios; asegúrese de que el script básico `rc` es ejecutado en el arranque y rearranque de su máquina. Los scripts rc que acompañen a su distribución de GNU/Linux favorita deberían ser un buen ejemplo.

5.1. Montando el Sistema de Ficheros /proc

Algunas de las herramientas de configuración de NET−2 y NET−3 utilizan el sistema de ficheros `proc` para comunicarse con el núcleo. Se trata de una interfaz que permite el acceso a la información del kernel en funcionamiento a través de un mecanismo que imita un sistema de ficheros. Una vez ha sido montado, se pueden listar los ficheros y ver su contenido como en cualquier otro sistema de ficheros. Normalmente aparecen ficheros como `loadavg`, que contiene la carga media del sistema, o `meminfo`, que contiene información sobre la memoria física y virtual.

Para esto, el código de redes añade el directorio `net`. Este directorio contiene una serie de ficheros que muestran cosas como las tablas ARP del núcleo, el estado de las conexiones TCP y las tablas de encaminamiento. La mayoría de las herramientas de administración de redes utilizan estos ficheros para acceder a la información que precisan.

El sistema de ficheros `proc` (también llamado `procfs`) se monta generalmente en `/proc` durante el arranque. El mejor método consiste en añadir la siguiente línea al fichero `/etc/fstab`:

```
# Punto de montaje de procfs
none                /proc           proc    defaults
```

Y ejecutar **mount /proc** desde alguno de los scripts `/etc/rc`.

El `procfs` viene configurado actualmente en la mayoría de los núcleos por omisión. Si no tiene el `procfs` en su núcleo, al intentar montarlo obtendrá un mensaje como este: `mount: fs type procfs not` `supported by kernel`. De ser así tiene que recompilar el núcleo asegurándose de configurarlo incluyendo el soporte para procfs.

5.2. Instalación de los ejecutables

Si está utilizando alguna de las distribuciones de GNU/Linux, incluirá las aplicaciones y utilidades de red fundamentales así como un conjunto coherente de ficheros de configuración de ejemplo. El único caso en el que tendría que conseguir e instalar las nuevas utilidades es en el caso de instalar una nueva versión del núcleo. De forma ocasional, esto supone cambios en la capa de comunicaciones del núcleo. Eso significaría tener que actualizar también las herramientas de configuración. Esto se traduce en, al menos, la necesidad de recompilar, aunque a veces es posible que sea necesario conseguir un conjunto de ejecutables actualizados. Estos ejectuables están disponibles en su sitio oficial, *ftp.inka.de/pub/comp/GNU/Linux/networking/NetTools/*, empaquetados en ficheros llamados `net-tools-XXX.tar.gz`, donde `XXX` es la versión de que se trate. En el caso de GNU/Linux 2.0 es `net-tools-1.45`.

Si quiere compilar e instalar las aplicaciones estándar de comunicaciones TCP/IP, puede obtener los ficheros fuente de la mayoría de los servidores FTP de GNU/Linux. Todas las distribuciones modernas de GNU/Linux incluyen un amplio grupo de aplicaciones de comunicaciones TCP/IP, como navegadores de la World Wide Web, programas de **telnet** y de **ftp**, y otras aplicaciones de red, como **talk**. Si encuentra algo que necesite compilar por usted mismo, es muy posible que pueda compilar sencillamente bajo GNU/Linux si sigue las instrucciones incluidas en el paquete de las fuentes.

5.3. Establecimiento del Nombre de la Máquina

La mayoría de las aplicaciones de red, si no todas, asumen que el nombre dado a la máquina local tiene un valor razonable. Este proceso tiene lugar durante el arranque cuando se ejecuta el comando **hostname**. Para llamar *nombre* haga lo siguiente:

```
# hostname nombre
```

Es una práctica común usar el nombre sin cualificarlo con el dominio de red. Así pues, supongamos que las máquinas de la Cervecera Virtual (descrita en Apéndice A) se llamarán vale.vbrew.com o vlager.vbrew.com. Estos son los nombres oficiales, los *nombres completamente cualificados de dominio* (FQDN1). Los nombres locales serían, por tanto, únicamente el primer componente del nombre, como por ejemplo vale. Sin embargo, dado que el nombre local se usa frecuentemente para buscar la dirección IP correspondiente, debe asegurarse de que la tabla que contiene esa información sea capaz de encontrar dicha IP. Esto generalmente equivale a añadir el nombre local al fichero `/etc/hosts`.

Algunas personas sugieren la utilización de la oreden **domainname** para fijar el valor del dominio para el núcleo. Así, para obtener el FQDN combinaríamos la salida de **hostname** y **domainname**. Sin embargo, esto es, en el mejor de los casos, una verdad a medias. **domainname** se usa por lo general para establecer el domino NIS al que pertenece la máquina, que puede ser completamente diferente al del servidor de nombres (DNS). En lugar de ello, para asegurarse de que la forma corta del nombre de su máquina es resoluble por todas las versiones recientes de la orden **hostname**, añádalo como una entrada en su Servidor de Nombres de Dominio (DNS) local, o ponga el FQDN en el fichero `/etc/hosts`. Puede usar entonces el parámetro *--fqdn* de la orden **hostname**, y se imprimirá el FQDN.

5.4. Asignación de una dirección IP

Si configura su software de red para operar su máquina de forma aislada (por ejemplo con el objeto de utilizar el software de noticias de red INN) puede saltarse esta sección pues sólo necesita la dirección de la interfaz de bucle local o loopback, que es siempre 127.0.0.1.

Las cosas son algo más complicadas en redes reales como las Ethernets. Si quiere conectar su computadora a una red, tiene que pedir a los administradores de la misma que le asignen una dirección IP para esa red. Cuando es usted mismo el que está estableciendo la red, tendrá que ser usted quien asigne las direcciones IP.

Las máquinas de una red local deben generalmente compartir direcciones de una subred lógica. Por ello lo primero es asignar una dirección IP para la red. Si tiene varias redes físicas, deberá asignar números de red diferentes a cada una o dividir el rango de direcciones IP disponibles en varias subredes. Las subredes se explican en la siguiente sección, Sección 5.5.

Cuando se está seleccionando un número de red IP, la elección dependerá de si tiene intención de conectarse a Internet en un futuro próximo. Si es así, debería conseguir una dirección IP oficial *ahora*. Pregunte a su proveedor de Internet para conseguir ayuda. Si quiere conseguir un número de red, por si acaso se conecta a Internet algún día, pida un Formulario de Solicitud de Dirección de Red a hostmaster@internic.net, o al Centro de Información de Red de su país, si existe.

Si su red no está conectada a Internet, y no lo va a estar en un futuro cercano, usted es libre de elegir cualquier dirección de red legal. Únicamente asegúrese de que no haya paquetes de su red interna que escapen a la Internet real. Para asegurarse de que no haya perjuicio aunque se escapen paquetes, debería usar uno de los números de red reservados para uso privado. La Autoridad de Números Asignados de Internet (IANA, por sus siglas en inglés) ha reservado varios números de red de las clases A, B y C, que usted puede usar sin registrarlas. Estas direcciones sólo son válidas dentro de su red privada, y no se encaminarán entre sitios de Internet reales. Los números están definidos por RFC 1597 y están listados en Tabla 2−1 en Capítulo 2. Dése cuenta de que el segundo y tercer bloques contienen 16 y 256 redes, respectivamente.

Seleccionar sus direcciones de una de estos números de red no es sólo útil para redes completamente desconectadas de Internet; todavía puede implementar un acceso algo más restringido usando un ordenador como pasarela o gateway. Desde la red local, la pasararela es accesible por su dirección IP interna, pero el mundo exterior la conoce por una dirección oficial (asignada por su proveedor). Volveremos sobre este concepto en la conexión con IP masquerade en Capítulo 11.

Durante el resto del libro, asumiremos que el administrador de red de la cervecera usa un número de red de tipo B, por ejemplo 172.16.0.0. Por supuesto, una red de clase C debería bastar para la red de la cervecera y la vinatera. Usaremos aquí una red de clase B para simplificar; esto hará que los ejemplos de subredes de la siguiente sección del capítulo sean algo más intuitivos.

5.5. Creación de Subredes

Para operar varias redes Ethernet (o de otro tipo una vez que el controlador correspondiente esté disponible), debe dividir su red en subredes. Es importante darse cuenta de que esto es únicamente necesario si tiene mas de una *dirección de "difusión" (broadcast)* en la red; las conexiones punto−a−punto no cuentan. Así, por ejemplo, si tiene una red Ethernet y uno o mas enlaces SLIP con el exterior no hace falta que divida su red. La razón se explica en Capítulo 7.

Para ajustar las dos Ethernets, el administrador de red de la cervecera decide usar ocho bytes la parte de la dirección correspondiente a los ordenadores como dirección de subred. Eso deja otros ocho bytes para las máquinas lo que equivale a 254 por cada subred. Asigna entonces el número de subred 1 a la cervecera, y le da a la vinatera el número 2. Las direcciones de red serán por tanto 172.16.1.0 y 172.16.2.0. La máscara de subred es 255.255.255.0.

A vlager, que actúa de pasarela entre las redes, se le asigna el número de máquina 1 en ambas redes, lo que significa que tiene las direcciones IP 191.72.1.1 y 191.72.2.1, respectivamente.

Es importante notar que en este ejemplo estamos usando una red de clase B para simplificar; una red de tipo C sería más realista. Con el nuevo código de red, la división en subredes no está limitada a nivel de byte, de forma que incluso una red de clase C puede dividirse en varias subredes. Por ejemplo, podría usar 2 bits del byte de los nodos para designar la subred lo que permite implementar cuatro subredes de 64 máquinas cada una.[47]

5.6. Preparación de los ficheros hosts y networks

Una vez ha dividido su red en subredes, debe habilitar un mecanismo simple de resolución de nombres usando el fichero `/etc/hosts`. Si no va a usar los sistemas DNS o NIS para la resolución de nombres, debe poner todos los nombres de las diferentes máquinas en el fichero `hosts`.

Incluso si planea utilizar los servicios DNS y NIS en condiciones normales de operación, es conveniente tener un reducido número de máquinas en `/etc/hosts`. Debe tener algún tipo de resolución de nombres, incluso cuando no hay servicios de red ejecutándose. Este es el caso del arranque. Se trata, no sólo de una cuestión de conveniencia, sino que permite el uso de nombres simbólicos para las máquinas citadas en los scripts de red `rc`. De esta forma, para cambiar las direcciones IP, sólo tiene que copiar el fichero `hosts` actualizado a todas las máquinas y rearrancar, en vez de tener que modificar un gran número de ficheros `rc` por separado. Generalmente, también debe incluir los nombres y direcciones locales en `hosts`, añadiendo todas las pasarelas y servidores NIS usados. [48]

Debería asegurarse de que el subsistema de resolución utiliza la información del fichero `hosts` únicamente. Los ficheros de ejemplo que vienen con su software DNS o NIS pueden producir resultados extraños. Para forzar a que todas las aplicaciones utilicen `/etc/hosts` de forma exclusiva cuando buscan una dirección IP, debe editar el fichero `/etc/host.conf`. Desactive con comentarios cualquier línea que comience por order añadiendo una almohadilla (#) e incluya la siguiente línea

```
order hosts
```

La configuración de la biblioteca de resolución se describe en detalle en Capítulo 6.

El fichero `hosts` contiene un registro por línea, consistente en una dirección IP, un nombre de máquina y de forma opcional, una lista de alias para esa máquina. Los campos se separan por tabuladores o espacios y el campo con la dirección debe empezar en la primera columna. Cualquier cosa a continuación de una almohadilla (#) es interpretada como un comentario y es consecuentemente ignorado.

Los nombres de las máquinas pueden ser con cualificación completa, o relativos al dominio local. Para la máquina vale, el registro generalmente incluiría el nombre con cualificación completa, vale.vbrew.com, y vale en el fichero `hosts`, de forma que pueda ser referido usando el nombre oficial y el nombre local que es más corto.

Este es un ejemplo del aspecto que el fichero `hosts` vlager−if1 y vlager−if2, correspondientes a las direcciones de ambas interfaces de la máquina existentes en vlager:

```
#
# Fichero Hosts de la Cervecera Virtual/Vinatera Virtual
#
# IP             FQDN                 aliases
#
127.0.0.1        localhost
#
172.16.1.1       vlager.vbrew.com      vlager vlager-if1
172.16.1.2       vstout.vbrew.com      vstout
172.16.1.3       vale.vbrew.com        vale
#
172.16.2.1       vlager-if2
172.16.2.2       vbeaujolais.vbrew.com vbeaujolais
172.16.2.3       vbardolino.vbrew.com  vbardolino
172.16.2.4       vchianti.vbrew.com    vchianti
```

Del mismo modo que con las direcciones IP, a veces también puede interesarle usar nombres simbólicos para los números de red. Con este objeto, el fichero `hosts` tiene un compañero llamado `/etc/networks`, que asocia nombres de red con los números correspondientes y viceversa. En la Cervecera Virtual, podríamos instalar un fichero `networks` como éste: [49]

```
# /etc/networks para la cervecera virtual.
brew-net      172.16.1.0
wine-net      172.16.2.0
```

5.7. Interfaz Configuración de la Interfaz para IP

Una vez ha configurado su hardware según se ha explicado en Capítulo 4, debe asegurarse de que el software de red del núcleo conoce esos dispositivos. Hay una serie de órdenes que se usan con objeto de configurar las interfaces de red e inicializar la tabla de encaminamiento. Esas tareas son ejecutadas generalmente por el script de inicialización de red cada vez que el sistema es arrancado. Las herramientas básicas son **ifconfig** (donde if significa interfaz), y **route**.

ifconfig se usa para dar acceso al núcleo a una interfaz. Esto incluye la asignación de una dirección IP y otros parámetros, así como la activación de la interfaz. Por activación nos referimos a permitir que el núcleo envía y recibe datagramas IP a través de la interfaz. El modo más sencillo de invocar esta herramienta es:

```
ifconfig interfaz dirección-ip
```

Esta orden asigna *`dirección-ip`* a *`interfaz`* y la activa. Los otros parámetros toman valores asignados por omisión. Por ejemplo, la máscara de subred por omisión toma el valor correspondiente al tipo de red al que pertenece la dirección IP. Así, tendríamos 255.255.0.0 para una dirección de clase B. **ifconfig** es descrito en detalle en Sección 5.8.

route permite añadir o quitar rutas de la tabla de encaminamiento del núcleo. Se puede invocar como:

```
route [add|del] [-net|-host] destino [if]
```

Los argumentos `add` y `del` determinan, respectivamente si se debe añadir o borrar la ruta hacia *`destino`*. Los argumentos `-net` y `-host` señalan al comando si el destino es una red o una máquina (que es lo que se supone si no se especifica). El argumento `if` es opcional también, y permite especificar a qué interfaz de red se dirige la ruta el núcleo de GNU/Linux hará una conjetura si no se aporta este dato. Este tema se explicará más detalladamente en las siguientes secciones.

5.7.1. La interfaz de bucle local

La primera interfaz en ser activada es la interfaz de bucle local o loopback:

```
# ifconfig lo 127.0.0.1
```

Ocasionalmente, también vera que el nombre comodín localhost es usado en vez de la dirección de IP. **ifconfig** buscará el nombre en el fichero `hosts` que debe contener un registro declarando localhost como nombre válido para la dirección 127.0.0.1:

```
# Registro de ejemplo para localhost en /etc/hosts
localhost     127.0.0.1
```

Para ver la configuración de una interfaz, use **ifconfig**, pasándole como argumento únicamente el nombre de la interfaz:

```
$ ifconfig lo
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
              TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
              Collisions:0
```

Como podrá observar, la máscara asignada a la interfaz del bucle local es 255.0.0.0, debido a que 127.0.0.1 es una dirección de clase A.

Ahora, ya casi puede empezar a jugar con su "mini−red". Sólo queda añadir una entrada en la tabla de encaminamiento que comunique al IP que puede usar esa interfaz como ruta hacia 127.0.0.1. Para llevar esto a cabo, basta escribir:

```
    # route add 127.0.0.1
```

También aquí puede usar localhost en lugar de la dirección IP, suponiendo que lo haya introducido en su `/etc/hosts`.

Lo siguiente es comprobar que todo funciona como es debido, por ejemplo usando **ping**. **ping** es el equivalente a un sonar en una red.[50] Esta orden se usa para verificar que una dirección dada es accesible y para medir el retraso entre el envío de un datagrama y su recepción de vuelta. Este tiempo es conocido como tiempo de ida y vuelta.

```
    # ping localhost
    PING localhost (127.0.0.1): 56 data bytes
    64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.4 ms
    64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.4 ms
    64 bytes from 127.0.0.1: icmp_seq=2 ttl=255 time=0.4 ms
    ^C
    --- localhost ping statistics ---
    3 packets transmitted, 3 packets received, 0% packet loss
    round-trip min/avg/max = 0.4/0.4/0.4 ms
    #
```

Cuando se ejecuta **ping** según se muestra aquí, la emisión de paquetes continúa a menos que sea interrumpida por el usuario. El `^C` marca el momento en el que se apretó Ctrl−C.

Este ejemplo muestra que los paquetes dirigidos a la máquina 127.0.0.1 están siendo entregados correctamente y la respuesta a **ping** es recibida de forma casi instantánea. Esto significa que ha establecido con éxito su primera interfaz de red.

Si la salida de **ping** no se parece a la de más arriba, usted tiene problemas. Compruebe la posibilidad de que algún fichero no haya sido instalado correctamente. Compruebe que los ejecutables **ifconfig** y **route** son compatibles con la versión del núcleo que usa y sobre todo que éste ha sido compilado con la opción de red activada (esto se puede ver comprobando que existe el directorio `/proc/net`). Si el mensaje de error es network unreachable(red inaccesible), seguramente ejecutó la orden **route** incorrectamente. Asegúrese de que es la misma dirección que la que usó con **ifconfig**.

Los pasos descritos arriba son suficientes para poder ejecutar aplicaciones de red en una máquina aislada. Una vez esas líneas son añadidas al script de inicialización de red y después de asegurarse de que es ejecutado en tiempo de arranque, puede proceder a rearrancar su máquina y probar las diferentes aplicaciones de red. Por ejemplo **telnet localhost** debería establecer una conexión **telnet** con su máquina, pidiéndole el nombre de usuario y la contraseña.

Sin embargo, la interfaz de bucle local es útil, no sólo como ejemplo en libros de redes, o como método de pruebas durante el desarrollo: también la utilizan algunas aplicaciones como modo normal de operacion. [51] Por ello, debe usted configurarla siempre, independientemente de que su máquina esté conectada a una red o no.

5.7.2. Interfaces Ethernet

La configuración de una interfaz Ethernet es más o menos igual que la de la interfaz de bucle local. Sólo requiere algunos parámetros más cuando está usando varias subredes.

En la Cervecera Virtual, hemos dividido la red IP, originalmente de clase B, en subredes de clase C. Para que la interfaz reconozca esto, usamos la orden **ifconfig**

```
# ifconfig eth0 vstout netmask 255.255.255.0
```

Esto asigna a la interfaz `eth0` la dirección IP de la máquina vstout (191.72.1.2). Si hubiésemos omitido la máscara de red, **ifconfig** habría deducido la máscara de la clase de la red IP, tomando por tanto 255.255.0.0, que es incorrecto. Una comprobación rápida nos da:

```
# ifconfig eth0
eth0      Link encap 10Mps Ethernet HWaddr  00:00:C0:90:B3:42
          inet addr 172.16.1.2 Bcast 172.16.1.255 Mask 255.255.255.0
          UP BROADCAST RUNNING  MTU 1500  Metric 1
          RX packets 0 errors 0 dropped 0 overrun 0
          TX packets 0 errors 0 dropped 0 overrun 0
```

Puede ver que **ifconfig** ha fijado la dirección de difusión automáticamente (el campo Bcast de arriba) a su valor usual, que es el de la red con todos los bits de la máquina activados. Además se fija la unidad de transferencia de mensajes (tamaño máximo que el núcleo va a generar para esa interfaz) a un máximo de 1500 bytes. Todos estos valores pueden ser especificados mediante opciones especiales que se explican en Sección 5.8.

De forma semejante al caso de la interfaz de bucle local, debe también establecer ahora una entrada en la tabla de encaminamiento que informe al núcleo de que la red es accesible mediante `eth0`. Para la Cervecera Virtual, ejecutaría:

```
# route add -net 172.16.1.0
```

Inicialmente podría parecer algo mágico, pues no esta claro como **route** detecta cuál es la interfaz que debe usar. Sin embargo el truco es sencillo: el núcleo comprueba todas las interfaces que han sido configuradas hasta el momento y compara la dirección de destino (191.72.1.0 en este caso) con la parte de red de las direcciones de las interfaces (o, lo que es lo mismo, ejecuta un "and" lógico de la dirección de la interfaz y la máscara de red). La única interfaz que cumple esto es `eth0`.

Veamos, ¿qué significa la opción `net`? Esta opción es necesaria porque el programa **route** es capaz de trabajar con rutas a redes o a máquinas concretas (como vimos arriba en el caso de localhost). Cuando la dirección es dada en notación de cuaterna, intenta adivinar si se trata de una red o una máquina fijándose en los bits de máquina de la dirección. Si esa parte es nula, **route** asume que se trata de una red, y de otro modo lo toma como dirección de una máquina. Por tanto, route supondría que 191.72.1.0 es la dirección de una máquina en vez de una red, debido a que no sabe que hemos dividido el espacio de direcciones en subredes. Por tanto hemos de decírselo de forma explícita utilizando el indicador `net`.

Por supuesto, escribir el comando **route** es tedioso y susceptible de muchos errores de escritura. Un método más conveniente es usar los nombres definidos en `/etc/networks` como vimos más arriba. Esto hace el comando más inteligible; de este modo incluso podemos evitar escribir el indicador `net`, porque **route** sabe que 191.72.1.0 representa una red:

```
# route add brew-net
```

Una vez finalizados los pasos básicos de configuración, debemos asegurarnos de que la interfaz Ethernet está funcionando correctamente. Elija una máquina de su red, por ejemplo vlager, y escriba:

```
# ping vlager
PING vlager: 64 byte packets
64 bytes from 172.16.1.1: icmp_seq=0. time=11. ms
```

```
64 bytes from 172.16.1.1: icmp_seq=1. time=7. ms
64 bytes from 172.16.1.1: icmp_seq=2. time=12. ms
64 bytes from 172.16.1.1: icmp_seq=3. time=3. ms
^C
----vstout.vbrew.com PING Statistics----
4 packets transmitted, 4 packets received, 0
round-trip (ms)  min/avg/max = 3/8/12
```

Si el resultado no es similar a éste, algo va mal, obviamente. Una tasa de pérdida de paquetes inusualmente alta, sugiere un problema de hardware, como terminaciones en mal estado o incluso la ausencia de las mismas, etc. Si no recibe ningún paquete, debe comprobar la configuración de la interfaz mediante **netstat**, que describiremos después en Sección 5.9. Las estadísticas de paquetes producidas por **ifconfig** le indican si algún paquete ha sido enviado mediante esa interfaz. Si tiene acceso a una máquina remota, también debería dirigirse a esa máquina y comprobar las estadísticas de la interfaz. De este modo puede determinar exactamente en qué momento se han descartado los paquetes. Además, debe consultar la información de encaminamiento con **route** para ver si ambas máquinas han registrado ésta correctamente en sus tablas. **route** imprime la tabla de encaminamiento del núcleo completa si se ejecuta sin argumentos (la opción `n` hace que utilice la notación de cuaternas en vez de los nombres de las máquinas):

```
# route -n
Kernel routing table
Destination  Gateway  Genmask         Flags Metric Ref Use    Iface
127.0.0.1    *        255.255.255.255 UH    1      0      112 lo
172.16.1.0   *        255.255.255.0   U     1      0       10 eth0
```

El significado de cada uno de los campos se detalla más adelante en Sección 5.9. La columna Flags contiene una lista de los indicadores activos en cada interfaz. U indica que la interfaz está activa y H indica que la dirección de destino es una máquina. Si encuentra que el indicador H se ha activado para una ruta que pretendía usar para una red, entonces debe usar la opción `net` con el comando **route**. Para comprobar si alguna ruta esta siendo usada o no, debe mirar si el campo Use en la penúltima columna se incrementa entre dos ejecuciones sucesivas de **ping**.

5.7.3. Encaminamiento a través de una pasarela

En la sección anterior, cubrimos sólo el caso en el que la máquina sólo tiene una única Ethernet. Frecuentemente, es posible encontrar redes conectadas unas a otras a través de pasarelas o máquinas de enlace. Estas pasarelas pueden simplemente unir dos o más Ethernets, pero pueden también servir de enlace con el exterior, con Internet. Para usar una pasarela, es necesario añadir información adicional a la capa de red.

Por ejemplo, las Ethernets de la Cervecera Virtual y de la Vinatera Virtual están unidas a través de una pasarela, vlager. Suponiendo que la máquina vlager ha sido configurada ya, sólo tenemos que añadir otro registro a la tabla de encaminamiento de la máquina vstout que le comunique al núcleo que puede acceder a todas los nodos de la red de la Vinatera a través de vlager. La orden apropiada usando **route** se muestra a continuación; la palabra clave gw indica que el argumento siguiente es una pasarela:

```
# route add wine-net gw vlager
```

Por supuesto, cualquier nodo en la red de la Vinatera al que quiera dirigirse debe tener un registro análogo referido a la red de la Cervecera, o de otro modo sólo podría enviar datos a la red de la Vinatera desde la Cervecera, pero las máquinas de la Vinatera serían incapaces de responder.

Este ejemplo describe únicamente una pasarela que conmuta paquetes entre dos redes Ethernet aisladas. Supongamos ahora que vlager también tiene una conexión a la Internet (digamos que a través de un enlace SLIP). Nos gustaría que los datagramas destinados a *cualquier* dirección fuera de la red de la Cervecera fueran entregados a vlager. Esto se puede conseguir convirtiéndolo en la pasarela por omisión para vstout:

```
# route add default gw vlager
```

El nombre de red default es una abreviatura que representa la red 0.0.0.0, o ruta por omisión. La ruta por omisión analiza cada destino, y es la que será usada si no se encuentra ninguna ruta más específica. No es necesario añadir este nombre a `/etc/networks`, porque esta información esta contenida en el código de **route**.

Una tasa alta de pérdida de paquetes usando **ping** hacia una máquina situada detrás de una o más pasarelas, puede deberse a que la red está muy congestionada. La pérdida de paquetes no se debe tanto a deficiencias técnicas como a exceso temporal de carga en las máquinas que actúan de enlace, provocando retrasos o incluso el descarte de datagramas entrantes.

5.7.4. Configuración de una Pasarela

Configurar una máquina para conmutar paquetes entre dos Ethernets es bastante sencillo. Suponga que nos encontramos en vlager, que contiene dos tarjetas Ethernet, respectivamente conectadas a cada una de las dos redes. Todo lo que necesitará hacer es configurar ambas interfaces de forma separada, dándole a cada una su dirección IP correspondiente, y eso es todo.

Es bastante útil incluir la información de ambas interfaces en el fichero `hosts` del modo indicado a continuación, de forma que tengamos nombres para referirnos a ellas también:

```
172.16.1.1      vlager.vbrew.com    vlager vlager-if1
172.16.2.1      vlager-if2
```

La secuencia de comandos para establecer las dos intefaces es por tanto:

```
# ifconfig eth0 vlager-if1
# route add brew-net
# ifconfig eth1 vlager-if2
# route add wine-net
```

Si esta secuencia no funciona, asegúrese de que el núcleo ha sido compilado con el soporte para transmisión IP (IP forwarding). Una buena forma de hacerlo es comprobar que el primer número de la segunda línea de `/proc/net/snmp` es un `1`.

5.7.5. La interfaz PLIP

Si usa un enlace PLIP para conectar dos máquinas, las cosas son un poco diferentes de lo visto para una Ethernet. En caso de PLIP se trata de un enlace conocido como *punto−a−punto*, lo que significa que sólo hay una máquina a cada extremo del enlace. A las redes como Ethernet se les llama redes de *difusión*. La configuración de enlaces punto a punto es diferente porque a diferencia de las redes de difusión, los enlaces punto a punto no son una red por sí mismos.

PLIP ofrece conexión muy barata y portable entre ordenadores. A modo de ejemplo, consideremos un ordenador portátil de un empleado en la Cervecera Virtual que se conecta a vlager mediante PLIP. El portátil se llama vlite, y tiene un único puerto paralelo. Durante el arranque, este puerto será registrado como `plip1`. Para activar el enlace, ha de configurar la interfaz `plip1` mediante las órdenes siguientes:[52]

```
# ifconfig plip1 vlite pointopoint vlager
# route add default gw vlager
```

La primera orden configura la interfaz, diciéndole al núcleo que se trata de un enlace punto−a−punto, donde la parte remota tiene la dirección de vlager. El segundo instala la ruta por omisión que usa a vlager como pasarela. En vlager se necesita ejecutar **ifconfig** con argumentos similares para activar el enlace (en este caso no es necesario usar **route**):

```
# ifconfig plip1 vlager pointopoint vlite
```

Es interesante notar que la interfaz `plip1` en vlager no necesita tener una dirección IP diferente, sino que puede usar la misma dirección 172.16.1.1. Las redes punto−a−punto no representan directamente una red, así que las interfaces no necesitan una dirección en ninguna red soportada. El núcleo usa la información de la interfaz que hay en la tabla de enrutamiento para prevenir cualquier posible equivocación. [53]

Una vez hemos configurado el encaminamiento desde el portátil a la red de la Cervecera, sólo resta arbitrar un modo para que cualquier máquina en esa red pueda acceder a vlite. Un modo particularmente enrevesado seríia añadir una ruta a las tablas de encaminamiento de cada una de las máquinas de la red para usar vlager como pasarela hacia vlite:

```
# route add vlite gw vlager
```

Una opción mejor cuando tenemos que trabajar con rutas temporales es usar encaminamiento dinámico. Una forma de conseguirlo es usando **gated**, un demonio de encaminamiento, que deberá instalar en cada una de las máquinas de la red de modo que distribuya la información de encaminamiento de forma dinámica. La forma más sencilla, sin embargo, consiste en usar *proxy ARP*. Con la sustitución ARP, vlager responde a cualquier pregunta ARP dirigida a vlite enviando su propia dirección Ethernet. El efecto conseguido es que todos los paquetes dirigidos a vlite terminan yendo a vlager, que se encarga de reenviárselos al portátil. Volveremos a hablar de sustitución ARP en la sección Sección 5.10.

Las actuales versiones de `net-tools` contienen una herramienta llamada **plipconfig**, que permite configurar algunos parámetros que permiten ajustar ciertos parámetros del la temporización de PLIP. La IRQ que va a usarse por el puerto de la impresora puese especificarse usando la orden **ifconfig**.

5.7.6. Las interfaces SLIP y PPP

A pesar de que los enlaces SLIP y PPP son simples enlaces punto−a−punto igual que las conexiones PLIP, hay mucho más que decir de ellas. Generalmente, el establecimiento de un enlace SLIP incluye una llamada a un lugar de conexión remoto a través de un módem y el establecimiento del modo SLIP en la línea de comunicaciones serie. El uso de PPP es similar. Las herramientas necesarias para establecer un enlace SLIP o PPP se describen en Capítulo 7 y Capítulo 8.

5.7.7. La Interfaz Comodín

La interfaz comodín (dummy) parece un tanto exótica y sin embargo es bastante útil. Resulta especialmente ventajosa para máquinas aisladas y para las que se conectan a una red IP mediante un enlace telefónico. Se trata en realidad de máquinas que trabajan de forma aislada la mayor parte del tiempo.

El dilema con las máquinas aisladas es que el único dispositivo activo es el de bucle local, al que generalmente se le asigna la dirección 127.0.0.1. En ocasiones, sin embargo, le resultará necesario enviar datos a la dirección IP oficial de la máquina. Supongamos, por ejemplo, el caso del portátil vlite cuando no esta conectado a ninguna red. Una aplicación en vlite puede querer enviar datos a otra aplicación en la misma máquina. Buscar vlite en `/etc/hosts` dará como resultado 172.16.1.65, y por tanto intentará enviar los datos a esa dirección. Como la única interfaz activa en ese momento es la de bucle local, el núcleo no sabe que la dirección se refiere a la misma máquina. En consecuencia el núcleo descarta el datagrama y genera un error en la aplicación.

En esta situación es cuando la interfaz comodín es útil, resolviendo el dilema actuando como alter ego de la interfaz de bucle local. En el caso de vlite, simplemente debe asignarle la dirección 172.16.1.65 y añadir una ruta que apunte a ella. Cada datagrama para 172.16.1.65 es enviado entonces localmente. La forma correcta es pues:[54]

```
# ifconfig dummy vlite
# route add vlite
```

5.7.8. Alias de IP

Los nuevos núcleos llevan una funcionalidad que puede sustituir por completo a la interfaz comodín, y que tiene otras útiles funciones. *IP Alias* permite configurar múltiples direcciones IP en un sólo dispositivo físico. En el caso más simple, usted puede reproducir la función de la interfaz comodín configurando la dirección del nodo como un alias de la interfaz de bucle local, y evitar por completo usar la intefaz comodín. Para usos más complejos, usted puede configurar su máquina para simular ser varias máquinas, cada una con su propia dirección IP. Esta configuración es llamaba a veces Hosting Virtual, aunque técnicamente se usa también para otras muchas técnicas.[55]

Para configurar un alias para una interfaz, primero debe asegurarse de que su núcleo ha sido compilado con soporte para Alias de IP (compruebe que tiene un fichero `/proc/net/ip_alias` ; si no es así, debe recompilar el núcleo). La configuración de un alias de IP es virtualmente idéntica a la configuración de un dispositivo de red real; se usa un nombre especial para indicar que lo que usted quiere es un alias. Por ejemplo:

```
# ifconfig lo:0 172.16.1.1
```

Esta orden creará un alias para la interfaz de bucle local con la dirección `172.16.1.1`. Los alias de IP se señalan anteponiendo :*n* al dispositivo actual de red, donde n es un entero. En nuestro ejemplo, el dispositivo de red donde estamos creando el alias es `lo`, y estamos creando un alias numerado como cero para él. De esta forma, un único dispositivo físico puede soportar varios alias.

Cada alias debe ser tratado como si fuera un dispositivo diferente, y en lo referente al software de IP del núcleo, así es; por más que esté compartiendo su hardware con otro interfaz.

5.8. Todo sobre ifconfig

El programa **ifconfig** tiene muchos más parámetros que los descritos hasta ahora. Generalmente se ejecuta en la forma:

```
ifconfig interfaz [dirección [parámetros]]
```

`interfaz` es el nombre de la interfaz y *`dirección`* es la dirección IP que se asigna a dicha interfaz. La dirección puede estar en forma de cuaterna o usando un nombre que **ifconfig** buscará en `/etc/hosts`.

Si **ifconfig** es ejecutado añadiendo únicamente el nombre de la interfaz, presentará la información de la configuración de dicha interfaz. Si se ejecuta sin parámetros, presenta todas las interfaces configuradas hasta el momento; usando la opción `a` fuerza a ifconfig a incluir la información de las interfaces inactivas. A modo de ejemplo, la consulta de la configuración de la interfaz Ethernet `eth0` sería:

```
# ifconfig eth0
eth0      Link encap 10Mbps Ethernet  HWaddr 00:00:C0:90:B3:42
          inet addr 172.16.1.2 Bcast 172.16.1.255 Mask 255.255.255.0
          UP BROADCAST RUNNING  MTU 1500  Metric 0
          RX packets 3136 errors 217 dropped 7 overrun 26
          TX packets 1752 errors 25 dropped 0 overrun 0
```

Los campos `MTU` y `Metric` informan sobre los valores actuales de la MTU (Unidad Máxima de Transferencia) y de la métrica para una interfaz dada. El valor de la métrica es usado tradicionalmente por algunos sistemas operativos para calcular el coste de una ruta. GNU/Linux no usa este valor por el momento, pero lo define por razones de compatibilidad.

Las líneas `RX` y `TX` dan idea de los paquetes recibidos o transmitidos sin errores, del número de errores ocurridos, de cuántos paquetes han sido descartados (seguramente por memoria insuficiente), y cuántos han sido perdidos por desbordamiento, condición que ocurre cuando la recepción de paquetes es demasiado rápida y el núcleo es incapaz de dar servicio al paquete anterior antes de la llegada del nuevo paquete. Los nombres de los campos que genera **ifconfig** coinciden más o menos con los parámetros con los que se puede ejecutar;

estos parámetros son explicados más abajo.

A continuación tenemos una lista de los parámetros reconocidos por **ifconfig**. Las opciones que simplemente activan alguna característica pueden usarse para desactivarla precediéndolas de un guión ().

up
: Marca la interfaz como disponible para que sea usada por la capa IP. Esta opción va implícita cuando lo que se da en la línea de órdenes es una *dirección*. También permite reactivar una interfaz que se ha desactivado temporalmente mediante la opción `down`.

  Esta opción corresponde a los indicadores `UP` y `RUNNING`.

down
: Marca la interfaz como inaccesible a la capa IP. Esto inhabilita cualquier tráfico IP a través de la interfaz. Es importante darse cuenta que esto también borra los registros de la tabla de encaminamiento correspondientes a esa interfaz de forma automática.

netmask máscara
: Esto asigna una máscara de subred a una interfaz. Se puede dar como un valor de 32 bits en hexadecimal precedido del prefijo 0x, o en notación de cuaterna usando números decimales separados por puntos. Aunque la notación en forma de cuaterna es más común, la representación hexadecimal es muchas veces más fácil de usar. Las máscaras de red son esencialmente binarias, y es más fácil hacer una conversión binario−a−hexadecimal que una binario−a−decimal.

pointopoint dirección
: Esta opción se usa para enlaces IP punto−a−punto en los que intervienen únicamente dos máquinas. Esta opción es necesaria para, por ejemplo, configurar las interfaces SLIP o PLIP. Si se ha definido una dirección punto a punto,**ifconfig** muestra el indicador `POINTOPOINT`.

broadcast dirección
: La dirección de difusión se obtiene, generalmente, usando la parte de red de la dirección y activando todos los bits de la parte correspondiente a la máquina. Algunas implementaciones de los protocolos IP (por ejemplo, sistemas derivados de BSD 4.2) utilizan un esquema diferente; esta opción proporciona un método para adaptarse a esos entornos más raros. **ifconfig** confirma el establecimiento de una dirección de difusión incluyendo el indicador `BROADCAST`.

irq
: Esta opción permite establecer la línea de IRQ usado por ciertos dispositivos. Esto es especialmente útil para PLIP, pero también puede ser de utilidad para algunas tarjetas Ethernet.

metric número
: Esta opción puede ser usada para asignar un valor de métrica a la tabla de encaminamiento creada para la interfaz. Esta métrica es usada por el Protocolo de Información de Encaminamiento (RIP) para construir las tablas de encaminamiento para la red.[56] El valor usado por omisión por **ifconfig** es cero. Si no esta ejecutando un demonio RIP, no necesita usar esta opción para nada; si por el contrario lo usa, sólo tendrá que modificar este valor en contadas ocasiones.

mtu bytes
: Esto fija la unidad máxima de transferencia, o lo que es lo mismo, el máximo número de octetos que la interfaz es capaz de manejar en una única transacción. Para Ethernets, la MTU toma el valor 1500 por omisión (que es el tamaño máximo permitido para un paquete Ethernet); para interfaces tipo SLIP, el valor por defecto es 296. No hay tamaño límite para el MTU en enlaces SLIP, pero este valor es una buena garantía.

arp

Esta opción es específica de redes de difusión como las Ethernets o las de radio−paquetes. Permite el uso de ARP, el Protocolo de Resolución de Direcciones, para detectar la dirección física de las máquinas conectadas a la red. Para redes de difusión, esta opción es habilitada por omisión. Si ARP está desactivado,**ifconfig** muestra el indicador `NOARP`.

arp

Inhabilita el uso de ARP para esta interfaz.

promisc

Pone la interfaz en modo promiscuo. En una red de difusión, esto hace que la interfaz reciba todos los paquetes, independientemente de si eran para ella o no. Esto permite el análisis del tráfico de red utilizando utilidades como filtros de paquetes, también llamado *fisgoneo de Ethernet*. Se trata de una buena técnica para localizar problemas de red que de otra forma resultan difíciles de detectar. Herramientas como **tcpdump** se basan en esto.

Por otro lado, esta opción permite a los atacantes hacer cosas feas, como filtrar el tráfico de su red en busca de contraseñas. Usted puede protegerse contra este tipo de ataques simplemente prohibiendo que nadie conecte un ordenador en la red. También puede usar protocolos de autentificación segura, como Kerberos o ssh (secure shell). [57] Esta opción corresponde al indicador`PROMISC`.

promisc

Esta opción apaga el modo promiscuo.

allmulti

Las direcciones de envío múltiple (multicast) son como las direcciones de difusión de Ethernet, excepto que en lugar de incluir automáticamente a todo el mundo, los únicos que reciben paquetes enviados a una dirección de envío múltiple son aquellos programados para escucharla. Esto es útil para aplicaciones como videoconferencia basada en Ethernet o audio para red, en los que sólo los interesados pueden escuchar. Las direcciones de envío múltiple están soportadas por casi todas las controladoras Ethernet (pero no todas). Cuando esta opción está activa, la interfaz recibe y envía paquetes de envío múltiple para su proceso. Esta opción corresponde al indicador `ALLMUTI`.

allmulti

Esta opción desactiva las direcciones de envío múltiple.

5.9. La orden netstat

netstat es una herramienta útil para comprobar la configuración y actividad de su red. Se llama netstat, aunque se trata en realidad de una colección de herramientas combinadas. Describiremos cada una de las funciones en las secciones siguientes.

5.9.1. Consulta de la tabla de encaminamiento

Si ejecuta **netstat** usando el indicador `r`, puede ver la información de la tabla de encaminamiento del núcleo igual que hemos venido haciendo hasta ahora con **route**. Para vstout, tendríamos:

```
# netstat −nr
Kernel IP routing table
Destination   Gateway      Genmask         Flags  MSS Window  irtt Iface
127.0.0.1     *            255.255.255.255 UH       0 0          0 lo
172.16.1.0    *            255.255.255.0   U        0 0          0 eth0
172.16.2.0    172.16.1.1   255.255.255.0   UG       0 0          0 eth0
```

La opción −n hace que **netstat** imprima las direcciones IP en notación de cuaterna en vez de usar los nombres simbólicos de las máquinas o las redes. Esto es especialmente útil si pretende evitar consultas para esos nombres a través de la red (por ejemplo consultas a un servidor DNS o NIS).

La segunda columna de la salida producida por **netstat** informa sobre las pasarelas a las que apunta la información de encaminamiento. Si una ruta no usa pasarela, el programa imprime un asterisco. La tercera columna imprime el nivel de generalización de una ruta. Dada una dirección IP para la que encontrar una ruta apropiada, el núcleo recorre la tabla registro a registro haciendo un "AND" lógico de la dirección y la máscara de nivel de generalización antes de compararla con el destino que muestra dicho registro.

La cuarta columna muestra varios indicadores que describen la ruta:

G
: La ruta utiliza una pasarela.

U
: La interfaz está activa.

H
: Esta interfaz permite el acceso a una sola máquina. Este es el caso de la interfaz de bucle local 127.0.0.1.

D
: Esta ruta es creada dinámicamente. Aparece si la entrada de la tabla ha sido generada por un demonio de encaminamiento como **gated** o por un mensaje de redirección ICMP (ver la sección Sección 2.5 en el capítulo 2).

M
: Presente cuando este registro ha sido modificado por un mensaje de redirección ICMP.

!
: La ruta es una ruta de rechazo, y los datagramas serán descartados.

Las siguientes tres columnas muestran el MSS, tamaño de ventana y irtt que serán aplicados a las conexiones TCP establecidas a través de esta ruta. El MSS es el Tamaño Máximo de Segmento, y es el tamaño del datagrama más grande que construirá el núcleo para transmitir a través de esta ruta. La Ventana es la cantidad máxima de datos que el sistema aceptará de una sola vez desde una máquina remota. El acrónimo `irtt` significa tiempo inicial de ida y vuelta, por sus iniciales en inglés. El protocolo TCP se asegura de que los datos han sido transmitidos de forma fiable entre máquinas retransmitiendo un datagrama si éste ha sido perdido. El protocolo TCP mantiene un contador de cuánto tarda un datagrama en ser enviado a su destino, y el "recibo" que se recibe, de forma que sabe cuánto esperar antes de suponer que un datagrama necesita retrasmitirse. Este proceso se llama tiempo de ida y vuelta. El tiempo de ida y vuelta inicial es el valor que el protocolo TCP usará cuando se establezca una conexión por primera vez. Para la mayoría de los tipos de redes, el valor por omisión es válido, pero para algunas redes lentas, especialmente ciertos tipos de redes de radiopaquetes de aficionados, el tiempo es demasiado pequeño y causa retransmisiones innecesarias. El valor de `irtt` puede ajustarse usando el comando **route**. Los campos a 0 significan que se está usando el valor por omisión.

Para terminar, el último campo muestra el interfaz de red que usará esta ruta.

5.9.2. Consulta de las estadísticas de una interfaz

Cuando se invoca con el indicador `i` **netstat** presenta las estadísticas para las interfaces de red configuradas en ese momento. Si también se pasa la opción `a`, mostrará *todas* las interfaces presentes en el núcleo, y no sólo aquellas que hayan sido configuradas. En vstout, la salida para **netstat** sería algo así:

```
# netstat −i
Kernel Interface table
Iface MTU Met  RX−OK RX−ERR RX−DRP RX−OVR  TX−OK TX−ERR TX−DRP TX−OVR Flags
lo      0   0   3185      0      0      0   3185      0      0      0 BLRU
eth0 1500   0 972633     17     20    120 628711    217      0      0 BRU
```

Los campos `MTU` y `Met` muestran los valores actuales de MTU y de métrica para esa interfaz. Las columnas `RX` y `TX` muestran cuántos paquetes han sido recibidos o transmitidos sin errores (`RX−OK`/`TX−OK`) o dañados (`RX−ERR`/`TX−ERR`); cuántos fueron descartados (`RX−DRP`/`TX−DRP`); y cuántos se perdieron por un desbordamiento. (`RX−OVR`/`TX−OVR`).

La última columna muestra los indicadores activos para cada interfaz. Son abreviaturas del nombre completo del indicador, que se muestran con la configuración de la interfaz que ofrece **ifconfig**:

B
: Dirección de difusión activa.

L
: La interfaz es un dispositivo de bucle local.

M
: Se reciben todos los paquetes (modo promiscuo).

O
: ARP no funciona para esta interfaz.

P
: Conexión punto a punto.

R
: La interfaz funciona.

U
: La interfaz está activa.

5.9.3. Mostrar conexiones

netstat ofrece una serie de opciones para mostrar los puertos activos o pasivos. Las opciones `t`, `u`, `w`, y `x` muestran conexiones activas a puertos TCP, UDP, RAW, o Unix. Si incluye además el indicador `a`, se mostrarán también los puertos que estén esperando una conexión (es decir, que estén escuchando). Esto le dará una lista de todos los servidores que estén corriendo actualmente en su sistema.

Llamar a **netstat −ta** en vlager produce esta salida:

```
$ netstat −ta
Active Internet Connections
Proto Recv−Q Send−Q Local Address    Foreign Address    (State)
tcp        0      0 *:domain         *:*                LISTEN
tcp        0      0 *:time           *:*                LISTEN
tcp        0      0 *:smtp           *:*                LISTEN
tcp        0      0 vlager:smtp      vstout:1040        ESTABLISHED
tcp        0      0 *:telnet         *:*                LISTEN
tcp        0      0 localhost:1046   vbardolino:telnet  ESTABLISHED
tcp        0      0 *:chargen        *:*                LISTEN
tcp        0      0 *:daytime        *:*                LISTEN
tcp        0      0 *:discard        *:*                LISTEN
tcp        0      0 *:echo           *:*                LISTEN
tcp        0      0 *:shell          *:*                LISTEN
```

```
tcp        0      0 *:login           *:*                    LISTEN
```

Esta salida muestra que la mayoría de los servidores están simplemente esperando una conexión externa. Sin embargo, la cuarta línea muestra una conexión SMTP desde vstout, y la sexta línea le indica que usted está haciendo una conexión telnet a vbardolino.[58]

El indicador `a` por sí sólo indicará todos los sockets de todo tipo.

5.10. Comprobación de las tablas ARP

En ciertas ocasiones, es útil poder ver o alterar el contenido de las tablas ARP del núcleo, por ejemplo, cuando usted sospecha que una dirección IP duplicada es la casa de algún problema intermitente en su red. La herramienta **arp** se hizo para situaciones como ésta. Sus opciones son:

```
arp [-v] [-t tipohw] -a [hostname]
arp [-v] [-t tipohw] -s hostname dirección hardware
arp [-v] -d máquina [hostname]
```

Todos los argumentos `hostname` pueden ser nombres simbólicos, o direcciones IP en notación de cuaterna.

El primer comando muestra el registro de la tabla correspondiente a la dirección IP o máquina especificada, o si no se pasa ninguna, se mostrarán todos los registros. Por ejemplo, al invocar **arp** en vlager obtendríamos:

```
# arp -a
IP address       HW type                HW address
172.16.1.3       10Mbps Ethernet        00:00:C0:5A:42:C1
172.16.1.2       10Mbps Ethernet        00:00:C0:90:B3:42
172.16.2.4       10Mbps Ethernet        00:00:C0:04:69:AA
```

Que muestra las direcciones Ethernet de vlager, vstout y vale.

Se puede limitar el listado a un tipo de hardware especificado usando la opción `t`. Los valores posibles son ether, ax25, o pronet, y se refieren a Ethernet de 10Mbps, AMPR AX.25, y equipos token ring IEEE 802.5, respectivamente.

La opción `s` se usa para añadir permanentemente la dirección Ethernet de la máquina especificada a las tablas ARP. El argumento `dirección hawdware` especifica la dirección de hardware, que por omisión se supone que es una dirección Ethernet especificada como seis bytes en hexadecimal separados por dos puntos. Usted puede incluso definir las direcciones de hardware para otros tipos de hardware, usando la opción `t`.

Por alguna razón, las peticiones ARP para máquinas remotas fallan algunas veces, por ejemplo cuando el controlador ARP no funciona, o cuando alguna otra máquina se identifica erróneamente como si ella misma tuviera esa dirección IP. Este problema requiere que usted añada manualmente una dirección IP en la tabla ARP. También es una forma (muy drástica) de protegerse a sí mismo de otras máquinas de su Ethernet que tratan de hacerse pasar por otras.

El uso de **arp** con el modificador `d` borra todas las entradas ARP referentes a la máquina dada. Este modificador puede ser usado para forzar a la interfaz a intentar obtener la dirección Ethernet correspondiente a la dirección IP en cuestión. Esto es útil cuando un sistema mal configurado ha emitido una información ARP errónea (por supuesto, usted debe reconfigurar la máquina estropeada primero).

La opción `s` también puede usarse para implementar un *proxy* ARP. Esta es una técnica especial, en la que una máquina, llamémosla gate, actúa como una pasarela a otra máquina llamada fnord simulando que las dos direcciones hacen referencia a la misma máquina, en este caso gate. Esto se consigue incluyendo una entrada ARP para fnord que apunte a su propia interfaz Ethernet. Cuando una máquina envíe una petición ARP para fnord, gate devolverá una respuesta con su propia dirección Ethernet. La máquina que hizo la petición enviará entonces todos los datagramas a gate, que se los pasará a fnord.

Estas virguerías pueden ser necesarias cuando usted quiera acceder a fnord desde una máquina DOS con una implementación errónea de TCP, que no entienda el enrutado demasiado bien. Cuando use proxy ARP, éste le aparecerá a la máquina DOS como si fnord estuviera en la subred local, así que no tiene que saber cómo enrutar a través de una pasarela.

Otra aplicación útil del proxy ARP es cuando una de sus máquinas actúe como una pasarela para otra máquina sólo temporalmente, por ejemplo a través de un enlace telefónico. En un ejemplo anterior, ya nos encontramos con que el portátil vlite se conectaba a vlager a través de un enlace PLIP de vez en cuando. Por supuesto, esta aplicación servirá sólo si la dirección de la máquina para la que quiere actuar como proxy ARP está en la misma subred que su pasarela. vstout podría hacer de proxy ARP para cualquier máquina de la red de la Cervecera (172.16.1.0), pero nunca para una máquina de la red de la Vinatera (172.16.2.0).

La invocación adecuada para hacer de proxy ARP para fnord se da abajo. Por supuesto, la dirección Ethernet dada debe ser la de gate:

```
# arp -s fnord 00:00:c0:a1:42:e0 pub
```

Para borrar el registro del proxy ARP:

```
# arp -d fnord
```

Capítulo 6. El servicio de nombres y su configuración

Como se comentó en Capítulo 2, la red TCP/IP puede utilizar diferentes métodos para convertir nombres en direcciones IP. El mecanismo más simple consiste en almacenar los nombres en una tabla de máquinas en el fichero `/etc/hosts`. Esto es únicamente interesante en el caso de pequeñas redes de área local que sólo requieran la administración de una persona, y que no tengan tráfico IP con el mundo exterior. Recordamos que el formato del fichero `hosts` fue descrito en Capítulo 5.

Alternativamente, puede utilizarse BIND el *servicio de nombres Internet de Berkeley* o ``Berkeley Internet Name Domain'', para traducir nombres de máquinas a direcciones IP (cosa que también se conoce como *resolución*). Configurar BIND puede ser una laboriosa tarea pero, una vez hecho, los cambios en la topología de la red serán mucho más fáciles de hacer. En GNU/Linux, como en muchos otros sistemas Unix, el servicio de nombres se realiza mediante un programa llamado **named**. Al iniciarse, carga un conjunto de ficheros maestros en su *caché* y espera peticiones de procesos locales o remotos. Existen distintas maneras de preparar BIND, y no es necesario ejecutar un servidor de nombres en cada máquina: generalmente, uno para toda la red es suficiente.

Este capítulo le dará ideas generales acerca de cómo configurar y ejecutar un servidor de nombres. Para un uso normal deberá bastarle esto, junto a la documentación contenida en las fuentes de BIND, páginas de manual y la guía *BIND Operator's Guide* (BOG). Si pretende usar BIND en un entorno más complejo que una pequeña red local (tal vez con conexión a Internet) debería echar un vistazo a un buen libro sobre BIND, como *DNS y BIND* de Paul Albitz y Cricket Liu (O'Reilly). También existe un grupo de *noticias* para cuestiones sobre DNS: el grupo comp.protocols.tcp−ip.domains. Para abundar más en los detalles técnicos, vea las RFCs 1033, 1034, y 1035.

6.1. La biblioteca de resolución

Cuando hablamos del *sistema de resolución*, no nos referiremos a una aplicación en particular, sino a la *biblioteca de resolución*: un conjunto de funciones que pueden encontrarse en las bibliotecas estándar del lenguaje C. Las rutinas principales son `gethostbyname(2)` y `gethostbyaddr(2)`, que buscan la dirección IP de una máquina a partir del nombre y viceversa. Es posible configurarlas para que simplemente miren en el fichero `hosts` local (o remoto, si se usa NIS).

Las funciones del sistema de resolución leen ficheros de configuración cuando son llamadas. Desde estos ficheros, determinan qué bases de datos hay que interrogar, en qué orden y otros detalles relevantes. En la antigua biblioteca libc de GNU/Linux, se utilizaba el fichero `/etc/host.conf` como fichero maestro, pero en la versión 2 de las bibliotecas, la glibc, se utiliza el fichero `/etc/nsswitch.conf`. Vamos a describir ambas formas, puesto que son muy usuales.

6.1.1. El fichero host.conf

El fichero `host.conf` se encuentra en el directorio `/etc` e indica al sistema de resolución qué servicios debe usar y en qué orden.

Las opciones del fichero `host.conf` deben estar en líneas distintas. Los campos deben separarse por blancos (espacios o tabuladores). Un símbolo almohadillado (#) supone desde ese punto hasta el final de la línea un comentario del fichero. Las opciones disponibles son las siguientes:

order
: Determina el orden en el que los servicios de resolución se usan. Opciones válidas son bind para usar el servidor de nombres, `/etc/hosts` para buscar en `/etc/hosts` y nis para buscar con NIS. Puede especificarse cualquiera de las anteriores, y el orden de aparición determina qué servicio se prueba en primer lugar para intentar resolver el nombre.

multi
: Va con las opciones on u off. Determina si una máquina del fichero `/etc/hosts` puede tener distintas direcciones IP o no. Esta opción no tiene efecto en peticiones via NIS o DNS.

nospoof
: Como se explicó en la sección Sección 6.2.4, DNS le permite encontrar un nombre de máquina perteneciente a una dirección IP utilizando el dominio in−addr.arpa. Los intentos de los servidores de nombres de proporcionar un nombre falso se conocen en inglés como *spoofing*[59]. Para evitar esto, el sistema puede configurarse para comprobar si las direcciones IP originales están de hecho asociadas con el nombre obtenido. Si no, el nombre será rechazado y se retornará un error. Esta opción se activa poniendo nospoof on.

alert
: Esta opción puede tomar el valor on u off como argumentos. Si se activa, cualquier intento de *spoof* será anotado con un mensaje enviado al sistema de registros **syslog**.

trim
: Esta opción lleva un nombre de dominio como argumento, que se quitará a los nombres antes de buscar su dirección. Es útil para las entradas del fichero `hosts`, que podrán así ir solos los nombres de máquinas, sin el dominio. Cuando se busque una máquina con el nombre de dominio local éste será eliminado, haciendo que la búsqueda en el fichero `/etc/hosts` tenga éxito. El dominio que añada debe terminar en un punto (.) (por ejemplo, `linux.org.au.`).

  Las opciones de trim se van acumulando; podemos considerar nuestra máquina como local de diversos dominios.

Veamos un fichero de ejemplo para vlager en Ejemplo 6−1.

Ejemplo 6−1. Ejemplo de fichero host.conf

```
# /etc/host.conf
# Tenemos servidor de nombres, pero no NIS (de momento)
order   bind hosts
# Permitir direcciones múltiples
multi   on
# Contra los nombres falsos
nospoof on
# Dominio local por defecto (no necesario).
trim    vbrew.com.
```

6.1.1.1. Variables de entorno

Existen algunas variables de entorno que establecen opciones que tienen más prioridad sobre las puestas en el fichero `host.conf`. Éstas son:

RESOLV_HOST_CONF
: Especifica un fichero alternativo a `/etc/host.conf`.

RESOLV_SERV_ORDER
: Establece la opción equivalente a la orden order del fichero anterior. Los servicios pueden ser hosts, bind y/o nis, separados por comas, espacios, puntos o puntos y coma.

RESOLV_SPOOF_CHECK
: Determina la política seguida frente a los nombres falsos. Estará completamente desactivada con la opción off. Con las opciones warn y warn off se realizarán comprobaciones contra los nombres falsos,

pero en el primer caso se mandarán los avisos al registro. Un valor * activa las comprobaciones contra nombres falsos, pero las anotaciones en el registro se dejan como diga el fichero `host.conf`.

RESOLV_MULTI
: El valor on activa la opción ``multi'', y el valor off la desactiva.

RESOLV_OVERRIDE_TRIM_DOMAINS
: Esta variable lleva una lista de dominios por defecto, similar a la puesta en el fichero `host.conf` con la opción trim.

RESOLV_ADD_TRIM_DOMAINS
: Esta variable lleva una lista de dominios por defecto que se *añade* a las que se dan en el fichero `host.conf`.

6.1.2. El fichero nsswitch.conf

La versión 2 de la biblioteca estándar de funciones de GNU incluye un fichero más flexible para sustituir a `host.conf`. El concepto de servicio de nombres se ha extendido para incluir una variedad de diferentes tipos de información. Las opciones para seleccionar las bases de datos a las que interrogar se han introducido todas en un fichero, que se llama `nsswitch.conf`.

El fichero `nsswitch.conf` permite al administrador de sistemas configurar una amplia variedad de diferentes bases de datos. Limitaremos nuestra discusión a opciones que se refieran a la resolución de nombres de máquina y direcciones IP. Se puede encontrar fácilmente mucha más información para aprovechar el resto de las características de este fichero, sin más que leer la documentación de la biblioteca estándar GNU.

Las opciones del fichero `nsswitch.conf` deben estar en líneas diferentes. Los campos se separan por blancos (espacios o tabuladores). Un signo de almohadillado (#) introducirá un comentario para todo el resto de esa línea. Cada línea describe un servicio en particular; la resolución de nombres es uno de ellos. El primer campo de cada línea es el nombre de la base de datos, finalizado en el signo de dos puntos. La base de datos relacionada con la resolución es hosts. Una base de datos relacionada es networks, que se usa para convertir nombres en direcciones de redes. El resto de cada línea lleva opciones que determinan cómo se hacen las búsquedas de los elementos de la base de datos.

Las opciones posibles son:

dns
: Indica que se usa el DNS para resolver la dirección. Esto solo sirve para resolución de nodos, no de redes. Para ello se mira primero el fichero `/etc/resolv.conf`, que veremos después.

files
: Hace la búsqueda en un fichero local. Es decir, en `/etc/hosts` para los nodos, y en `/etc/network` para las redes.

nis o nisplus
: Usará el sistema NIS (sistema de información en red) para resolver nodos o redes. NIS y NIS+ se discuten en detalle en Capítulo 13.

El orden en el que los servicios estén listados es el orden en el que serán interrogados para buscar un nombre. Es decir, los servicios son interrogados leyéndolos de izquierda a derecha, hasta encontrar la respuesta.

Un ejemplo de fichero `nsswitch.conf` lo tenemos en Ejemplo 6–2.

Ejemplo 6−2. Ejemplo de fichero nsswitch.conf

```
# /etc/nsswitch.conf
#
# Ejemplo de configuracion del nsswitch de GNU.
# En el paquete `libc6-doc' se documentan estos ficheros.

hosts:          dns files
networks:       files
```

Este ejemplo hace que el sistema busque los nodos, primero en el DNS y después en `/etc/hosts`, si no se encuentra. En cambio las redes se buscan exclusivamente en `/etc/networks`.

Podemos controlar el comportamiento más precisamente, usando items de acción que describen qué hacer tras el último intento de búsquedas. Los items de acción aparecen entre los servicios, y se encierran entre corchetes, [. La sintaxis general es:

```
[ [!] estado = acción ... ]
```

Hay dos posibles acciones:

return
: Hace que el control retorne al programa que hizo la petición de resolución. Si la búsqueda tuvo éxito, retornará los detalles. Si no, retornará un cero.

continue
: El sistema seguirá buscando a través del siguiente servicio de la lista.

El carácter opcional (!) especifica que el valor de estado debe considerarse invertido antes de comprobarlo, es decir, es un not.

Los valores de estado posible son:

success
: La petición se encontró sin errores. La acción por defecto aquí es return.

notfound
: No hubo error en la búsqueda, pero no se encontró el nodo o la red. La acción predeterminada aquí es continue.

unavail
: El servicio solicitado no está disponible. Por ejemplo, que el fichero `/etc/hosts` no esté en su sitio, o que el servidor DNS o NIS requeridos no respondan. La acción predeterminada es continue.

tryagain
: Significa que el servicio estaba no disponible temporalmente. Por ejemplo, que el fichero `hosts` esté bloqueado por otro proceso, o que el DNS esté muy cargado. La acción predeterminada para este estado es continue.

Un ejemplo de uso de todo esto se muestra en Ejemplo 6−3.

Ejemplo 6−3. Ejemplo de nsswitch.conf con acciones

```
# /etc/nsswitch.conf
```

```
#
# Ejemplo de configuracion del nsswitch de GNU.
# En el paquete `libc6-doc' se documentan estos ficheros.

hosts:          dns [!UNAVAIL=return] files
networks:       files
```

Este ejemplo intentará resolver los nodos usando el DNS. Si se devuelve un error que no sea UNAVAIL, el sistema devolverá lo que ha encontrado. En otro caso intentará buscarlo en `/etc/hosts`. Esto significa que este fichero solo se usará en caso de que el DNS no funcione bien.

6.1.3. Configuración del fichero resolv.conf

Cuando se configura la biblioteca de resolución para utilizar los servicios de BIND, tiene que indicarse también qué servidores utilizar. El fichero `resolv.conf` contiene una lista de servidores, que si está vacía hará considerar al sistema que el servidor está en su máquina.

Si ejecuta un servidor de nombres en su máquina local, tendrá que configurarlo por separado, como se explicará después. Si se encuentra en una red local y puede usar un servidor de nombres existente, mejor. Si estamos conectados a Internet por módem, lo habitual es especificar en `resolv.conf` el servidor de nombres que nos diga nuestro proveedor de servicios.

La opción más importante del fichero `resolv.conf` es nameserver, que tiene la dirección IP del servidor de nombres a usar. Si especifican varios servidores poniendo varias líneas nameserver, se intentarán usar en el orden dado; por lo que debería poner en primer lugar el servidor de nombres más rápido o cercano. Actualmente, puede ponerse un máximo de tres servidores distintos. Si no se pone ninguno, intentará buscar un servidor de nombres en la máquina local.

Otras dos opciones, domain y search, nos permiten usar nombres cortos (sin dominio) para máquinas que estén en nuestro dominio. Normalmente, para conectarnos a una máquina de la misma red, no queremos poner el dominio completo, sino su nombre. Por ejemplo, gauss en lugar de gauss.mathematics.groucho.edu.

Para esto sirve la palabra domain. Nos permite especificar un dominio predeterminado que se añade a las peticiones cuando su búsqueda inicial falla. Por ejemplo, al buscar gauss y fallar el servidor de nombres buscándolo en Internet, le añade automáticamente su dominio predeterminado y ya sí puede resolverlo.

Esto está bien, pensaremos, pero tan pronto como nos refiramos a una máquina que esté fuera del Departamento de Matemáticas, tendremos que volver a teclear el dominio completo. A lo mejor queremos teclear solo quark.physics para referirnos a una máquina del Departamento de Físicas.

Para esto podemos usar la *lista de búsqueda*, que puede especificarse con la opción search. En esta lista se especifica una lista de dominios donde resolver nombres cortos. Los elementos de la lista deben especificarse separándolos por espacios o tabuladores.

Las opciones search y domain son mutuamente excluyentes y no pueden aparecer más de una vez. Si ninguna de las dos se pone, el sistema intentará asignar a los nombres cortos el dominio de la máquina local, que averiguará usando la llamada al sistema `getdomainname(2)`. Si el nodo local no tiene dominio, se asumirá que el dominio predeterminado es el raíz.

Si decidimos poner una opción search en el fichero `resolv.conf`, habrá que ser cuidadosos con los dominios que añadimos a la lista. Las bibliotecas de resolución anteriores a BIND 4.9 solían construir una lista de búsqueda predeterminada para el dominio cuando no se proporciona otra lista. Esta lista predeterminada se hacía con el dominio del nodo, más todos los dominios padre hasta llegar a la raíz. Esto daba lugar a búsquedas innecesarias a los servidores de nombres externos.

Asumamos que estamos en la Cervecera Virtual y queremos conectarnos al sistema foot.groucho.edu. Por un error tecleamos el nombre foo, que no existe. El servidor de la universidad nos responderá que no existe el nodo. Con la búsqueda antigua, intentará buscar ese nombre en los dominios vbrew.com y com. Este último es problemático porque causa una búsqueda innecesaria y además podría existir. Al final nos habremos intentado conectar a una máquina totalmente ajena.

En algunos casos, esto es un potencial problema de seguridad. De hecho las listas de búsqueda deben limitarse a dominios de la organización local o algo similar. La lista en el Departamento de Matemáticas debe limitarse a los dominios maths.groucho.edu y groucho.edu.

Como lo anterior puede resultar confuso, sea el siguiente ejemplo de fichero `resolv.conf` para la Cervecera Virtual:

```
# /etc/resolv.conf
# Nuestro dominio
domain         vbrew.com
#
# Nuestro servidor principal va a ser vlager:
name server     172.16.1.1
```

Cuando se trate de traducir el nombre vale, el sistema empezará por buscar directamente vale y si falla, probará con vale.vbrew.com.

6.1.4. Robustez del sistema de resolución

Si tiene en funcionamiento una red local dentro de otra más grande, deberá usar servidores de nombres principales siempre que sea posible. La ventaja de hacerlo así es que se consiguen generosas memorias *caché*, ya que todas las peticiones de nombres les llega a ellos. Este esquema, sin embargo, tiene un inconveniente: cuando un incendio inutilizó el cable de red dorsal de nuestro departamento en la Universidad, no pudimos trabajar, pues ninguno de los servidores de nombres estaban accesibles. No funcionaban ni los terminales X ni las impresoras...

Aunque no es muy habitual que las redes dorsales de las universidades sean pasto de las llamas, deberían tomarse precauciones para casos como éste.

Una solución es poner un servidor de nombres local que se ocupe de sus nombres locales, y reenvíe todas las peticiones de otros nombres a los servidores principales. Por supuesto, esto sólo es posible si usted tiene un dominio propio.

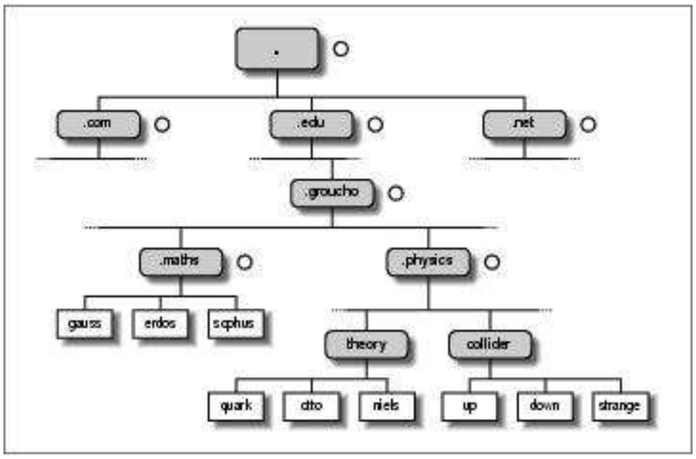
Alternativamente, puede mantener una copia de la tabla de nombres para su dominio o red local en el fichero `/etc/hosts`. En el fichero `/etc/host.conf` deberá incluir la opción order bind,hosts, para obligar a usar el fichero local si el servidor principal de nombres falla. Si usamos `/etc/nsswitch.conf`, habrá que cambiar la línea que haga referencia a *hosts*, dejándola así: hosts: dns files.

6.2. Cómo funciona el DNS

El DNS organiza los nombres de máquina (hostname) en una jerarquía de dominios. Un *dominio* es una colección de nodos relacionados de alguna formaporque estén en la misma red, tal como los nodos de una universidad. Por ejemplo, las universidades americanas se agrupan en el dominio edu. Cada universidad tiene allí un *subdominio*, tal como la universidad Groucho Marx, que posee el subdominio groucho.edu. A su vez, podemos encontrar nuevos subdominios dentro, como el Departamento de Matemáticas (maths). Finalmente, un nodo de ese departamento llamado erdos tendrá un nombre completo (conocido como totalmente cualificado) tal como erdos.maths.groucho.edu. Este nombre totalmente cualificado también se conoce por las siglas FQDN.

En Figura 6−1 vemos una parte del espacio de nombres. La raíz del árbol, que se identifica con un punto sencillo, es lo que se denomina *dominio raíz* y es el origen de todos los dominios. Para indicar que un nombre es FQDN, a veces se termina su escritura en un punto. Este punto significa que el último componente del nombre es el dominio raíz.

Figura 6−1. Una parte del espacio de nombres de dominios



Dependiendiendo de su localización en la jerarquía, un dominio puede ser de primer nivel (top−level), segundo nivel o tercer nivel. Se pueden añadir todos los niveles que queramos, pero no son habituales. Los que siguen son los dominios de primer nivel que veremos con frecuencia:

Dominio	Descripción
edu	Instituciones universitarias, casi todas norteamericanas.
com	Organizaciones comerciales.
org	Organizaciones no comerciales. Las redes privadas UUCP suelen estar en este dominio.
net	Pasarelas y otras redes administrativas.
mil	El ejército norteamericano.
gov	El gobierno norteamericano.
uucp	Dominio para redes UUCP.

Inicialmente los cuatro primeros dominios de la lista anterior pertenecían solo a los Estados Unidos, sin embargo, los cambios de política posteriores han hecho que estos dominios, llamados de dominios globales primer nivel (gTLD) sean realmente globales. Además se están negociando nuevos dominios de primer nivel.[60]

Fuera de los Estados Unidos, cada país suele tener su propio dominio de primer nivel codificado con las dos letras del país definidas en la tabla ISO−3166. Finlandia, por ejemplo, usa el dominio fi; en España se usa el dominio es; en México se usa mx; en Argentina, ar, etc. Por debajo de cada dominio de primer nivel, cada país organiza los dominios a su manera. Algunos crean a segundo nivel una serie de dominios similares a los gTLD. Por ejemplo, en Argentina encontramos los dominios com.ar para las empresas, y org.ar para las organizaciones sin ánimo de lucro. Otros países, como España, ponen directamente como nombres de segundo nivel las instituciones o empresas que los solicitan. Por ejemplo, tenemos hispalinux.es.

Por supuesto, el hecho de que un nombre esté en uno de estos dominios nacionales, no implica que la máquina esté realmente en ese país; significa simplemente que ha sido registrada en el NIC de ese país. Un fabricante sueco puede tener oficinas en Australia y tener sus ordenadores de allá registrados en el dominio se.

La organización del espacio de nombres en una jerarquía de nombres de dominio sirve para resolver fácilmente el problema de la unicidad de los nombres; además muchos nombres completamente cualificados son fáciles de recordar. Bajo esta premisa es conveniente dividir un dominio con gran número de máquinas en subdominios.

El sistema DNS hace más cosas. Permite delegar la *autoridad* de un subdominio a sus administradores. Por ejemplo, los responsables del Centro de Cálculo Groucho pueden crear un subdominio para cada departamento, y delegar su control a éstos. Así, cada departamento puede definir libremente todos los nodos que quiera dentro de su subdominio e incluso crear nuevos subdominios y delegarlos.

Para esto, el espacio de nombres se divide en *zonas*, cada una asignada a un dominio. Hay que ver la diferencia entre *zona* y *dominio*: por ejemplo, el dominio groucho.edu incluye todas las máquinas y subdominios de éste. Mientras que la zona groucho.edu solo incluye las máquinas del dominio, no los subdominios delegados. Es decir, los nodos del subdominio physics.groucho.edu pertenecen a una zona diferente. En Figura 6−1, el inicio de la zona se marca con un pequeño círculo a la derecha del nombre de dominio.

6.2.1. Búsquedas con DNS

Veremos ahora la parte más ingeniosa del DNS. La idea es que si queremos buscar la dirección IP del sistema erdos, DNS pensará, Preguntemos a la gente que lo maneja, y nos lo dirá.

De hecho, el DNS es como una gigantesca base de datos distribuida. Está realizada a través de los llamados servidores de nombres, que proporcionan la información de uno o varios dominios. Para cada zona, debe haber dos o más servidores de nombres capaces de responder por ella. Para obtener la dirección IP de erdos, todo lo que necesitamos es contactar con el servidor de nombres de la zona groucho.edu y obtendremos los datos solicitados.

Pensaremos, es más fácil decirlo que hacerlo pues, ¿cómo llegamos al servidor de nombres de la Universidad Groucho Marx? En el caso de que nuestro ordenador no esté equipado con un oráculo de resolución de direcciones, el DNS nos lo hace también. Cuando nuestra aplicación quiera buscar la información de erdos, contactará con un servidor de nombres local, quien lleva a cabo una secuencia de peticiones. En primer lugar pregunta al servidor de nombres raíz, preguntando por erdos.maths.groucho.edu. El servidor raíz reconoce que el nombre no pertenece a ninguna de sus zonas *de autoridad* pero sí sabe qué hacer con la zona *edu*. Esto es, devuelve a nuestro servidor más información sobre los servidores de nombres que pueden servir la zona edu. Ahora nuestro servidor preguntará por este nombre a uno de esos servidores. Ellos nos enviarán a uno que tenga información autorizada del dominio groucho.edu. Ahora nuestro servidor interrogará a éste y finalmente obtendrá la dirección de erdos.

Aparentemente la búsqueda de una dirección IP supone mucho tráfico, sin embargo es minúsculo si lo comparamos con la consulta de un gigantesco fichero `HOSTS.TXT`. Aun así hay técnicas para mejorar el rendimiento.

Para acelerar futuras peticiones de nombres, el servidor almacena la información obtenida en la búsqueda anterior en su *caché* local. Así, la próxima vez que busquemos algún nodo de groucho.edu, ya no habrá que ir a los servidores raíz o los de la zona edu.[61]

Por supuesto, el servidor de nombres no almacenará para siempre la información en la caché; la limpiará cada cierto tiempo. El tiempo de vida se llama *TTL* (del inglés *time to live*). En cada zona DNS el administrador asigna un valor de TTL.

6.2.2. Tipos de servidores de nombres

Los servidores de nombres que mantienen oficialmente la información de una zona se conocen como *autorizados* de la zona, y a veces se conocen como *servidores principales o maestros*. Cualquier petición de nodos de esa zona irá a parar a uno de estos servidores principales.

Los servidores principales deben estar bien sincronizados. Es decir, uno de ellos será llamado *primario*, que carga su información de un fichero, y hacer a los demás *secundarios*, que obtienen su información pidiéndosela periódicamente al primario.

El objetivo de tener varios servidores principales es distribuir la carga y dar cierta tolerancia a fallos. Cuando uno de los servidores principales falla, todas las peticiones acabarán en los demás. Por supuesto, este esquema no nos protege de fallos del servidor que produzcan errores en todas las peticiones DNS, como podrían ser errores del software.

También podemos instalar un servidor de nombres que no es maestro de ninguna zona.[62] Esto es útil, para dar servicio de nombres a una red local aprovechando sus características de ahorro de ancho de banda gracias a su caché. Estos servidores se conocen como de *sólo−caché*.

6.2.3. La base de datos DNS

Hemos visto que el DNS no sólo sabe de direcciones IP de máquinas, pero también almacena otras informaciones.

Cada unidad de información del DNS se llama *Registro de Recurso* (RR). Cada registro tiene un tipo asociado que describe el dato que contiene, y una clase que especifica el tipo de red al que se aplica. Esto último se adapta a diferentes esquemas de dirección, como direcciones IP (la clase IN), direcciones Hesiod (utilizadas por el sistema Kerberos del MIT) y algunas más. El RR típico es el registro A, que asocia un nombre completamente cualificado con una dirección IP.

Un nodo puede ser conocido por más de un nombre. Por ejemplo, podemos tener un servidor que proporciona tanto servicio FTP como WWW, y tendrá dos nombres: ftp.maquinas.org y www.maquinas.org. Sin embargo, uno de estos nombres debe ser identificado como oficial o *canónico*. La diferencia es que el canónico es el único registro A que debe existir apuntando a esa dirección IP, mientras que el resto de los nombres deben ser alias (registros CNAME), que apuntan al nombre canónico.

No vamos a revisar todos los tipos de RR aquí, pero veremos algún ejemplo más amplio. En Ejemplo 6−4 vemos una parte de la base de datos DNS que está cargada en los servidores de nombres para la zona physics.groucho.edu.

Ejemplo 6−4. Extracto del fichero named.hosts del Departamento de Físicas

```
; Informacion autoritativa physics.groucho.edu.
@  IN  SOA niels.physics.groucho.edu. janet.niels.physics.groucho.edu. {
                  1999090200       ; numero de serie
                  360000           ; refresco
                  3600             ; reintento
                  3600000          ; caducidad
                  3600             ; TTL predeterminado
                }
;
; Servidores de nombres
              IN    NS       niels
              IN    NS       gauss.maths.groucho.edu.
gauss.maths.groucho.edu. IN A 149.76.4.23
```

```
;
; Fisica Teorica (subred 12)
niels          IN    A        149.76.12.1
               IN    A        149.76.1.12
nameserver     IN    CNAME    niels
otto           IN    A        149.76.12.2
quark          IN    A        149.76.12.4
down           IN    A        149.76.12.5
strange        IN    A        149.76.12.6
...
; Laboratorio (subred 14)
boson          IN    A        149.76.14.1
muon           IN    A        149.76.14.7
bogon          IN    A        149.76.14.12
...
```

Aparte de los registros A y CNAME, vemos al principio un registro especial, de varias líneas. Es el registro SOA, que señaliza el *inicio de autoridad*, que almacena diversos parámetros de la zona de la que es autoritativo el servidor. El registro SOA incluye, por ejemplo, el tiempo de vida predeterminado de los registros (TTL).

Nótese que todos los nombres del fichero de ejemplo que no finalizan en un punto deben interpretarse relativos al dominio physics.groucho.edu. El nombre especial (@) utilizado en el registro `SOA` representa al propio nombre del dominio.

Hemos visto antes que los servidores de nombres para el dominio groucho.edu tienen que saber acerca de la zona physics para poder realizar peticiones a sus servidores de nombres. Esto normalmente se realiza mediante dos registros: los registros DNS que proporcionan el FQDN del servidor de nombres, y el registro A que asocia ese FQDN con una dirección IP. Puesto que estos registros son los que mantienen el espacio de nombres, se conocen frecuentemente como *registros glue*. Sólo son instancias de registros para los que una zona padre mantiene información sobre nodos de la zona subordinada. Los registros glue apuntando a los servidores de nombres de physics.groucho.edu se muestran en Ejemplo 6−5.

Ejemplo 6−5. Un extracto del fichero named.hosts de GMU

```
;  Zona de datos para la zona groucho.edu
@  IN  SOA vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
                     1999070100       ; serie no
                     360000           ; refresco
                     3600             ; reintento
                     3600000          ; expiración
                     3600             ; ttl por omisión
              }
....
;
; Registros glue para la zona  physics.groucho.edu
physics         IN     NS        niels.physics.groucho.edu.
                IN     NS        gauss.maths.groucho.edu.
niels.physics   IN     A         149.76.12.1
gauss.maths     IN     A         149.76.4.23
...
```

6.2.4. Resolución inversa

La operación más habitual con el DNS es obtener la dirección IP correspondiente a un nombre de nodo. Sin embargo, a veces queremos hacer la operación opuesta: encontrar el nombre a partir de la dirección IP. Esto se conoce como *resolución inversa*, y la usan diversas aplicaciones para comprobación de identidad del cliente. Cuando se utiliza el fichero `hosts`, la resolución se realiza mediante una búsqueda simple en el fichero. Con el DNS, una búsqueda exhaustiva en el espacio de nombres carece de sentido. En su lugar, existe un dominio especial, el in−addr.arpa, que contiene las direcciones IP de todos los sistemas en una notación de puntos invertida. Por ejemplo, a la dirección 1.2.3.4 le corresponde el nombre 4.3.2.1.in−addr.arpa. El registro de recurso (RR) que define esto se llama registro PTR.

Cuando se crea una zona de autoridad, ello suele significar que sus administradores tienen control total sobre cómo se asignan los nombres a las direcciones. Puesto que normalmente tienen bajo su control una o más redes o subredes IP, se da una situación de mapeo uno−a−varios entre zonas DNS y redes IP. El Departamento de Física, por ejemplo, comprende las subredes 149.76.8.0, 149.76.12.0 y 149.76.14.0.

En consecuencia, deben crearse nuevas zonas en el dominio in−addr.arpa para la zona de Física, delegándose a ésta las siguientes: 8.76.149.in−addr.arpa, 12.76.149.in−addr.arpa, y 14.76.149.in−addr.arpa. De otro modo, cada vez que instalásemos un nuevo nodo en el laboratorio Collider, habría que contactar con el que gestiona la red padre para que actualizase su fichero de zona in−addr.arpa.

En Ejemplo 6−6 se muestra la base de datos para la subred 12. Los registros glue correspondientes a la base de datos de la zona padre se muestran en Ejemplo 6−7.

Ejemplo 6−6. Extracto del fichero named.rev de la subred 12

```
; dominio 12.76.149.in-addr.arpa
@  IN  SOA  niels.physics.groucho.edu. janet.niels.physics.groucho.edu. {
                     1999090200 360000 3600 3600000 3600
           }
2        IN     PTR       otto.physics.groucho.edu.
4        IN     PTR       quark.physics.groucho.edu.
5        IN     PTR       down.physics.groucho.edu.
6        IN     PTR       strange.physics.groucho.edu.
```

Ejemplo 6−7. Extracto del fichero named.rev de la Red 149.76

```
; dominio 76.149.in-addr.arpa
@  IN  SOA vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
                     1999070100 360000 3600 3600000 3600
                 }
...
; subnet 4: Mathematics Dept.
1.4        IN     PTR      sophus.maths.groucho.edu.
17.4       IN     PTR      erdos.maths.groucho.edu.
23.4       IN     PTR      gauss.maths.groucho.edu.
...
; subnet 12: Physics Dept, separate zone
12         IN     NS       niels.physics.groucho.edu.
           IN     NS       gauss.maths.groucho.edu.
niels.physics.groucho.edu.  IN  A 149.76.12.1
gauss.maths.groucho.edu.  IN  A   149.76.4.23
...
```

Las zonas de in−addr.arpa sólo pueden ser creadas por superconjuntos de redes IP. Hay una restricción más severa: las máscaras de estas redes deben contener los octetos completos. Es decir, podemos crear una zona para una red con máscara 255.255.255.0 pero no para una del tipo 255.255.255.128. El motivo es que para especificar la red delegada 149.76.4.0 tenemos el dominio 4.76.149.in−addr.arpa, pero para la red 149.76.4.128 no tenemos forma de nombrar el dominio in−addr correspondiente.

6.3. Ejecución de named

named (pronúnciese *n'eim−di:*) es el servidor DNS en casi todas las máquinas Unix. Es un programa desarrollado originalmente para BSD. La versión 4 se ha usado mucho tiempo y venía con cualquier distribución GNU/Linux. La nueva edición, la versión 8, se ha introducido después y supone grandes cambios desde la anterior.[63] Tiene muchas características nuevas, como el soporte de actualización dinámica del DNS, notificaciones de cambios, mejoras importantes de rendimiento y una nueva sintaxis de fichero de configuración. Para más detalle debemos comprobar la documentación que viene con el código fuente.

Esta sección requiere ideas acerca de cómo funciona el Sistema de Nombres y Dominios (DNS). Si lo que sigue a continuación le suena a chino, puede releer el capítulo Sección 6.2, que le dará información acerca de cómo funciona básicamente el DNS.

named suele iniciarse al arrancar la máquina, y ejecutarse hasta que se apaga. Las versiones anteriores de BIND hasta la 8 obtienen la información que necesitan de un fichero llamado `/etc/named.boot`. Las nuevas versiones usan el fichero `/etc/named.conf`. Además, hay que configurar los ficheros de zona.

Para ejecutar **named**, sólo tiene que teclear:

```
# /usr/sbin/named
```

El programa **named** se iniciará y leerá el fichero **named.boot** y los ficheros de zona que se especifiquen en él. Su número de proceso será anotado en ASCII en el fichero `/var/run/named.pid`, recibirá ficheros de zona de los servidores principales si es necesario y comenzará a escuchar las peticiones de DNS por el puerto 53.

6.3.1. El fichero named.boot

El fichero de configuración para los BIND anteriores a la 8 tenían una estructura muy simple. En la versión 8 el fichero se llama `/etc/named.conf` y es totalmente distinto. Nos pararemos en ambas versiones y veremos las diferencias, y cómo convertir del formato antiguo al nuevo.

El fichero **named.boot** suele ser muy pequeño y contiene punteros a ficheros con información de zonas y a otros servidores de nombres. Los comentarios en este fichero comienzan con un punto y coma y se extienden hasta el siguiente fin de línea. Antes de que veamos con más detalle el formato de este fichero, observaremos el ejemplo para la máquina vlager dado en Ejemplo 6−8.

Ejemplo 6−8. Fichero named.boot para vlager

```
;
; Fichero /etc/named.boot para vlager.vbrew.com
;
directory     /var/named
;
;             domain                    file
;-----------------
caché         .                         named.ca
primary       vbrew.com                 named.hosts
primary       0.0.127.in-addr.arpa      named.local
primary       16.172.in-addr.arpa       named.rev
```

Veamos cómo es el fichero. La palabra directory indica a **named** el directorio donde están los demás ficheros de configuración (los ficheros de zona).

Los comandos caché y primary sirven para cargar información en \prog{named}. Esta información se obtiene de los ficheros espeficados en el segundo argumento. Contienen representaciones textuales de los registros DNS, que veremos a continuación.

En este ejemplo, se configura **named** como el servidor de nombres principal para tres dominios: los que se indican con la orden primary. La primera línea dice que **named** actúe como servidor principal para vbrew.com, tomando la información de zona del fichero **named.hosts**.

La entrada iniciada con la palabra caché es muy especial y debe estar presente en casi todas las máquinas que ejecuten un servidor de nombres. Su función es doble: indica a **named** que active su *caché*, y también que cargue la información de los servidores raíz del fichero indicado (en este caso, `named.ca`). Regresaremos a este concepto más tarde.

A continuación se presenta una lista de las opciones más importantes que podemos poner en el fichero `named.boot`:

directory
: Especifica un directorio donde estén los ficheros de zona. Pueden ponerse varios directorios repitiendo el comando directory. De acuerdo con el estándar de sistema de ficheros para Linux, el directorio debería ser `/var/named`.

primary
: Los argumentos que lleva son un nombre de dominio y un nombre de fichero, declarando el servidor local primario para el dominio de named. Como servidor primario, **named** carga la información de zona del fichero dado.

  Normalmente, siempre habrá por lo menos una orden primary en cada fichero `named.boot`, para traducción inversa del IP `127.0.0.1`, que es el interfaz de bucle o «loopback», como ya sabemos.

secondary
: Esta sentencia tiene como parámetros un nombre de dominio, una lista de direcciones y un nombre de fichero. Declara el servidor local como servidor maestro secundario para el dominio indicado.

  Un servidor secundario mantiene también información «autorizada» como el primario, pero en lugar de obtenerla de un fichero, la intenta obtener de un servidor primario. Debe proporcionarse al menos una dirección IP de servidor primario en la lista de direcciones. El servidor local irá contactando con cada uno de ellos hasta que transfiera con éxito la base datos de zona, que será almacenada en el fichero de respaldo −copia de seguridad o backup− dado en el tercer argumento del comando. Si ninguno de los servidores primarios responde, se obtendrá la información de zona del fichero de respaldo.

  named intentará entonces refrescar los datos almacenados regularmente. Esto se explica después cuando se vean las entradas SOA de los ficheros.

caché
: Tiene como argumentos un dominio y un nombre de fichero. Contiene la lista de servidores de nombres raíz. Sólo se reconocerán registros NS y A. El argumento domain es normalmente el nombre del dominio raíz (.).

  Esta información es fundamental: si la orden caché no existiera, **named** no haría una *caché* local. Esto degradaría de forma importante el rendimiento e incrementaría la carga de la red si los nombres que se buscan no están en la red local. Además, **named** tampoco será capaz de contactar con cualquier servidor de nombres raíz, y por ello, no podrá resolver ninguna dirección excepto aquellas para las que esté autorizado. Una excepción a esta regla, ocurre cuando se usan servidores redirigidos (con la opción forwarders explicada a continuación).

forwarders
: Esta opción lleva una lista de direcciones como argumento. Las direcciones IP en la lista especifican servidores de nombres a los que **named** puede preguntar si falla una traducción de un nombre mediante su *caché* local. Se intenta preguntar a todos en orden hasta que uno de ellos responda.

slave
: Esta opción hace que el servidor sea *esclavo*. Esto significa que nunca realizará consultas recursivas, sino que las redirigirá a los servidores especificados con forwarders.

Hay dos opciones adicionales que no vamos a describir: sortlist y domain. Además, hay dos directivas que pueden aparecer en los ficheros de zona. Son INCLUDE y ORIGIN, que tampoco vamos a describir, ya que raramente se utilizan.

6.3.2. El fichero named.conf de BIND 8

En la versión 8 de BIND se han incluido nuevas características, lo cual ha requerido una nueva sintaxis del fichero de configuración principal. El fichero `named.boot` ha sido reemplazado por otro, de nombre `named.conf`, que tiene una sintaxis similar a la del programa **gated** y recuerda a la del lenguaje C.

La nueva sintaxis es más compleja, pero por suerte disponemos de una utilidad para convertir automáticamente los ficheros `named.boot` de sintaxis antigua. Esta utilidad es un script de PERL llamado **named−bootconf.pl**, que encontraremos en el código fuente de BIND 8; y lee un fichero en sintaxis antigua, devolviendo por su salida estándar el fichero en sintaxis nueva. Naturalmente, para utilizarlo es necesario tener correctamente instalado el intérprete de lenguaje PERL.

Al script lo invocaremos, por ejemplo, así:

```
# cd /etc
# named-bootconf.pl <named.boot >named.conf
```

El script produce entonces un fichero similar al que se muestra en Ejemplo 6−9, donde hemos eliminado algunos comentarios que produce adicionalmente el script.

Ejemplo 6−9. Fichero named.conf para usar BIND 8 con vlager

```
//
// /etc/named.boot para vlager.vbrew.com
options {
    directory "/var/named";
};

zone "." {
    type hint;
    file "named.ca";
};

zone "vbrew.com" {
    type master;
    file "named.hosts";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "named.local";
};

zone "16.172.in-addr.arpa" {
    type master;
    file "named.rev";
};
```

Si observamos el ejemplo, veremos que cada línea de `named.boot` ha sido convertida a un bloque en estilo C, encerrado entre llaves (signos { y }).

Los comentarios del fichero se escriben ahora en notación similar a C++, es decir, dos barras (signo //).

La sentencia directory va ahora dentro del bloque options, junto a otras posibles opciones globales de configuración.

Las sentencias caché y primary se convierten en bloques de zona, con sentencias type específicas, de valor hint y master, respectivamente.

Los ficheros de zona no necesitan modificarse, ya que su sintaxis sigue siendo la de antes.

La nueva sintaxis de configuración se ha pensado para poder incluir muchas más opciones de configuración, en las que no vamos a detenernos. Si deseamos conocerlas, la mejor fuente de información es la que viene con el paquete de fuentes de BIND 8.

6.3.3. Ficheros de base de datos DNS

Los ficheros incluidos con **named**, como **named.hosts**, siempre tienen un dominio asociado a ellos llamado *origen*. Este es el nombre de dominio especificado con los comandos caché y primary. En un fichero maestro, se pueden especificar nombres de máquinas y dominios relativos a este dominio. Un nombre dado en un fichero de configuración se considera *absoluto* si termina con un punto. En caso contrario se considera relativo al origen. Al origen en sí mismo nos podemos referir con «@».

Todos los datos en un fichero principal se dividen en *registros de recursos* o RRs. Son la unidad de información del DNS. Cada RR tiene un tipo. Los registros de tipo A, por ejemplo, asocian un nombre a una dirección IP. Los registros de tipo CNAME asocian un alias de una máquina con su nombre oficial. Como ejemplo, obsérvese Ejemplo 6−11, que muestra el fichero `named.hosts` para nuestro sistema.

La representación de los RRs en los ficheros utiliza el siguiente formato:

```
    [domain] [ttl] [class] type rdata
```

Los campos se separan por espacios o tabulaciones. Una entrada puede continuarse en varias líneas si se abre un paréntesis antes del primer fin de línea y el último campo es seguido de un cierre de paréntesis. Cualquier cosa entre un punto y coma y el siguiente salto de línea será un comentario.

domain
: Aquí va el nombre del dominio que se aplica al RR actual. Si no se da nombre de dominio, se asume el mismo que se puso para el RR anterior.

ttl
: Con el fin de forzar al sistema DNS a descartar información después de cierto tiempo, cada RR lleva asociado un *tiempo de vida* o *ttl*[64]. El campo `ttl` especifica, en segundos, el tiempo de validez de la información desde que se obtiene del servidor. Es un número decimal de hasta ocho dígitos.

  Si no se especifica ningún valor, tomará uno por defecto del campo *minimum* del registro SOA precedente.

class
: Aquí se indica la clase de dirección: IN para direcciones IP, HS para objetos de la clase Hesiod. Trabajando con redes TCP/IP debe usarse siempre la clase IN.

  Si no se especifica ningún valor, se toma el valor del RR anterior.

type
: Describe el tipo de RR. Los tipos habituales son A, SOA, PTR y NS. En las siguientes secciones comentaremos estos tipos de RRs.

`rdata`
: Contiene los datos asociados al RR. El formato depende del tipo, y se describirán más adelante.

A continuación se presenta una lista incompleta de RRs que se utilizan en los ficheros de DNS. Hay algunos más que no vamos a comentar. Son experimentales, y de escaso uso.

SOA
: Describe una zona de autoridad (SOA significa «Start of Authority», es decir, «Comienzo de Autoridad»). Señala que los registros siguientes contienen información «autorizada» para el dominio. Cada fichero incluido en la opción primary debe tener un registro SOA para esta zona. Los datos asociados contienen los siguientes campos:

  `origin`
  : Nombre canónico del servidor de nombres primario para este dominio. Se suele dar como nombre absoluto.

  `contact`
  : Dirección de correo electrónico de la persona responsable de mantener el dominio, reemplazando el carácter «@» por un punto. Por ejemplo, si el responsable de nuestra red fuese janet, este campo contendrá: janet.vbrew.com.

  `serial`
  : Este es el número de versión del fichero de zona, expresado con un número decimal. Cuando se cambien datos del fichero, deberá incrementarse este número. Se suele expresar como número de versión en el día actual, es decir, en el formato AAAAMMDDnn siendo AAAA el año, MM el mes, DD el día y nn el número de revisión de ese día (01 si no hay más de una). Por ejemplo, 2001072201 para el 22 de julio de 2001.

    El número de versión es utilizado por los servidores secundarios para saber cuándo la información de una zona ha cambiado. Para mantenerse actualizados, los servidores secundarios piden cada cierto tiempo el registro SOA del primario, y comparan el número de versión con el que tienen en la *caché*. Si ha cambiado, el servidor secundario pedirá de nuevo la información de zona al primario.

  `refresh`
  : Especifica el intervalo, en segundos, que esperan los servidores secundarios entre peticiones de registros SOA a los primarios. De nuevo, se trata de un número decimal de hasta ocho dígitos.

    Normalmente, la topología de la red no cambia mucho, con lo que este número será como poco de un día para grandes redes, y de mucho más tiempo para redes pequeñas.

  `retry`
  : Este número determina los intervalos de tiempo entre reintentos de comunicación con servidores primarios cuando una petición de una zona falla. No debe ser pequeño ya que un fallo temporal del servidor primario hará que el secundario cargue inútilmente la red. Buenas elecciones son una hora o como poco media hora.

  `expire`
  : Especifica el tiempo, en segundos, que tardará el servidor en descartar los datos de zona si no ha podido contactar con el servidor primario. Normalmente se pondrá un valor grande, de por lo menos una semana (604800 segundos), aunque si se incrementa a un mes o más será incluso más razonable.

`minimum`
: Valor predeterminado para el valor del *ttl* en los registros de recursos que no lo especifiquen. Sirve para indicar a otros servidores de nombres que descarten el RR tras cierto tiempo. No tiene efecto, sin embargo, sobre el tiempo en el que un servidor secundario intenta actualizar la información de zona.

  El valor de `minimum` debe ser grande, en especial para redes locales con topologías poco cambiantes. Una buena elección puede ser de una semana o un mes. En el caso de que haya registros RR que cambien con frecuencia, siempre podrá asignarle valores particulares de `ttl`.

A
: Asocia direcciones IP con nombres. El campo de datos contiene la dirección separando los octetos por puntos, como es habitual.

  Para cada máquina sólo puede haber un registro A, que se considera nombre oficial o *canónico*. Cualquier otro nombre será un alias y debe ser incluido con registros CNAME.

NS
: Apunta a un servidor de nombres maestro de una zona subordinada. El campo de datos contiene el nombre del servidor. Para traducir ese nombre debe proporcionarse un registro A adicional, que se conoce como *glue*, al proporcionar la dirección IP del servidor.

  Hay que incluir registros NS en dos casos: primero, cuando delegamos la autoridad a una zona subordinada. Segundo, en la base de datos del servidor principal de cualquier zona. Los servidores NS especificados en el fichero de zona deben coincidir exactamente con los que especifica la zona padre que delega.

  El registro NS especifica el nombre del servidor de nombres primario y los secundarios para una zona. Estos nombres deben poderse resolver para poderse usar. A veces los servidores pertenecen al mismo dominio que sirven, lo que ocasiona un problema de *el huevo o la gallina*: no podemos obtener el servidor de nombres hasta que accedamos al dominio, pero para acceder el dominio hay que conocer la IP del servidor de nombres... Para resolver este problema se crean registros A directamente en la zona padre, para resolver esas direcciones. Estos son los que ya comentamos antes, los *registros glue* (podríamos traducirlos como registros−pegamento), puesto que unen o *pegan* la zona hija a la zona padre.

CNAME
: Asocia un alias con su *nombre canónico*. El nombre canónico se determina con un registro A. Los alias son indicados mediante registros CNAME.

PTR
: Se usa para asociar nombres del dominio in−addr.arpa con sus nombres normales. Se usa para obtener nombres a partir de direcciones IP (traducción inversa). El nombre de la máquina debe ser el canónico.

MX
: Especifica el *servidor de correo* para un dominio. En la sección Sección 17.4.1 se explica por qué son necesarios estos servidores. La sintaxis del registro MX es:

```
    [domain] [ttl] [class] MX preference host
```

  `host` nombra el servidor de correo para el dominio `domain`. Cada servidor tiene asociado un valor de `preference` (preferencia). Cuando un agente de transferencia de mensajes quiere entregar correo al dominio, intentará conectarse a esos servidores hasta conseguir entregar el mensaje; empezando por el que tenga menor valor de preferencia.

HINFO

Este registro da información sobre el hardware y el software de la máquina. Su sintaxis es:

```
[domain] [ttl] [class] HINFO hardware software
```

El campo `hardware` identifica el hardware usado en este nodo. Para indicarlo, se siguen ciertas convenciones, especificadas en el RFC 1700. Si el campo contiene blancos, debe encerrarse entre comillas dobles. El campo `software` indica el sistema operativo que ejecuta el nodo, que también está normalizado.

Por ejemplo, un registro `HINFO` para describir un sistema Intel ejecutando Linux podría ser:

```
tao  36500  IN  HINFO  IBM−PC  LINUX2.2
```

y para el caso de que se tratara de un sistema basado en Motorola 68000:

```
cevad 36500 IN  HINFO  ATARI−104ST LINUX2.0
jedd  36500 IN  HINFO  AMIGA−3000  LINUX2.0
```

6.3.4. Configuración de named sólo para caché

Hay una clase especial de configuración de `named`, que nos servirá para introducirnos en su funcionamiento. Se llama *sólo−caché*. No sirve ningún dominio propio, pero actúa como repetidor de otros DNS para nuestra red local. Cuando se repitan peticiones a un mismo nodo, el servidor responderá con la información que ya tiene, evitando peticiones repetidas que ocupen ancho de banda en Internet. Esto es especialmente útil cuando contamos con una conexión de banda estrecha, como las que veremos en Capítulo 7 y Capítulo 8.

El fichero `named.boot` para un servidor sólo de caché, es similar a éste:

```
; named.boot para servidor de sólo caché
directory                            /var/named
primary       0.0.127.in−addr.arpa   named.local ; red local
caché         .                      named.ca    ; servidores raiz
```

Además de este fichero, hay que tener el correspondiente `named.ca`, con una lista válida de servidores raíz. Debemos copiar y usar Ejemplo 6−10 para esto. No se requieren otros ficheros para una configuración de sólo caché.

6.3.5. Cómo hacer los ficheros maestros

Ejemplo 6−10, Ejemplo 6−11, Ejemplo 6−12, y Ejemplo 6−13 muestran ficheros de ejemplo para un servidor de nombres de la Cervecera Virtual, localizada en vlager. Debido a la naturaleza de la red propuesta (una simple LAN), el ejemplo es muy simple también.

El fichero de caché `named.ca` mostrado en Ejemplo 6−10 contiene ejemplos de registros de servidores raíz. Un fichero típico de caché contiene como una docena de servidores de esta clase. Se puede obtener una lista de los servidores raíz usando la utilidad The `named.ca` caché file shown in **nslookup** mostrada en la siguiente sección. [65]

Ejemplo 6−10. El fichero named.ca

```
;
; /var/named/named.ca          Fichero de caché para la cervecera.
```

```
;                Al no estar en Internet no necesitamos servidores
;                raiz. Si no fuera asi, descomentense.
;
;.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
;A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;.                        3600000      NS    B.ROOT-SERVERS.NET.
;B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;.                        3600000      NS    C.ROOT-SERVERS.NET.
;C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;.                        3600000      NS    D.ROOT-SERVERS.NET.
;D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
;.                        3600000      NS    E.ROOT-SERVERS.NET.
;E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
;.                        3600000      NS    F.ROOT-SERVERS.NET.
;F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
;.                        3600000      NS    G.ROOT-SERVERS.NET.
;G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
;.                        3600000      NS    H.ROOT-SERVERS.NET.
;H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
;.                        3600000      NS    I.ROOT-SERVERS.NET.
;I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
;.                        3600000      NS    J.ROOT-SERVERS.NET.
;J.ROOT-SERVERS.NET.      3600000      A     198.41.0.10
;.                        3600000      NS    K.ROOT-SERVERS.NET.
;K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
;.                        3600000      NS    L.ROOT-SERVERS.NET.
;L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
;.                        3600000      NS    M.ROOT-SERVERS.NET.
;M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
;
```

Ejemplo 6−11. El Fichero named.hosts

```
;
; /var/named/named.hosts       Nodos de la cervecera
;                               El origen es vbrew.com
;
@                IN  SOA   vlager.vbrew.com. janet.vbrew.com. (
                           2000012601 ; serie
                           86400      ; refresco: uno al dia
                           3600       ; reintento: una hora
                           3600000    ; caducidad:  42 dias
                           604800     ; minimo: 1 semana
                           )
                 IN  NS    vlager.vbrew.com.
;
; Correo local se entrega a vlager
                 IN  MX    10 vlager
;
; direccion `loopback'
localhost.       IN  A     127.0.0.1
;
; La ethernet de la cervecera virtual
vlager           IN  A     172.16.1.1
vlager-if1       IN  CNAME vlager
; vlager es tambien servidor de noticias
news             IN  CNAME vlager
vstout           IN  A     172.16.1.2
vale             IN  A     172.16.1.3
;
; Ethernet de la vinatera virtual
vlager-if2       IN  A     172.16.2.1
vbardolino       IN  A     172.16.2.2
vchianti         IN  A     172.16.2.3
vbeaujolais      IN  A     172.16.2.4
```

```
;
; Ethernet (subsidiaria) de los Espiritus Virtuales
vbourbon         IN  A     172.16.3.1
vbourbon-if1     IN  CNAME vbourbon
```

Ejemplo 6−12. Fichero named.local

```
;
; /var/named/named.local      Resolucion inversa de 127.0.0
;                             El origen es 0.0.127.in-addr.arpa.
;
@             IN  SOA   vlager.vbrew.com. joe.vbrew.com. (
                        1          ; serie
                        360000     ; refresco: 100 horas
                        3600       ; reintento:   una hora
                        3600000    ; caducidad:  42 dias
                        360000     ; minimo: 100 horas
                        )
              IN  NS    vlager.vbrew.com.
1             IN  PTR   localhost.
```

Ejemplo 6−13. Fichero named.rev

```
;
; /var/named/named.rev         Resolucion inversa de nuestras IPs
;                               El origen es 16.172.in-addr.arpa.
;
@             IN  SOA   vlager.vbrew.com. joe.vbrew.com. (
                          16         ; serie
                          86400      ; refresco: una vez diaria
                          3600       ; reintento: una hora
                          3600000    ; caducidad:  42 dias
                          604800     ; minimo: 1 semana
                          )
              IN  NS    vlager.vbrew.com.
; cervecera
1.1           IN  PTR   vlager.vbrew.com.
2.1           IN  PTR   vstout.vbrew.com.
3.1           IN  PTR   vale.vbrew.com.
; vinatera
1.2           IN  PTR   vlager-if2.vbrew.com.
2.2           IN  PTR   vbardolino.vbrew.com.
3.2           IN  PTR   vchianti.vbrew.com.
4.2           IN  PTR   vbeaujolais.vbrew.com.
```

6.3.6. Cómo verificar la configuración

nslookup es una estupenda utilidad para comprobar el funcionamiento de un servidor de nombres. Se puede usar interactivamente o pasándole la pregunta por la línea de órdenes. En este último caso podemos invocar la orden así:

```
$ nslookup
nombre-de-host
```

nslookup envía sus peticiones al servidor citado en `resolv.conf`. Si este fichero tiene más de un servidor, **nslookup** eligirá uno al azar.

El modo interactivo es mucho más interesante. No sólo sirve para buscar la IP de un nodo, sino que también podemos interrogar acerca de cualquier tipo de registro DNS y transferirnos toda la información de una zona si queremos.

Si se invoca sin argumentos, **nslookup** muestra el nombre del servidor elegido y entra en modo interactivo. En el prompt **>** podemos escribir cualquier nombre de dominio. Al principio preguntará sólo por registros A, es decir, obtención de la IP asociada.

Podemos elegir un tipo de registro diferente con la orden:

```
set type=tipo
```

donde *`tipo`* es uno de los tipos de RR descritos antes, o ANY.

Veamos una posible sesión de **nslookup**:

```
$ nslookup
Default Server:  tao.linux.org.au
Address:  203.41.101.121

> metalab.unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

Name:    metalab.unc.edu
Address:  152.2.254.81

>
```

La salida muestra el servidor DNS interrogado y el resultado obtenido.

Si preguntamos por algo que no tiene IP asociada pero sí otros registros de otra clase, el programa nos devolverá una advertencia del tipo `No type A records found`. Sin embargo, podemos usar el citado comando **set type** para buscar registros de otras clases. Por ejemplo, el registro SOA de un dominio puede ser pedido así:

```
> unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

*** No address (A) records available for unc.edu
> set type=SOA
> unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

unc.edu
        origin = ns.unc.edu
        mail addr = host-reg.ns.unc.edu
        serial = 1998111011
        refresh = 14400 (4H)
        retry   = 3600 (1H)
        expire  = 1209600 (2W)
        minimum ttl = 86400 (1D)
unc.edu name server = ns2.unc.edu
unc.edu name server = ncnoc.ncren.net
unc.edu name server = ns.unc.edu
ns2.unc.edu     internet address = 152.2.253.100
ncnoc.ncren.net internet address = 192.101.21.1
ncnoc.ncren.net internet address = 128.109.193.1
ns.unc.edu      internet address = 152.2.21.1
```

De manera parecida, para preguntar por registros MX haremos:

```
> set type=MX
> unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121
```

```
unc.edu preference = 0, mail exchanger = conga.oit.unc.edu
unc.edu preference = 10, mail exchanger = imsety.oit.unc.edu
unc.edu name server = ns.unc.edu
unc.edu name server = ns2.unc.edu
unc.edu name server = ncnoc.ncren.net
conga.oit.unc.edu       internet address = 152.2.22.21
imsety.oit.unc.edu      internet address = 152.2.21.99
ns.unc.edu      internet address = 152.2.21.1
ns2.unc.edu     internet address = 152.2.253.100
ncnoc.ncren.net internet address = 192.101.21.1
ncnoc.ncren.net internet address = 128.109.193.1
```

Con el tipo ANY obtendremos todos los registros existentes asociados al nombre dado.

Una aplicación práctica de **nslookup**, para depurar un servidor, es obtener la lista de servidores raíz. Para ello no hay más que pedir los NS del registro raíz (.):

```
> set type=NS
> .
Server:  tao.linux.org.au
Address:  203.41.101.121

Non-authoritative answer:
(root)  name server = A.ROOT-SERVERS.NET
(root)  name server = H.ROOT-SERVERS.NET
(root)  name server = B.ROOT-SERVERS.NET
(root)  name server = C.ROOT-SERVERS.NET
(root)  name server = D.ROOT-SERVERS.NET
(root)  name server = E.ROOT-SERVERS.NET
(root)  name server = I.ROOT-SERVERS.NET
(root)  name server = F.ROOT-SERVERS.NET
(root)  name server = G.ROOT-SERVERS.NET
(root)  name server = J.ROOT-SERVERS.NET
(root)  name server = K.ROOT-SERVERS.NET
(root)  name server = L.ROOT-SERVERS.NET
(root)  name server = M.ROOT-SERVERS.NET

Authoritative answers can be found from:
A.ROOT-SERVERS.NET      internet address = 198.41.0.4
H.ROOT-SERVERS.NET      internet address = 128.63.2.53
B.ROOT-SERVERS.NET      internet address = 128.9.0.107
C.ROOT-SERVERS.NET      internet address = 192.33.4.12
D.ROOT-SERVERS.NET      internet address = 128.8.10.90
E.ROOT-SERVERS.NET      internet address = 192.203.230.10
I.ROOT-SERVERS.NET      internet address = 192.36.148.17
F.ROOT-SERVERS.NET      internet address = 192.5.5.241
G.ROOT-SERVERS.NET      internet address = 192.112.36.4
J.ROOT-SERVERS.NET      internet address = 198.41.0.10
K.ROOT-SERVERS.NET      internet address = 193.0.14.129
L.ROOT-SERVERS.NET      internet address = 198.32.64.12
M.ROOT-SERVERS.NET      internet address = 202.12.27.33
```

Para ver el conjunto completo de comandos, podemos usar **help** dentro de **nslookup**.

6.3.7. Otras Utilidades Interesantes

Hay algunas utilidades que pueden ayudarnos en nuestro trabajo como administradores de BIND. Vamos va ver algunas de ellas. Para cualquier detalle adicional se recomienda consultar la documentación específica de dichas utilidades.

hostcvt le ayuda con la configuración inicial de BIND convirtiendo su fichero `/etc/hosts` en ficheros maestros para **named**. Genera tanto las entradas del mapeado directo (A) como el mapeado inverso (PTR), y toma en cuenta los alias. Por supuesto, no hará todo el trabajo por usted, así que todavía tendrá que ajustar los palores de temporización el el registro SOA, por ejemplo, a añadir registros MX. También, le puede ayudar a ahorrarse algunas aspirinas. **hostcvt** es parte de las fuentes BIND, pero puede encontrase como un paquete individual en unos pocos servidores FTP.

Tras la puesta en marcha del servidor, normalmente habrá que probarla. Algunas utilidades facilitan la vida para esto. Una de ellas es **dnswalk**, que está basada en Perl. La segunda es **nslint**. Ambas recorren la base de datos DNS buscando errores habituales y verificando que la información que encuentran es consistente. Otras dos utilidades interesantes son **host** y **dig**, que vienen con el paquete BIND y pueden usarse para una inspección manual del las bases de datos.

Estas utilidades suelen venir empaquetadas. **dnswalk** y **nslint** están disponibles en fuentes en http://www.visi.com/~barr/dnswalk/ y ftp://ftp.ee.lbl.gov/nslint.tar.Z. En cuanto a **host** y **dig**, el código fuente se encuentra en ftp://ftp.nikhef.nl/pub/network/ y ftp://ftp.is.co.za/networking/ip/dns/dig/.

Capítulo 7. SLIP: IP por línea serie

Los servidores y terminales que manejan protocolos tales como IP o SLIP, en los cuales los datos están empaquetados, necesitan saber dónde empieza y dónde termina cada uno de esos paquetes en la cadena de datos. El mecanismo para marcar y detectar el comienzo y el fin de cada paquete, se denomina delimitación *(delimitation)*. El protocolo Ethernet utiliza este mecanismo en entornos de redes de área local, y los protocolos SLIP y PPP lo utilizan en comunicaciones del tipo serie.

El bajo costo de los modems y de las comunicaciones telefónicas, han hecho al protocolo serial IP inmensamente popular, especialmente proveyendo a los usuarios un acceso de bajo costo a Internet. El hardware requerido para utilizar SLIP o PPP es de fácil instalación y bajo costo. Lo único que se requiere es un módem conectado a un puerto serie con búffer FIFO.

El protocolo SLIP es fácil de implementar y al mismo tiempo es el más común de los dos, pero PPP viene ganando terreno rápidamente. El protocolo PPP agrega muchas y sofisticadas opciones que contribuyen a su creciente popularidad en el presente y que lo convertirán en un protocolo más importante en el futuro.

El núcleo básico Linux soporta SLIP y PPP de forma muy estable y segura. En este capítulo y el siguiente, se discutirán ambos protocolos y cómo configurarlos.

7.1. Requerimientos Generales

Para utilizar SLIP o PPP, se requieren algunas configuraciones básicas de red ya descritas en capítulos anteriores. Usted debe configurar la interfaz de bucle *(loopback)* y el sistema de traducción de nombres. Cuando se conecta a Internet, querrá utilizar el servidor DNS. Sus opciones aquí son las mismas que en PPP: Se puede interrogar a los servidores DNS de su proveedor de Internet, colocando sus direcciones en el fichero `/etc/resolv.conf`, o instalar y configurar un servidor de nombres sólo con caché como se describe en Sección 6.3.4, en Capítulo 6.

7.2. Operación de SLIP

Los servidores IP que ofrecen enlaces telefónicos vía SLIP, por lo general, utilizan cuentas de usuario especiales. Una vez iniciada exitosamente la sesión, el servidor comienza a ejecutar un guión *(script)* para la activación del manejador SLIP y las interfaces de red apropiadas. Al mismo tiempo, en su terminal debe ocurrir exactamente lo mismo.

En algunos sistemas operativos, el gestor SLIP es un programa de usuario. Bajo Linux, es parte del núcleo del sistema, cosa que lo hace mucho más rápido. Esta ventaja, sin embargo, requiere que la línea sea convertida a modo SLIP de forma explícita. Esta conversión, es llevada a cabo mediante una disciplina de terminal (tty) especial llamada SLIPDISC. Mientras que un terminal (tty) trabaja en forma normal (DSIC0), los datos intercambiados entre los procesos del usuario, se realizan mediante las llamadas `read(2)` y `write(2)` estándar, y el manejador SLIP es incapaz de leer o escribir en este modo. En SLIPDISC se invierten los roles: ahora ningún proceso de usuario podrá escribir o leer desde un terminal (tty), ya que son dirigidos desde el puerto serie al gestor de SLIP.

El gestor SLIP entiende por sí mismo distintas variantes del protocolo SLIP. Además, de las variantes ordinarias, es capaz de interpretar una variante del protocolo llamada CSLIP, cuya particularidad es la de utilizar el método de compresión de cabeceras de Van Jacobson*(descritas en el RFC−1144)* para los paquetes IP salientes. Este método aumenta el rendimiento de las sesiones interactivas. Además, existen versiones de seis bits de cada uno de estos protocolos.

Una forma simple de convertir una línea serie al modo SLIP es usando la herramienta **slattach**. Suponiendo que su módem está en `/dev/ttyS3` y que ha podido acceder correctamente al servidor SLIP de forma correcta, deberá ejecutar:

```
# slattach /dev/ttyS3 &
```

Esta herramienta cambiará la disciplina de línea de `ttyS3` a `SLIPDISC` y lo enlazará a una de las interfaces SLIP. Si este es el primer enlace activo SLIP, será enlazado a `sl0`; el segundo, será enlazado a `sl1` y así, sucesivamente. Los núcleos actuales soportan hasta un máximo de 256 enlaces SLIP simultáneamente.

Por omisión **slattach**usa CSLIP como método de compresión de cabeceras. Con el parámetro `p`, usted puede seleccionar cualquier otra disciplina de línea. Para utilizar SLIP de forma normal (sin compresión) se debe teclear:

```
# slattach −p slip /dev/ttyS3 &
```

Las disciplinas disponibles se muestran en laTabla 7−1. Una seudo−disciplina disponible llamada `adaptive`, (adaptativa) deja al núcleo averiguar que tipo de encapsulado SLIP se está utilizando.

Tabla 7−1. Disciplinas de línea SLIP bajo Linux

Disciplina	**Descripción**
slip	Encapsulación tradicional.
cslip	Encapsulación SLIP con compresión de cabeceras Van Jacobsen.
slip6	Encapsulación SLIP con codificación a 6 bits. Este método de codificación es similar al usado por la orden **uuencode**, y causa que los datagramas SLIP sean convertidos a caracteres ASCII. Esta conversión es útil cuando no se poseen enlaces en serie con el octavo bit vacío.
cslip6	Encapsulación SLIP con compresión tipo Van Jacobsen de cabeceras y y codificado a 6 bits.
adaptive	No es una disciplina de línea real, y posibilita que el núcleo intente identificar la disciplina de línea usada en la máquina remota y hacer que concuerden.

Observe que debe utilizarse el mismo sistema de encapsulación que la máquina remota. Por ejemplo, si cowslip usara CSLIP, tendría que usarlo usted también. Si su conexión SLIP no funciona, lo primero que hay que saber es si en los dos puntos de conexión, se está utilizando compresión de cabeceras o no. Si no está seguro, intente configurar el gestor SLIP para que trabaje de forma adaptativa y que el núcleo se tome el trabajo de averiguarlo por usted.

slattach no sólamente configura el protocolo, sino también PPP o KISS (otro protocolo utilizado en redes tipo ham radio). Hacer esto no es común, ya que existen mejores herramientas para gestionar estos protocolos. Para más detalles, consulte las páginas man de `slattach(8)`.

Teniendo al gestor SLIP funcionando correctamente, se debe configurar la interfaz de red. Nuevamente, puede utilizar las órdenes **ifconfig** y **route** para configurar la interfaz. Asumiendo que ya estableció una conexión telefónica con un servidor llamado cowslip desde vlager. En **vlager** se debe ejecutar:

```
# ifconfig sl0 vlager−slip pointopoint cowslip
# route add cowslip
# route add default gw cowslip
```

La primera orden realiza un enlace punto a punto con cowslip, mientras que la segunda y la tercera orden sirven para añadir la ruta correspondiente a cowslip como ruta predeterminada y configurar a cowslip como pasarela *(gateway)*.

Dos cosas que no tienen nada que ver con la invocación de **ifconfig**: La opción pointopoint especifica la dirección del servidor remoto en nuestro enlace punto a punto y vlager−slip es la dirección local de la interfase SLIP.

Ya se ha mencionado que se puede utilizar la misma dirección asignada para la interfaz Ethernet vlagercomo para su enlace SLIP. En este caso, vlager−slip necesita otro alias para la dirección 172.16.1.1. Sin embargo, es posible tener una dirección completamente distinta para su enlace SLIP. Este es el caso cuando no se tienen direcciones IP registradas como con Cervecera. En la próxima sección se hablará con más detalle de este tipo de escenario.

Como referencia, siempre se usará vlager−slip para referirse a su interfase local SLIP.

Cuando se quiera terminar el enlace SLIP, debe empezarse por eliminar todas las rutas que pasan por cowslip usando la orden **route** con la opción `del`, luego desactivar la interfaz, y enviar al proceso **slattach** la señal de colgar. Después se podrá colgar el módem usando el terminal:

```
# route del default
# route del cowslip
# ifconfig sl0 down
# kill −HUP 516
```

Nota: el número *516* deberá ser reemplazado por el correspondiente identificador de proceso (como muestran las salidas de las órdenes **ps ax**) para el proceso **slattach** que controla al gestor SLIP que se quiera desconectar.

7.3. Trabajando con direcciones de red IP privadas

Como usted recordará del Capítulo 5, donde la Cervecera Virtual era una red Ethernet basada en direcciones IP usando números de red sin registrar que se usaban para uso interno, los paquetes de o desde esa red no estaban encaminados hacia Internet; si se quiere que vlager llame a través de cowslip y actúe como encaminador para la red en la Cervecera Virtual, las máquinas pertenecientes a la red de la cervecera no podrán acceder directamente a Internet ya que los paquetes serán descartados por el encaminador principal.

Para trabajar alrededor de este dilema, se tendrá que configurar a vlager para que actúe como plataforma de lanzamiento para el acceso a Internet. Para el resto del mundo, esta conexión se presenta como un enlace normal SLIP, accediendo a los servidores con una dirección IP registrada (probablemente asignada por el proveedor de servicios corriendo cowslip). Cualquier usuario conectado a vlager puede usar clientes basados en texto como **ftp**, **telnet**, o incluso **lynx** para usarlos en Internet. Un usuario de la Cervecera Virtual puede invocar a telnet y teclear a vlager para usar dichos clientes de forma normal. Para algunas aplicaciones, existen algunas soluciones para evitar el proceso de registro e identificación de los usuarios que accedan a vlager. Para usuarios de Web, por ejemplo, se puede activar un servidor llamado *proxy* en vlager, que encaminará todas las peticiones de los usuarios hacia los servidores respectivos.

El proceso de autenticación en vlager para poder acceder a los servicios de Internet, no tiene mucho sentido. Pero aparte de eliminar el papeleo y el costo de registrar una dirección IP, se tiene el beneficio de poder instalar un cortafuegos. Los cortafuegos son servidores dedicados usados para proveer acceso limitado a los usuarios de una red sin exponer a los servidores internos a ataques desde fuera. Configuraciones simples de cortafuegos se explican al detalle en Capítulo 9. En el Capítulo 11, se discute una característica de Linux llamada IP masquerade (enmascaramiento de IP) esto provee una alternativa poderosa comparada a los servidores proxy.

Asumiendo que la cervecera ha sido asignada a la dirección IP 192.168.5.74 para acceso a SLIP. Todo lo que usted debe realizar para la puesta en marcha discutida anteriormente es teclear la dirección en el fichero `/etc/hosts`, nombrándolo como vlager−slip. Este procedimiento no cambiará su enlace SLIP.

7.4. Usando dip

Lo visto anteriormente es simple. Sin embargo, se pueden automatizar estas tareas. Es mucho más práctico tener solamente un orden que realice los pasos necesarios para activar la línea serie, que el módem llame al proveedor de Internet, comenzar la sesión, activar la disciplina de línea SLIP, y por último, configurar la interfaz de red. Para todo esto esta la orden **dip**.

dip significa *Dialup IP* (enlace IP telefónico). Fue escrito por Fred van Kempen y ha sufrido bastantes modificaciones de mucha gente. Es actualmente utilizado por todo el mundo. La versión `dip337p-uri`, se encuentra en casi todas las distribuciones de GNU/Linux actuales, o también a través de FTP en metalab.unc.edu.

dip provee un intérprete para un lenguaje de guiones simple que puede manejar el módem , convertir la línea a modo SLIP y configurar las interfaces. Este lenguaje de guiones es poderoso a la hora de manejar diferentes configuraciones.

Para poder configurar la interfaz SLIP, **dip** requiere privilegios de superusuario. Puede hacerse cambiando al programa **dip** el bit setuid como root para que todos los usuarios puedan conectarse a cualquier servidor SLIP sin tener privilegios de superusuario. Esto es muy peligroso, ya que una configuración incorrecta del encaminamiento de **dip** puede estropear el encaminamiento en su red. Aun peor, esto dará a los usuarios la posibilidad de conectarse a *cualquier* servidor SLIP y lanzar desde allí, peligrosos ataques a su red. Si desea que los usuarios puedan activar conexiones SLIP, escriba pequeños programas empaquetados por cada perspectiva de conexión a los diferentes servidores SLIP y que esos pequeños programas invoquen a **dip** con guiones (scripts) específicos para establecer las conexiones. Bien escritos, estos programas pueden ser fácilmente habilitados con el bit setuid de superusuario (root).[66] Una alternativa un poco más flexible, es darle a los usuarios, acceso verdadero a **dip** como superusuario, utilizando alguna herramienta como por ejemplo **sudo**.

7.4.1. Un guión de ejemplo

Asumiendo que el servidor al cual nos queremos conectar vía SLIP se llama cowslip, y que se ha escrito un guión para que **dip** lo interprete llamado `cowslip.dip`, el cual hará la conexión. Al programa **dip**, hay que pasarle como argumento, el nombre del guión:

```
# dip cowslip.dip
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation.
connected to cowslip.moo.com with addr 192.168.5.74
#
```

El codigo del guión es mostrado en el Ejemplo 7−1.

Ejemplo 7−1. Un ejemplo de guión para dip

```
# Ejemplo de guión en dip para conectarse al servidor cowslip
# Configurar los nombres locales y remotos  y las direcciones
    get $local vlager−slip
    get $remote cowslip
    port ttyS3              # Selección del puerto serie
    speed 38400             # Configurar la velocidad máxima
    modem HAYES             # Selección del modelo del Módem
    reset                   # reiniciar el módem y la terminal
    flush                   # limpiar el búffer de respuesta del módem
# Prepararse para marcar.
    send ATQ0V1E1X1\r
    wait OK 2
```

```
        if $errlvl != 0 goto error
        dial 41988
        if $errlvl != 0 goto error
        wait CONNECT 60
        if $errlvl != 0 goto error

    # Bien, se estableció la conexión
        sleep 3
        send \r\n\r\n
        wait ogin: 10
        if $errlvl != 0 goto error
        send Svlager\n
        wait ssword: 5
        if $errlvl != 0 goto error
        send knockknock\n
        wait running 30
        if $errlvl != 0 goto error
    # Se comenzó la sesión, y del extremo remoto se activó SLIP.
        print Conectado a $remote con dirección $rmtip
        default                  # hacer que este enlace sea la ruta predeterminada
        mode SLIP                # Pasemos a modo SLIP
    # en caso de error se ejecuta lo siguiente
    error:
        print La onexión SLIP a $remote ha fallado.
```

Una vez conectado a cowslip y activo el SLIP, **dip** pasará a ejecutarse en segundo plano. Ahora puede empezar a trabajar con sus los programas habituales de red a través del enlace SLIP. Para terminar la conexión, simplemente invoque a **dip** con el parámetro `k`. Esto envía la señal de cortar (hangup) a **dip**, junto con el identificador de proceso que **dip** escribió en el fichero `/etc/dip.pid` al comenzar:

```
    # dip -k
```

En el lenguaje de guiones que **dip** interpreta, las palabras clave precedidas con el signo de dólar significan nombre de variables. **dip** tiene un conjunto de variables que están listadas a continuación. $remote y $local, por ejemplo, contienen los nombres de los computadores remoto y local involucrados en la conexión SLIP.

Las dos primeras declaraciones en el ejemplo, son órdenes **get**, que es la forma en que **dip** declara una variable. Aquí, los nombres de las computadoras local y remota son vlager y cowslip, respectivamente.

las cinco declaraciones siguientes preparan la terminal de línea y el módem . la orden reset envía la cadena de reinicio al módem. La siguiente sentencia limpia el búffer de salida del módem , para conseguir que el diálogo de registro (login) en las siguientes líneas trabaje correctamente. Este dialogo es extremadamente simple: simplemente marca 41988, el número telefónico de cowslip, y ae regiatra en la cuenta Svlager usando la contraseña knockknock. La orden wait hace que **dip** espere una cadena dada como primer argumento; el número dado como segundo argumento es el tiempo (en segundos) que se debe esperar por esa cadena. la orden if ,en el proceso de entrada, revisa que no se produzcan errores.

Las órdenes finales, tras un correcto registro, son default, que hace que el enlace SLIP sea la ruta predeterminada a todos los servidores, y mode, que activa el modo SLIP en la línea y configura la interfaz y la tabla de rutas.

7.4.2. Referencia de dip

En esta sección, se hará referencia a las órdenes de **dip** más usadas. usted puede obtener un vistazo de todos los órdenes reconocidos invocando a **dip** en modo prueba e introduciendo la orden help. Para conocer más sobre la sintaxis de una orden, se debe teclear sin argumentos. Recuerde que esto no funcionará con órdenes que no aceptan argumentos. El siguiente ejemplo ilustra el funcionamiento de la orden help:

```
    # dip -t
    DIP: Dialup IP Protocol Driver version 3.3.7p-uri (25 Dec 96)
```

```
Written by Fred N. van Kempen, MicroWalt Corporation.
Debian version 3.3.7p−2 (debian).

DIP> help
DIP knows about the following commands:

    beep        bootp       break       chatkey     config
    databits    dec         default     dial        echo
    flush       get         goto        help        if
    inc         init        mode        modem       netmask
    onexit      parity      password    proxyarp    print
    psend       port        quit        reset       securidfixed
    securid     send        shell       skey        sleep
    speed       stopbits    term        timeout     wait

DIP> echo
Usage: echo on|off
DIP>
```

En los párrafos siguientes, los ejemplos que muestran el cursor **DIP>** indican como teclear un orden en modo prueba y cual será su respuesta. Los ejemplos mostrados sin el cursor, deben tomarse como trozos de guiones.

7.4.2.1. Las órdenes del módem

dip provee algunas órdenes para configurar el puerto serie y el modem . Algunos son obvios como la orden port, que selecciona el puerto serie, y speed, databits, stopbits, y parity, que configura los parámetros más comunes de la línea. La orden módem selecciona el tipo de módem . Actualmente, sólo esta soportado el tipo HAYES (en mayúsculas). Se debe proveer a **dip** con el tipo de módem , o éste se negará a ejecutar las órdenes dial y reset. La orden reset envía la cadena de reinicio (reset) al módem; el tipo de cadena depende del modelo y marca del módem. Para modems compatibles con Hayes, esta cadena es ATZ.

la orden flush puede usarse para vaciar todas las respuestas que el módem envió hasta ese momento. De otro modo, un guión de diálogo que ejecute un reset podría confundirse si leyese un OK como respuesta de algún orden anterior.

La orden init selecciona la cadena de inicialización enviada al módem antes de marcar. predeterminada, para modems Hayes es ATE0 Q0 V1 X1, que activa el eco de las órdenes y los códigos de retorno, además selecciona el modo de marcado a ciegas (no comprueba si la línea tiene tono). Los modems modernos, vienen con una buena configuración de fabrica, así que esto es un poco innecesario, pero no hace daño alguno.

La orden dial envía la cadena de inicialización al módem y llama al sistema remoto. La orden predeterminada para modems tipo Hayes es ATD.

7.4.2.2. La Orden echo

La orden echo sirve como depurador. Invocar echo on hace que **dip** copie en la consola todo lo que se envía por el puerto serie. Este modo puede desactivarse invocando echo off.

dip Puede salir del modo guión temporalmente para entrar en modo terminal. En este modo, usted puede usar a **dip** como cualquier programa de terminal ordinario, enviando caracteres a través de la línea serie, leyéndolos y mostrarlos. Para abandonar este modo, presione Ctrl−].

7.4.2.3. La Orden get

la orden get es la forma en que **dip** carga una variable. Su uso más simple es inicializar una constante, como se vió en el ejemplo de `cowslip.dip`. usted también puede utilizarlo desde la consola en conjunto con ask:

```
DIP> get $local ask
```

```
Enter the value for $local: _
```

Un tercer método, es usado para obtener el nombre del servidor remoto. Aunque extraño parezca al principio, es muy útil en algunos casos. Algunos servidores, no permiten que usted use su propia dirección IP en un enlace SLIP, sino que le asignará una de un conjunto ya establecido cuando se establezca la conexión, mostrando un mensaje que le informe que dirección le fue asignada. Si el mensaje luce parecido a `Your address: 192.168.5.74`, el siguiente ejemplo hará que **dip** use la dirección asignada:

```
# finish login
wait address: 10
get $locip remote
```

7.4.2.4. La Orden print

Esta orden es usado para enviar cualquier texto a la consola cuando se invoque a **dip**. Cualquier variable usada por **dip** puede ser utilizada para enviar mensajes, como por ejemplo:

```
DIP> print Usando puerto $port a la velocidad de $speed baudios
Usando puerto ttyS3 a la velocidad de 38400 baudios
```

7.4.2.5. Nombres de variables

dip entiende solamente un grupo predefinido de variables. Un nombre de variable siempre debe comenzar con el signo de dólar y debe estar en minúsculas.

Las variables $local y $locip contienen el nombre de la máquina local y su dirección IP. Cuando se guarda el nombre canónico de la máquina local en $local, **dip** intentará resolverlo para conseguir la dirección IP y guardarla en la variable $locip. Un proceso similar, pero al revés, sucede cuando se guarda la dirección IP en la variable $locip; **dip** intentará resolver el nombre de la máquina local a partir de la dirección IP y guardarlo en la variable $local.

Las variables $remote y $rmtip operan de la misma manera, pero con el nombre y la dirección IP de la máquina remota. la variable $mtu contiene el valor MTU para la conexión actual.

Estas cinco variables son las únicas que pueden ser asignadas con valores usando la orden get. El contenido de algunas variables, son el resultado de configuraciones realizadas por órdenes que llevan el mismo nombre, pero pueden ser utilizadas junto con la orden print; Estas variables son $modem, $port, y $speed.

La variable $errlvl contiene el resultado de la última orden ejecutada. Un nivel de error 0 indica que la orden se ejecutó satisfactoriamente, si este número es mayor o menor, indica que hubo algún problema en la ejecución.

7.4.2.6. Órdenes if y goto

la orden if ejecuta un salto condicional, y se comporta de la misma manera que su par usado en programación. Su sintaxis es:

```
if var op number goto label
```

La expresión, realiza una simple comparación entre una de estas variables $errlvl, $locip, y $rmtip. *`var`* debe ser un numero entero (integer); el operador *`op`* debe ser uno de estos: ==, !=, <, >, <=, y >=.

La orden goto (ir a) hace que la ejecución del guión continúe donde se encuentra definida la etiqueta pasada como parámetro a la orden (*`label`*). Una etiqueta (label) debe ser la primer palabra en una línea, seguida de dos puntos(:)

7.4.2.7. Órdenes send, wait, y sleep

Estas órdenes ayudan a implementar sencillos guiones de diálogo en **dip**. la orden send envía sus argumentos a la línea serie. No soporta el uso de variables, pero entiende todas las secuencias de escape al estilo del lenguaje C, como \n para nueva línea y \b para retroceso. El carácter de tilde (~) puede ser usada como una abreviatura del carácter de retorno de carro / nueva línea.

La orden wait toma una palabra como argumento y leerá todo lo que entre por la línea serie hasta que detecte una secuencia de caracteres que coincida con esa palabra. Esa palabra no puede contener caracteres en blanco. Opcionalmente, se le puede pasar a wait un tiempo de espera como segundo argumento; Si la palabra esperada, no es recibida en ese tiempo de espera la variable $errlvl se cargará con un 1. Esta orden se usa generalmente en la detección de registros (login) y otros símbolos de espera.

La orden sleep puede usarse para esperar una determinada cantidad de tiempo; Por ejemplo, esperar pacientemente a que la secuencia de ingreso se complete. Nuevamente, el intervalo se expresa en segundos.

7.4.2.8. Órdenes mode y default

Estas órdenes se usan para pasar la línea de modo serie a modo SLIP y para configurar la interfaz.

La orden mode es la última ejecutada por **dip** antes de pasar al modo demonio. A menos que ocurra un error, esta orden no retorna.

mode toma el nombre del protocolo como argumento. La versión actual de **dip** reconoce los siguientes: SLIP, CSLIP, SLIP6, CSLIP6, PPP, y TERM como nombres válidos. Esta versión de **dip** no entiende SLIP adaptable.

Después de poner la línea en modo SLIP, **dip** ejecuta **ifconfig** para configurar la interfaz como enlace punto a punto, e invocar a **route** para cambiar el encaminamiento hacia el servidor remoto.

Si, además, el guión se ejecutase la orden default antes que mode, **dip** hará que el camino predeterminado de los paquetes sea encaminado al enlace SLIP.

7.5. Funcionamiento en modo Servidor

Configurar el cliente SLIP fue la parte difícil. Configurar su máquina para que actúe como servidor SLIP, es mucho más fácil

Existen dos formas de configurar al servidor SLIP. Las dos requieren que se cree una cuenta de acceso por cada cliente SLIP. Asuma, por ejemplo, que le desea conceder acceso al servicio SLIP a Arthur Dent en dent.beta.com. Debería crearse una cuenta llamada dent añadiendo la siguiente línea al fichero `passwd`:

```
    dent:*:501:60:Arthur Dent's SLIP account:/tmp:/usr/sbin/diplogin
```

Luego, se puede establecer la contraseña de dent utilizando la herramienta **passwd**.

La orden **dip** puede usarse en modo servidor invocando **diplogin**. Usualmente **diplogin** es un enlace a **dip**. Su fichero de configuración principal es `/etc/diphosts`, donde se especifican las direcciones IP que serán asignadas a los usuarios cuando se conecten vía SLIP. De forma alternativa, se puede usar la orden **sliplogin**, una herramienta derivada de BSD que contiene muchos y más flexibles esquemas de configuración posibilitando la ejecución se guiones cuando el servidor se conecta y desconecta.

Cuando dent se conecta usando SLIP vía, **dip** activa el servidor. Para saber si tiene acceso al uso de SLIP, el servidor se fija en el fichero `/etc/diphosts`. En este fichero se detallan los derechos de acceso y algunos parámetros de conexión por cada usuario que accede vía SLIP. El formato general para el fichero

`/etc/diphosts` es el siguiente:

```
# /etc/diphosts
user:password:rem-addr:loc-addr:netmask:comments:protocol,MTU
#
```

Cada uno de los campos es descrito en la Tabla 7−2.

Tabla 7−2. Descripción de campos en /etc/diphosts

Campo	**Descripción**
`user`	El nombre de usuario que invoca a **dip**.
password	El segundo campo en `/etc/diphosts` sirve para darle una capa extra de seguridad basada en una segunda contraseña a la conexión. Se puede teclear una contraseña encriptada aquí (como en `/etc/passwd`) y **diplogin** se la preguntará al usuario antes del inicio de conexión SLIP. Tenga en cuenta que esta contraseña es adicional al proceso de ingreso estándar (**login**).
`rem-addr`	La dirección que se le asignará al cliente remoto. Esta dirección se puede especificar mediante el nombre de la máquina (que debe ser resuelto) o por medio del formato numérico tradicional
`loc-addr`	La dirección que es usada por el servidor en el enlace SLIP. Como la anterior, puede ser escrita como nombre de máquina o como dirección de IP.
`netmask`	La máscara de red (netmask) es usada para propósitos de enrutamiento. Mucha gente se confunde con este campo. La máscara de red no es aplicada al enlace SLIP en si mismo, pero es usada en combinación con el campo `rem-addr` para producir una ruta a la máquina remota. La máscara de red sólo debe usarse para dar soporte de red al cliente remoto.
`comments`	Este es un campo donde puede anotar comentarios acerca del usuario, no tiene otro propósito.
`protocol`	Este es el campo donde se especifica el protocolo o la disciplina de línea que se desea aplicar a la conexión. Son entradas válidas aquí de la misma forma que el parámetro `p` en la orden **slattach**.
`MTU`	Es la máxima unidad de transmisión que el enlace puede manejar. Este campo describe el tamaño máximo del datagrama que puede ser transmitido por ese enlace. Cualquier datagrama enviado a ese enlace SLIP que sea más grande que los especificado en el valor MTU, será fragmentado en trozos más pequeños. Usualmente, el valor MTU es idéntico en ambos extremos del enlace.

Un ejemplo para el usuario dent se vería de esta manera::

```
dent::dent.beta.com:vbrew.com:255.255.255.0:Arthur Dent:CSLIP,296
```

Este ejemplo le da al usuario dent acceso a SLIP sin contraseña adicional. Es asociado con la dirección IP de dent.beta.com,y la máscara de red `255.255.255.0`. Por omisión será encaminado por vbrew.com, , y se usará CSLIP como protocolo con un MTU de 296 bytes.

Cuando dent entra en su cuenta, **diplogin** extrae la información de él desde el fichero `diphosts`. Si el segundo campo contiene algún valor, **diplogin** le preguntará la segunda contraseña de seguridad. La cadena introducida por el usuario, se encripta y se compara con la que existe en el fichero `diphosts`. Si éstas no coinciden, el intento de registro se rechazao. Si la cadena de contraseña usa el método s/key, y **dip** fue compilado para dar soporte a S/Key, el proceso de autenticación tendrá lugar. Este proceso de autenticación, se describe en la documentación que acompaña a los fuentes de **dip**.

Tras un registro exitoso, **diplogin** procede a convertir la línea serie en modo CSLIP o SLIP y prepara la interfaz y el encaminamiento. Esta conexión permanecerá activa hasta que el usuario decida cortarla, con lo cual **diplogin** restaurará la disciplina de línea y terminará.

diplogin requiere privilegios de superusuario. Si no tiene a **dip** con el setuid de root, se debe hacer que **diplogin** sea una copia separada de **dip** o un enlace. **diplogin** puede tener este privilegio de forma segura, sin afectar a **dip** en sí mismo.

Capítulo 8. El Protocolo Punto−a−Punto

Como SLIP, PPP es un protocolo usado para mandar datagramas a través de una conexión serie; sin embargo, resuelve varias de las deficiencias de SLIP. Primero, puede transportar un alto número de protocolos y no está limitado al protocolo IP. Proporciona detección de errores en el mismo enlace, mientras que SLIP acepta y reenvía datagramas corruptos mientras que la corrupción no se produzca en la cabecera. Igualmente importante, permite a los extremos comunicantes negociar opciones, como la dirección IP y el tamaño máximo del datagrama, y provee autentificación del cliente. Esta negociación interna, permite una automatización fiable del establecimiento de la conexión. Mientras la autentificación elimina la necesidad de cuentas de usuario que requiere SLIP. Para cada una de estas capacidades, PPP tiene un protocolo específico. En este capítulo cubrimos brevemente estos elementos básicos que forman PPP. Esta discusión de PPP está lejos de ser completa; si quiere aprender más sobre PPP, le instamos a que lea el RFC de su especificación y la alrededor de docena de RFCs que lo complementan.[67] Además hay un extenso libro de O'Reilly sobre el tema *Using & Managing PPP*, por Andrew Sun.

En la parte más baja de PPP está el protocolo de *Control de Conexión de Datos de Alto−Nivel* (HDLC), que define los límites de las tramas PPP individuales, y proporciona un control de errores de 16 bit.[68] Al contrario de lo que ocurría con SLIP, una trama PPP es capaz de llevar paquetes de otros protocolos distintos al IP, como IPX de Novell o Appletalk. PPP consigue esto añadiendo a la trama básica HDLC un campo de control que identifica el tipo de paquete contenido en la misma.

El *Protocolo de Control de Enlace*, es utilizado en la parte más alta del HDLC para negociar las opciones concernientes a la conexión de datos, tales como la *Unidad Máxima de Recepción* (MRU), que establece el tamaño máximo del datagrama que cada extremo de comunicación acepta recibir.

Un paso importante en la configuración del enlace PPP corresponde a la autentificación del cliente. Aunque no es obligatorio, es casi un deber para las líneas telefónicas y así mantener fuera a los intrusos. Normalmente la máquina llamada (el servidor) pide al cliente que se identifique probando que se sabe alguna clave secreta. Si el llamante se equivoca con la clave, la conexión termina. Con PPP, las autorizaciones se producen en los dos sentidos; es decir, el que llama también puede pedir al servidor que se autentifique. Estos procedimientos de autentificación son totalmente independientes entre sí. Hay dos protocolos distintos, según el tipo de autentificación, los cuales discutiremos más adelante en este capítulo: *el Protocolo de Autentificación por Contraseña* (PAP) y *el Protocolo de Autentificación por Reto* (CHAP).

Cada protocolo de red que es encaminado a través de la conexión de datos, como el IP, el Appletalk, etc; se configura dinámicamente usando el correspondiente *Protocolo de Control de Red* (NCP). Por ejemplo, para enviar datagramas IP a través del enlace, los dos nodos tienen que negociar en primer lugar qué direcciones IP van a utilizar. El protocolo de control utilizado para esto es el *Protocolo de Control del IP* (IPCP).

Aparte de enviar datagramas IP estándar a través del enlace, el PPP también permite la compresión Van Jacobson de las cabeceras en los datagramas IP. Es una técnica para reducir las cabeceras de los paquetes TCP a un espacio de tan sólo tres bytes. También se utiliza en el CSLIP, y es conocida coloquialmente como compresión de cabeceras VJ. La utilización de la compresión puede negociarse también al comienzo de la conexión gracias al IPCP.

8.1. PPP en Linux

En GNU/ Linux, la funcionalidad del PPP está dividida en dos partes: un componente del núcleo que controla los protocolos de bajo nivel (HDLC, IPCP, IPXCP, etc.) y el demonio **pppd** en espacio de usuario que controla varios protocolos de alto nivel, como PAP Y CHAP. La versión actual de PPP para Linux contiene el demonio PPP pppd y un programa llamado **chat** utilizado para llamar al sistema remoto.

El controlador de PPP para el núcleo fue escrito por Michael Callahan y reescrito por Paul Mackerras. El

pppd fue escrito a partir de una implementación [69] gratuita de PPP para máquinas Sun y 386BSD que a su vez fue escrita por Drew Perkins y otros programadores, y mantenida por Paul Mackerras. Fue adaptada a GNU/Linux por Al Longyear.7 **Chat** fue escrito por Karl Fox. [70]

Al igual que SLIP, PPP está implementado a través de una disciplina especial para la utilización de las líneas. Para utilizar una línea serie como enlace PPP, en primer lugar tendrá que establecer la conexión con su módem, como es usual; y posteriormente pasar la línea al modo PPP. En este modo, todos los datos que nos llegan son pasados al controlador PPP, que comprueba la validez de las tramas HDLC que llegan (cada trama HDLC trae un código de control de errores de 16 bit), las descompone y las despacha. Actualmente, PPP es capaz de transportar indistintamente el protocolo IP, opcionalmente usando la compresión de cabeceras Van Jacobson, y el protocolo IPX.

El controlador del núcleo es ayudado por **pppd**, el demonio del PPP, que realiza toda la fase de inicialización y autentificación necesaria antes de que el verdadero tráfico de red pueda ser enviado a través del enlace. El comportamiento del **pppd** puede ser ajustado utilizando varias opciones. Como el PPP es bastante complejo, es imposible explicar todas ellas en un sólo capítulo. Por eso, este libro no puede cubrir todos los aspectos de **pppd**, sino sólamente darle una introducción. Para más información, consulte *Using & Managing PPP* y las páginas de manual y los ficheros `README` de la distribución con las fuentes de **pppd**, que deberían ayudarle a comprender la mayor parte de las cuestiones que este capítulo no trata. El PPP−HOWTO también debería serle de ayuda.

Probablemente la mejor ayuda que encontrará para configurar PPP vendrá de otros usuarios de su misma distribución. Las preguntas sobre la configuración de PPP son muy comunes, así que pruebe en su grupo lista de correo local o en el canal de Linux del IRC. Si su problema persiste incluso después de leer toda la documentación, debería pasarse por el grupo de noticias comp.protocols.ppp para solicitar ayuda, que es el lugar donde encontrará a la mayor parte de la gente envuelta en el desarrollo del **pppd**.

8.2. Ejecutando pppd

Cuando quiere conectarse a Internet a través de un enlace PPP, tiene que configurar las capacidades básicas de red como el dispositivo de bucle local (loopback) y el sistema de resolución de direcciones. Las dos han sido explicadas en Capítulo 5, y Capítulo 6. Usted puede configurar simplemente el servidor de nombres de su proveedor de servicios de Internet en el fichero `/etc/resolv.conf`, pero esto supondrá que cada consulta DNS será enviada a través de su enlace serie. Esta situación no es óptima; mientras más cerca se encuentre de su servidor de nombres, más rápida será la búsqueda. Una solución alternativa es configurar una estación en su red que dé servicio de servidor de nombres en modo sólo−cachéo. Esto significa que la primera vez que realice una consulta DNS de un nodo en particular, su consulta será enviada a través de su línea serie, pero el resto de las consultas a esta misma máquina, serán directamente resueltas por su servidor de nombres local de una forma mucho más rápida. Esta configuración está descrita en el Capítulo 6, en Sección 6.3.4.

Como ejemplo introductorio de como establecer una conexión PPP con **pppd**, suponga que está de nuevo en vlager. Ya ha llamado al servidor PPP, c3po, y entrado en la cuenta del usuario ppp. c3po ya ha lanzado su controlador PPP. Después de salir del programa de comunicaciones que utilizó para llamar, ejecute la siguiente orden, sustituyendo el nombre del dispositivo serie que usted usa por el `ttyS3` mostrado aquí:

```
# pppd /dev/ttyS3 38400 crtscts defaultroute
```

Esto cambiará la línea serie `ttyS3` al modo PPP y establecerá un enlace IP con c3po. La velocidad de transferencia utilizada en el puerto de serie será de 38400bps. La opción crtscts activa el control de flujo por hardware en el puerto, que es una obligación para velocidades superiores a los 9600 bps.

Lo primero que hace **pppd** tras ejecutarse es negociar varias características para el enlace con el extremo remoto utilizando LCP. Normalmente, el conjunto de opciones que intenta negociar **pppd** funcionará, así que no nos meteremos más con este asunto. Digamos que parte de esta negociación envuelve la solicitud o asignación de las direcciones IP en ambos extremos del enlace.

Hasta ahora, también hemos asumido que c3po no necesita ninguna autentificación de nosotros, así que la fase de configuración habrá sido completada con éxito.

pppd negociará entonces los parámetros IP con su compañero usando IPCP, el protocolo de control IP. Al no especificar dirección IP alguna, el **pppd** intentará usar la dirección que se obtiene al resolver el nombre del ordenador local. Decididas las direcciones, cada **pppd** comunicará su dirección al otro extremo.

Normalmente no habrá ningún problema con esta configuración predeterminada. Incluso si su máquina esta en una Ethernet, puede utilizar la misma dirección IP tanto para la Ethernet como para el interfaz PPP. No obstante, **pppd** le permite utilizar una dirección diferente, o incluso pedir a su compañero que utilice alguna dirección específica. Estas opciones serán discutidas más adelante en la sección Sección 8.5.

Tras pasar por la fase de configuración IPCP, **pppd** configurará la red de su ordenador para utilizar el enlace PPP. En primer lugar, configurará el interfaz de red PPP como un enlace punto−a−punto, utilizando `ppp0` para el primer enlace PPP que esté activo, `ppp1` para el segundo, y así sucesivamente. A continuación preparará una entrada de la tabla de encaminamiento que apunte al ordenador del otro extremo del enlace. En el ejemplo anterior, el **pppd** hará que el encaminamiento de red predeterminado apunte a c3po, debido a que lo especificamos con la opción defaultroute.[71] Esto provoca que todos los datagramas dirigidos a ordenadores que no estén en su red sean enviados a c3po; esto es debido a que es el único camino por el que se puede llegar a esas máquinas. Hay un variado número de formas de encaminamiento que acepta el **pppd**, y las cubriremos en mayor detalle más adelante.

8.3. Usando los Ficheros de Opciones

Antes de que el **pppd** procese los argumentos de su línea de órdenes, echa un vistazo a varios ficheros para establecer sus opciones por omisión. Estos ficheros pueden contener cualquier argumento de línea de órdenes válido, distribuidos a través de un cierto número de líneas. Los comentarios se escriben tras el símbolo de almohadillado (#).

El primer fichero de opciones es el `/etc/ppp/options`, que es leído cada vez que el **pppd** arranca. Utilizarlo para establecer algunas opciones globales por omisión es una buena idea, pues le permite evitar que sus usuarios hagan ciertas cosas que podrían comprometer la seguridad del sistema. Por ejemplo, para hacer que el **pppd** necesite algún tipo de autentificación del otro sistema, añadiría la opción `auth` a este fichero. Esta opción no puede ser evitada por el usuario, de forma que se hace totalmente imposible el establecer una conexión PPP con cualquier sistema que no esté en nuestras bases de datos para la autentificación. De todas formas, algunas opciones sí que pueden ser evitadadas; la cadena CONNECT es un buen ejemplo.

El otro fichero de opciones, que es leído después del `/etc/ppp/options`, es el `.ppprc` situado en el directorio de inicio del usuario. Permite que cada usuario especifique su propio conjunto de opciones predeterminadas.

Un fichero `/etc/ppp/options` de ejemplo puede parecerse a éste:

```
# Opciones globales para el pppd de vlager.vbrew.com

lock                         # usar el bloqueo de dispositivo tipo UUCP
auth                         # obligar a autentificación
usehostname                  # usar el nombre del ordenador local para el CHAP
domain vbrew.com    # nombre de nuestro dominio
```

La expresión lock hace que **pppd** utilice el método de bloqueo de dispositivos estándar de UUCP. De esta manera, cada proceso que accede a un dispositivo serie, por ejemplo el `/dev/ttyS3`, crea un fichero de bloqueo llamado `LCK..ttyS3` en el directorio de spool del UUCP para señalizar que ese dispositivo esta siendo usado. Esto es necesario para evitar que otros programas, como pueden ser **minicom** o **uucico**, abran el dispositivo de serie mientras éste se usa por PPP.

La tres opciones siguientes se refieren a la autentificación y, por contrapartida, a la seguridad del sistema. Las opciones de autentificación están mejor colocadas en el fichero global de configuración porque tienen privilegios y no pueden ser sobreescritas por los ficheros `~/.ppprc`de los usuarios.

8.4. Realización de la Llamada con chat

Uno de los problemas que puede haberle dado el ejemplo anterior es que tenía que establecer la conexión manualmente antes de poder ejecutar **pppd**. Al contrario que **dip**, **pppd** no tiene su propio lenguaje de scripts para llamar al sistema remoto y entrar en él, sino que confía en otro programa externo para que haga esto. El comando que tiene que ser ejecutado puede dársele **pppd** con la opción connect en la línea de órdenes. El **pppd** redirigirá la entrada y salida estándar de órdenes a la línea serie.

El paquete **pppd** incluye un programa muy simple llamado chat que es capaz de automatizar secuencias de registro simples. Hablaremos sobre este programa con detalle.

Si su secuencia de registro es compleja, necesitará algo más potente que **chat**. Una alternativa útil que debería considerar es **expect**, escrito por Don Libes. Tiene un lenguaje basado en Tcl, y fue diseñado exactamente para este tipo de aplicación. Aquellos de vosotros cuyas secuencias de registro requieran, por ejemplo, autentificación por reto/respuesta con generación de claves encontraréis **expect** lo suficientemente potente para ocuparse de la tarea. Puesto que hay tantas posibles variaciones de este tema, no describiremos como desarrollar un guión de expect apropiado en este libro. Es suficiente decir, que usted llama a su guión expect especificando su nombre con la opción connect de **pppd**. También es importante señalar que cuando el guión está ejecutándose, la entrada y la salida estándar estarán desviadas al módem, y no a la terminal que invocó al **pppd**. Si usted necesita interactuar como usuario, usted deberá hacerlo mediante un terminal virtual compartido, o mediante otros medios.

La orden **chat** le permite especificar un script del estilo de los de UUCP. Básicamente, un script de chat consiste en una secuencia alterna de cadenas que esperamos recibir del sistema remoto y las respuestas que hemos de enviar. Las llamaremos respectivamente, cadenas *esperadas* y cadenas *enviadas*. Este es un extracto de un típico script de chat:

```
ogin: b1ff ssword: s3|<r1t
```

Este script le indica a **chat** que espere a que el sistema remoto le envíe el mensaje de petición de usuario y entonces le devuelve el nombre del usuario b1ff. Sólo esperamos por ogin: para que no importe si el mensaje de registro empiece por l mayúscula o minúscula, o si llega con basura. La siguiente cadena es una cadena esperada que hace que **chat** espere al mensaje de petición de contraseña y le envíe nuestra contraseña como respuesta.

Esto es básicamente lo que hacen los scripts de chat. Un script completo para llamar a un servidor PPP debería, además, incluir las órdenes apropiados para el módem. Suponga que su módem entiende los comandos Hayes, y que el número de teléfono del servidor es el 318714. En ese caso, la línea completa de **chat** para que pudiésemos establecer una conexión con c3po sería:

```
$ chat -v '' ATZ OK ATDT318714 CONNECT '' ogin: ppp word: GaGariN
```

Por definición, la primera cadena que damos a chat tiene que ser una cadena esperada, pero como el módem no dirá nada hasta que hablemos con él, hacemos que **chat** la ignore especificando una cadena vacía. Continuamos enviando ATZ, la orden de inicialización para los módems compatibles Hayes, y esperamos a que nos responda con (OK). La siguiente cadena envía a **chat** la orden de marcado junto con el número de teléfono, y espera a que aparezca el mensaje CONNECT como respuesta. Esto está seguido de otra cadena vacía otra vez, porque ahora no queremos enviar nada, sino esperar a que aparezca el mensaje de petición de registro. El resto del script de chat funciona exactamente como antes. Esta descripción probablemente parezca algo confusa, pero veremos en un momento que hay una forma de hacer los scrips chat mucho más fáciles de entender.

La opción `v` hace que **chat** capture todas las actividades hacia la facilidad local2 del demonio **syslog.** [72]

El escribir el script de chat directamente en la línea de órdenes implica un cierto riesgo, pues los usuarios pueden ver la línea de órdenes de un proceso con la orden **ps**. Puede evitar esto colocando el script del chat en un fichero, por ejemplo llamado `dial-c3po`. Entonces, podrá hacer al **chat** leer el script del fichero en vez de la línea de comando utilizando la opción `f`, seguida por el nombre del fichero.Esto supondrá el beneficio añadido de hacer nuestra secuencia de chat expect más fácil de entender. Al modificar nuestro ejempo, nuestro fichero `dial-c3po` tendrá ahora un aspecto como éste:

```
''      ATZ
OK      ATDT318714
CONNECT ''
ogin:   ppp
word:   GaGariN
```

Cuando usamos un script de chat de esta manera, la cadena que esperamos recibir está en la izquierda y la respuesta que devolveremos está en la derecha. Presentándolas así son mucho más fáciles de leer y entender.

Por lo tanto la invocación completa al **pppd** tendrá ahora un aspecto como éste:

```
# pppd connect "chat -f dial-c3po" /dev/ttyS3 38400 -detach \
    crtscts módem defaultroute
```

Además de la opción connect que se refiere al script de llamada, hemos añadido dos opciones más a la línea de órdenes: detach, que le indica al **pppd** que no se separe de la consola y se convierta en un proceso en segundo plano. La palabra modem activa algunas acciones específicas para módem sobre el dispositivo serie, como colgar la línea antes y después de la llamada. Si no utiliza esta opción, **pppd** no se preocupará de la línea DCD del puerto, y por lo tanto no podrá detectar si el extremo remoto cuelga de forma imprevista.

Los ejemplos anteriores eran bastante simples; **chat** permite el uso de scripts mucho más complejos. Una característica muy útil es la capacidad de especificar cadenas frente a las cuales parar chat con un error. Unas cadenas típicas para parar pueden ser mensajes como BUSY o NO CARRIER, que son los que su módem produce cuando el número al que llama comunica o cuando la línea no tiene tono. Para hacer que **chat** las reconozca inmediatamente en vez de esperar, puede introducirlas al principio del script utilizando la opción ABORT:

```
$ chat -v ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ...
```

Igualmente, puede variar el valor del tiempo de espera para algunas partes de los scripts de chat insertando opciones TIMEOUT.

Algunas veces, también querrá disponer de algún tipo de ejecución condicional de algunas partes del script de chat. Por ejemplo, cuando reciba el mensaje de petición de registro (login) desde el extremo remoto, puede que quiera enviar un BREAK, o un retorno de carro. Puede conseguir esto añadiendo un sub−script a la parte de la cadena esperada. Consiste en una secuencia de cadenas de envío y esperadas, de la misma forma que el script en su totalidad, pero separadas por guiones. El sub−script es ejecutado desde el momento en que la cadena esperada a la que están ligados no es recibida a tiempo. Para este ejemplo, modificaríamos el script de chat de la siguiente manera:

```
ogin:-BREAK-ogin: ppp ssword: GaGariN
```

Ahora, cuando **chat** no recibe el mensaje de login del sistema remoto, se ejecuta el sub−script enviando un BREAK y esperando de nuevo por el mensaje de login. Si ahora ya aparece, el script continúa como usualmente y si no, termina con un error.

8.5. Opciones de Configuración IP

El IPCP se utiliza para negociar varios parámetros IP a la hora de configurar la conexión. Normalmente, cada extremo de comunicación puede enviar un Paquete de Petición de Configuración IPCP, indicando qué valores quiere cambiar de los que vienen predeterminados, y a qué valor. Tras la recepción, el extremo remoto inspecciona cada opción sucesivamente, y responde que la acepta, o la rechaza.

pppd le da gran control sobre qué opciones intentará negociar el IPCP. Puede ajustar esto a través de varias opciones en la línea de órdenes de las que hablamos a continuación.

8.5.1. Eligiendo Direcciones IP

Todos los interfaces IP requieren de direcciones IP asignadas a ellos; Un dispositivo PPP siempre tiene una dirección IP. El conjunto de protocolos PPP provee un mecanismo que permite la asignación automática de direcciones IP a interfaces PPP. Es posible para el programa PPP en un extremo del enlace punto a punto asignar una dirección IP al extremo opuesto para que la use, o que cada uno use la suya.

Algunos servidores PPP que sirven a muchos clientes asignan direcciones dinámicamente: las direcciones son asignadas a los sistemas sólo cuando llaman, y son reclamadas de nuevo una vez que se desconecta. Esto permite que el número de direcciones IP requeridas esté limitado al número de líneas conectadas. Mientras la limitación es conveniente para quienes gestionan los servidoresde marcado PPP, es a menudo menos conveniente para los usuarios que están intentando conectar. Ya discutimos la forma en la que los nombres de nodo son transformados en direccionnes IP mediante una base de datos en Capítulo 6. Para permitir que la gente se conecte a su nodo, ellos deben saber su dirección IP o el nombre del nodo asociado a ella. Si usted es usuario de un servicio PPP que le asigna una dirección IP de forma dinámica, este conocimiento es difícil sin permitir de alguna manera a la base de datos DNS que se actualice después de que le es asignada la dirección IP. Este tipo de sistemas existen, pero nosotros no los cubriremos en detalle aquí; en cambio, miraremos hacia una situación más preferible, la cual implica que sea capaz de utilizar la misma dirección IP cada vez que se establece su conexión de red. [73]

En el ejemplo anterior, hacíamos que **pppd** llamase a c3po y estableciera una conexión IP. No nos preocupábamos de elegir una dirección IP particular en ninguno de los extremos de la conexión. En vez de ello, tomábamos la dirección de vlager como la dirección IP local, y dejábamos a c3po darse su propia dirección. El pppd soporta diferentes alternativas a esta aproximación.

Para pedir direcciones particulares, normalmente dé a **pppd** la siguiente opción:

```
local_addr:remote_addr
```

local_addr y *remote_addr* pueden ser especificados tanto en notación de cuaternas numéricas o como nombres de nodo.[74] Esta opción hace a **pppd** intentar usar la primera dirección como su propia dirección IP, y la segunda como la de su compañero. Si el compañero rechaza alguna de ellas durante la negociación IPCP, no se establecerá ninguna conexión IP.[75]

Si usted está llamando a un servidor y espera que éste le asigne una dirección IP, debe asegurarse de que **pppd** no intenta negociar una por sí mismo. Para hacer esto, use la opción `noipdefault` y deje la opción *local_addr* en blanco. La opción `noipdefault` evitará que **pppd** intente usar la dirección IP asociada al nombre de ordenador como la dirección local.

Si sólo quiere establecer la dirección local, y aceptar cualquier dirección que utilice el compañero, simplemente deseche la parte *remote_addr*. Por ejemplo, para hacer a vlager usar la dirección IP 130.83.4.27 en vez de la suya propia, le escribiría `130.83.4.27:` en la línea de orden. De forma similar, para establecer la dirección remota únicamente, dejaría el campo de la *dir_local* en blanco. Por omisión, **pppd** utilizará entonces la dirección asociada al nombre de su ordenador.

8.5.2. Encaminamiento a través de un enlace PPP

Tras configurar el interfaz de red, **pppd** preparará un encaminamiento que solamente le sirve para comunicarse con el otro extremo. Si el ordenador remoto está en una red de área local, seguramente usted deseará conectar también con los ordenadores que están "detrás" de él; para eso, se ha de configurar un encaminamiento de red adecuado.

Ya hemos visto antes que se puede pedir a **pppd** que configure el encaminamiento predeterminado utilizando la opción `defaultroute`. Esta opción es muy útil si el servidor PPP al que llama va a actuar como su pasarela a Internet.

El caso contrario, cuando su sistema actúa como una pasarela para un sólo ordenador, es también relativamente fácil de llevar a cabo. Por ejemplo, imagine a algún empleado de la Cervecera Virtual cuyo ordenador de casa se llama oneshot. Cuando esté conectando a vlager a través de PPP, él utiliza una dirección de la subred de la Cervecera. Podremos dar a **pppd** del ordenador vlager la opción `proxyarp`, que instalará una entrada proxy−ARP para el ordenador oneshot. Esto hará que oneshot sea automáticamente accesible desde todos los ordenadores de la Cervecera y la Vinatera.

De cualquier manera, las cosas no son siempre tan fáciles como esto, por ejemplo cuando intentamos unir dos redes de área local. Esto requiere normalmente el añadir una ruta de red especifica, porque estas redes tendrán ya sus propios encaminamientos por defecto. Por otra parte, el tener los dos extremos de comunicación utilizando la conexión PPP como encaminamiento por defecto generaría un ciclo sin fin, donde los paquetes con destinos desconocidos rebotarían entre los dos ordenadores hasta que su tiempo de vida (TTL) expirase.

Pongamos un ejemplo: suponga que la Cervecera Virtual abre una sucursal en alguna otra ciudad. La sucursal utiliza su propia red Ethernet utilizando el número de red IP 172.16.3.0, que es la subred 3 de la red de clase B de la Cervecera. Quieren conectarse a la red Ethernet principal de la Cervecera a través de PPP para actualizar las bases de datos de clientes, etc. De nuevo, vlager actuará como pasarela; la otra máquina se llama vbourbon y tiene una dirección IP de 172.16.3.1. Esta red está ilustrada en Figura A−2 en Apéndice A.

Cuando vbourbon conecta a vlager, hará que el punto de encaminamiento predeterminado sea vlager, como es habitual. En vlager, de todas formas, tendremos que instalar un encaminamiento de red para la subred 3 que vaya a través de vbourbon. Podriamos hacer esto manualmente usando la orden **route** después de que el enlace PPP sea establecido, pero esta no es una solución muy práctica. Afortunadamente, podemos configurar la ruta automáticamente utilizando una característica de **pppd** de la que no hemos hablado hasta ahora − la orden **ip−up**. Es un script de shell situado en `/etc/ppp` que se ejecuta después de que el interfaz PPP ha sido configurado. Cuando está presente, se le llama con los siguientes parámetros:

```
    ip−up interface dispositivo velocidad dir_local dir_remota
```

La tabla siguiente resume el significado de cada uno de los argumentos (en la primera columna, se muestra el número usado por el script de shell para referirse a cada argumento):

Argumento	Nombre	Propósito
1	*`interfaz`*	Interfaz de red usado, e.g., `ppp0`
2	*`dispositivo`*	dispositivo es la ruta al dispositivo serie utilizado,(`/dev/tty` si se utiliza la salida y entrada estándar)
3	*`velocidad`*	La velocidad del dispositivo en bits por segundo.
4	*`dir_local`*	La dirección IP del extremo local del enlace en notación de cuarteto.
5	*`dir_remota`*	La dirección IP del extremo remoto de la conexión

En nuestro caso, el script **ip−up** puede contener el siguiente fragmento de código:[76]

```
#!/bin/sh
case $5 in
172.16.3.1)           # este es vbourbon
        route add −net 172.16.3.0 gw 172.16.3.1;;
...
esac
exit 0
```

De una forma análoga,**/etc/ppp/ip−down** se utiliza para deshacer todas las acciones de **ip−up** después de que la conexión PPP ha sido cortada. Asi en nuestro script **/etc/ppp/ip−down** tendremos una orden **route** que elimine la ruta que creamos con el script **/etc/ppp/ip−up**.

A pesar de todo, la tabla de encaminamiento aún no está completa. Hemos configurado las entradas de la tabla de encaminamiento para los dos ordenadores con PPP, pero hasta ahora, todos los demás ordenadores de las dos redes no saben nada sobre la conexión PPP. Esto no es un gran problema si todos los ordenadores de la sucursal tienen su encaminamiento predeterminado encaminado a vbourbon, y todos los ordenadores de la Cervecera encaminan hacia vlager por omisión. Si éste no fuera el caso, su única posibilidad normalmente será usar un demonio de encaminamiento como **gated**. Tras crear el encaminamiento de la red en vlager, el demonio de encaminamiento pasará el nuevo encaminamiento a todos los ordenadores de las redes dependientes de ésta.

8.6. Opciones de Control de Enlace

Anteriormente, ya hemos tratado sobre LCP, el protocolo de control de enlace (Link Control Protocol), que se utiliza para negociar las características de la conexión y comprobarla.

Las dos opciones más importantes que pueden ser negociadas por el LCP son la *unidad máxima de recepción* (MRU) y el *mapa de caracteres de control asíncronos*. También hay varias opciones más de configuración LCP, pero son demasiado específicas como para comentarlas aquí. Eche un vistazo a la RFC 1548 para ver una descripción de éstas.

El mapa de caracteres de control asíncronos, también conocido como el *mapa asíncrono*, se usa en enlaces asíncronos, como las líneas telefónicas, para identificar los caracteres de control que deben de ser reemplazados por una secuencia específica de dos caracteres, para evitar que sean interpretados por el equipamiento utilizado para establecer el enlace. Por ejemplo, puede que quiera evitar los caracteres XON y XOFF utilizados con el control de flujo hardware activado, pues algún módem mal configurado puede parar hasta que reciba un XOFF. Otro candidato puede ser Ctrl−] (el carácter de escape del **telnet**). PPP le permite obviar/rehuir de cualquiera de los caracteres con códigos ASCII comprendidos entre 0 y 31 especificándolos en el mapa asíncrono.

El mapa asíncrono (async map) es un mapa de bits de 32 bits de ancho, y cuyo bit menos significativo corresponde al carácter ASCII NUL, y cuyo bit mas significativo corresponde al ASCII 31. Estos 32 caracteres ASCII son los caracteres de control. Si un bit se pone a 1, indica que el carácter correspondiente debe de ser "escapado" antes de ser enviado a través de la conexión.

Para decir al otro ordenador que no tiene que rehuir de todos los caracteres de control sino sólo de algunos, puede establecer un nuevo mapa asíncrono al **pppd** utilizando la opción **asyncmap**. Por ejemplo, si sólo `^S` y `^Q` (los códigos ASCII 17 y 19, normalmente utilizados para XON y XOFF) deben de ser "escapados", utilice la siguiente opción:

```
asyncmap 0x000A0000
```

Mientras sepa convertir binario a hexadecimal la conversión es simple. Coloque 32 bits enfrente de usted. El bit más a la derecha corresponde al ASCII 00(NULL), y el de más a la izquierda al ASCII 32 decimal. Establezca los bits que corresponden a los caracteres que quiera "escapar" a uno, y el resto a 0. Para convertir eso al número en hexadecimal que el **pppd** espera, simplemente coja cada grupo de 4 bits y conviértalos en hexadecimal. Debería terminar con ocho figuras en hexadecimal. Póngalos todos juntos en cadena y antepóngale 0x para mostrar que es un número hexadecimal, y habrá terminado.

Inicialmente, el mapa asíncrono se establece como `0xffffffff` lo que significa que todos los caracteres de control serán "escapados". De partida esto es seguro, pero normalmente es más de lo que necesita. Cada carácter que aparece en el mapa asíncrono produce dos caracteres que son transmitidos a través del enlace, así al introducir estos caracteres de escape se produce un incremento de la utilización del enlace y la correspondiente reducción del rendimiento.

En la mayoria de las circunstancias, un mapa asíncrono de `0x0` funcionará correctamente. No se producen caracteres de escape.

La unidad máxima de recepción, o MRU, señaliza al otro extremo el tamaño máximo de las tramas HDLC que queremos recibir. Aunque esto puede que le recuerde al valor de la MTU (unidad máxima de transferencia), tienen poco en común. El MTU es un parámetro del dispositivo de red del núcleo, y describe el tamaño máximo de la trama que el interfaz es capaz de soportar. El MRU es más bien un consejo al ordenador remoto para que no genere ninguna trama más grande que la MRU; no obstante, el interfaz ha de ser capaz de recibir tramas de hasta 1500 bytes.

Por lo tanto, elegir un MRU no es tanto una cuestión de que es capaz de transmitir la conexión, sino de cómo conseguir el mejor rendimiento. Si va a usar la conexión para aplicaciones interactivas, el poner en el MRU valores tan bajos como 296 es una buena idea, de forma que un paquete ocasional mayor (digamos, de una sesión de FTP) no haga a su cursor saltar.. Para decir al **pppd** que pida un MRU de 296, pondría la opción `mru 296`. Las MRUs pequeñas, de todas maneras, sólo tienen sentido si no tiene la compresión de cabecera VJ desactivada (esta activada por omisión), de otra manera desaprovechará una gran cantidad de su ancho de banda sólo transportando la cavecera IP de cada datagrama.

El **pppd** también entiende un par de opciones LCP que configuran el comportamiento general del proceso de negociación, como es el máximo número de peticiones de configuración que pueden ser intercambiadas antes de que se corte la conexión. A menos que sepa exactamente lo que esta haciendo, deberá dejar este valor fijo.

Finalmente, hay dos opciones que se aplican a los mensajes de eco del LCP. El PPP define dos mensajes,*Petición de Eco* y *Respuesta de Eco*. El **pppd** usa esta característica para comprobar si la conexión esta aún operativa. Puede habilitarla utilizando la opción `lcp-echo-interval` junto con el tiempo en segundos. Si no se reciben tramas del ordenador remoto en este intervalo, el **pppd** genera una Petición de Eco, y espera a que el compañero devuelva una Respuesta de Eco. Si el compañero no produce una respuesta, la conexión es cortada después de que se hayan enviado un cierto número de peticiones. Este número puede ser establecido utilizando la opción `lcp-echo-failure`. predeterminado, esta característica también esta desactivada.

8.7. Consideraciones de Seguridad General

Un demonio de PPP mal configurado puede ser un peligroso agujero en la seguridad. Es equivalente a dejar a cualquiera enganchar su máquina a su red Ethernet (y eso es muy malo). En esta sección, discutiremos algunas medidas que deberían hacer su configuración de PPP segura.

> **Nota:** Configurar el dispositivo de red y la tabla de encaminamiento requiere los privilegios de root. Normalmente resolverá esto ejecutando **pppd** como setuid de root. Sin embargo, **pppd** permite a los usuarios establecer varias opciones relacionadas con la seguridad.

Para protegerse contra cualquier ataque que pueda lanzar algún usuario manipulando estas opciones, se sugiere que establezca un par de valores predeterminado en el fichero global `/etc/ppp/options`, tal como los mostrados en el fichero de ejemplo en Sección 8.3, al principio de este capítulo. Algunos de ellos, como los de las opciones de autentificación, no pueden ser después modificados por el usuario, así que proporcionan una razonable protección contra las manipulaciones. Una opción importante que proteger es la opción connect. Si pretende permitir a usuarios no root invocar **pppd** para conectar a Internet, debería siempre añadir las opciones `connect` y `noauth` al fichero de opciones globales `/etc/ppp/options`. Si no hace esto, los usuarios serán capaces de ejecutar ordenes arbitrarias con privilegios de `root` especificandolas como argumento del **pppd** en la orden de connect o en sus ficheros de opciones personales.

Otra buena idea es restringir qué usuarios pueden ejecutar **pppd** creando un grupo en `/etc/group` e introducir sólo aquellos usuarios que usted desea que tengan la habilidad de ejecutar el demonio PPP. Despues debería cambiar la propiedad de grupo del demonio **pppd** a ese grupo y quitar los privilegios de ejecución globales. Para hacer esto, asumiendo que ha llamado a su grupo dialout, podria usar algo como esto:

```
# chown root /usr/sbin/pppd
# chgrp dialout /usr/sbin/pppd
# chmod 4750 /usr/sbin/pppd
```

Por supuesto, también tiene que protegerse de los sistemas con los que habla PPP. Para evitar que otros ordenadores puedan hacerse pasar por quien no son, debe utilizar siempre algún tipo de autentificación con el otro extremo de la comunicación. Además, no debería permitir a ordenadores desconocidos usar cualquier dirección IP que elijan, sino restringirlas a unas pocas. La siguiente sección tratará sobre estos asuntos.

8.8. Autentificación con PPP

Con el PPP, cada sistema puede obligar al otro ordenador a identificarse usando uno de los dos protocolos de autentificación disponibles. Estos son el Protocolo de Autentificación por Contraseña (PAP), y el Protocolo de Autentificación por Reto (CHAP). Cuando se establece una conexión, cada extremo puede pedir al otro que se autentifique, independientemente de que sea el llamante o el llamado. Más adelante, utilizaré relajadamente cliente y servidor cuando quiera distinguir entre el sistema autentificado y el autentificador. Un demonio PPP puede pedir a la otra máquina autentificación enviando otra petición más de configuración de LCP indicando el protocolo de autentificación deseado.

8.8.1. PAP frente a CHAP

PAP, que es utilizado por muchos proveedores de Internet (ISP), funciona básicamente de la misma forma que el procedimiento normal de registro. El cliente se autentifica a sí mismo enviando un nombre de usuario y una contraseña (opcionalmente encriptada) al servidor, la cual es comparada por el servidor con su base de datos de claves o secrets. [77] Esta técnica es vulnerable a los intrusos que pueden intentar obtener la contraseña escuchando en una línea serie y a otros que hagan sucesivos intentos de ataque por el método de prueba y error.

CHAP no tiene estos defectos. Con CHAP, el autentificador (i.e. el servidor) envía una cadena de reto generada aleatoriamente al cliente, junto a su nombre de ordenador. El cliente utiliza el nombre del ordenador para buscar la clave apropiada, la combina con el reto, y encripta la cadena utilizando una función de codificación de un solo sentido. El resultado es devuelto al servidor junto con el nombre del ordenador cliente. El servidor realiza ahora la misma computación, y advierte al cliente si obtiene el mismo resultado.

Otra característica de CHAP es que no solicita autentificación al cliente solamente al comienzo de la sesión, sino que envía retos a intervalos regulares para asegurarse de que el cliente no ha sido reemplazado por un intruso, por ejemplo cambiando la línea telefónica, o debido a una configuración errónea del módem que causa que el demonio PPP no se perciva que la llamada original de teléfono se ha cortado y algún otro se ha conectado.

El **pppd** mantiene las claves secretas para el CHAP y el PAP en dos ficheros separados, llamados `/etc/ppp/pap-secrets` y `/etc/ppp/chap-secrets` respectivamente. Si introduce un ordenador remoto en alguno de los dos ficheros, tendrá un buen control de cual de los protocolos CHAP o PAP se utilizará para autentificarnos con él y viceversa.

Por omisión, **pppd** no pide autentificación al ordenador remoto, pero aceptará el autentificarse a sí mismo cuando se lo pida el ordenador remoto. Como CHAP es mucho más fuerte que PAP, el **pppd** intenta usar el anterior siempre que es posible. Si el otro ordenador no lo acepta, o **pppd** no encuentra una clave CHAP para el sistema remoto en su fichero `chap-secrets`, cambia al PAP. Si tampoco tiene clave PAP para su compañero, renunciará a autentificarse. Como consecuencia de esto, se cerrará la conexión.

Este comportamiento puede ser modificado de varias formas. Por ejemplo, cuando se añade la palabra `auth`, **pppd** solicitará al otro ordenador que se autentifique. **pppd** aceptará el uso de CHAP o PAP para ello, siempre y cuando tenga una clave para su compañero en su base de datos CHAP o PAP respectivamente. Hay otras opciones para activar o no un determinado protocolo de autentificación, pero no las describiré aquí. Puede leer la página de manual del pppd(8) para más detalles.

Si todos los sistemas con los que conversa PPP están de acuerdo en autentificarse con usted, debería poner la opción `auth` en el fichero global `/etc/ppp/options` y definir contraseñas para cada sistema en el fichero `chap-secrets`. Si un sistema no acepta CHAP, añada una entrada para él al fichero `pap-secrets`. De esta forma, puede asegurarse de que ningún sistema sin autentificar se conecta a su ordenador.

Las dos secciones siguientes hablan sobre los dos ficheros de claves de PPP, `pap-secrets` y `chap-secrets`. Están situados en `/etc/ppp` y contienen tripletes de clientes, servidores y contraseñas, seguidas opcionalmente por una lista de direcciones IP. La interpretación de los campos de servidor y cliente es distinta en CHAP y PAP, y también depende de si nos autentificamos nosotros con el otro ordenador, o si solicitamos al servidor que se autentifique con nosotros.

8.8.2. El fichero de claves CHAP

Cuando tiene que autentificarse con algún servidor utilizando CHAP, **pppd** busca en el fichero `chap-secrets` una entrada cuyo campo de cliente sea igual al nombre del ordenador local, y cuyo campo de servidor sea igual al nombre del ordenador remoto enviado en el reto del CHAP. Cuando solicita a la otra máquina que se autentifique, los roles son simplemente al revés: el pppd entonces buscará una entrada que tenga el campo de cliente igual al nombre del ordenador remoto (enviado en la respuesta del CHAP del cliente), y el campo de servidor igual al nombre del ordenador local.

El siguiente es un fichero de ejemplo de `chap-secrets` para vlager:[78]

```
# CHAP secrets for vlager.vbrew.com
#
# client          server            secret                addrs
#---------------------------------------------------------------------
vlager.vbrew.com  c3po.lucas.com    "Use The Source Luke" vlager.vbrew.com
c3po.lucas.com    vlager.vbrew.com "arttoo! arttoo!"     c3po.lucas.com
*                 vlager.vbrew.com "TuXdrinksVicBitter"  pub.vbrew.com
```

Cuando se intenta establecer una conexión PPP con c3po, c3po pide a vlager que se autentifique usando el CHAP mediante el envío de un reto de CHAP. El demonio **pppd** entonces examina `chap-secrets` buscando una entrada cuyo campo de cliente sea igual a vlager.vbrew.com y el campo de servidor sea c3po.lucas.com, y encuentra la primera línea mostrada en el ejemplo.[79] Entonces produce la respuesta de CHAP a partir de la cadena del reto y la clave (`Use The Source Luke`), y la envía de vuelta a c3po.

Al mismo tiempo, **pppd** produce un reto del CHAP para c3po, conteniendo una única cadena de reto y su nombre de ordenador completo vlager.vbrew.com. c3po construye una respuesta de CHAP de la manera que acabamos de decir, y se la devuelve a vlager. **pppd** extrae ahora el nombre del cliente (c3po.vbrew.com) de la respuesta, y busca en el fichero `chap-secrets` una línea que tenga c3po como cliente y vlager como servidor. La segunda línea se corresponde con esto, así que el **pppd** combina el reto del CHAP y la clave arttoo! arttoo!, las encripta, y compara el resultado con la respuesta de CHAP de c3po.

El cuarto campo opcional lista las direcciones IP que son aceptables por los clientes nombrados en el primer campo. Las direcciones pueden ser dadas en notación cuaternaria o como nombres de ordenador que son resueltos posteriormente. Por ejemplo, si c3po solicita usar una dirección IP que no estáa en esta lista durante la negociación IPCP, la petición será rechazada, e IPCP se desconectará. En el fichero de ejemplo anterior, c3po está limitado a poder usar sólo su propia dirección. Si el campo de dirección está vacío, se permitirá cualquier dirección; un valor de − evita el uso de una cierta dirección IP con un cliente.

La tercera línea del fichero `chap-secrets` de prueba, permite a cualquier ordenador establecer un enlace PPP con vlager, pues si aparece la expresión `*` en los campos de cliente o servidor, será valido cualquier nombre. El único requisito es que sepa la clave, y utiliza la dirección de pub.vbrew.com. Pueden aparecer perfectamente entradas con comodines en los nombres en cualquier lugar del fichero de claves, pues el **pppd** siempre utilizará la entrada más específica que pueda ser aplicada a un par cliente/servidor.

Hay algunas cosas que decir sobre la manera en que **pppd** encuentra los nombres de ordenadores que busca en el fichero de claves. Como se explicó anteriormente, el nombre del ordenador remoto es siempre proporcionado por el otro ordenador en el paquete de reto o respuesta del CHAP. El nombre del ordenador local será obtenido por omisión llamando a la función `gethostname(2)`. Si ha configurado el nombre del sistema como el nombre del ordenador sin calificar, entonces tendrá que dar al **pppd** el nombre del dominio a añadir usando la opción `domain`:

```
# pppd  domain vbrew.com
```

Esto añadirá el nombre del dominio de la Cervecera a vlager para todas las actividades relacionadas con la autentificación. Otras opciones que modifican la idea que tiene el **pppd** del nombre del ordenador local son `usehostname` y `name`. Cuando da la dirección IP local en la línea de comando usando ***`local`*****:*****`remote`*** y *`local`* es un nombre en vez de un cuarteto numérico, el **pppd** utilizará éste como el nombre local.

8.8.3. El fichero de claves PAP

El fichero de claves PAP es muy similar al utilizado por CHAP. Los dos primeros campos siempre contienen un nombre de usuario y un nombre de servidor; el tercero alberga la clave PAP. Cuando el sistema remoto envía una petición de autentificación, **pppd** usa la entrada en la que el campo de servidor es igual al nombre del ordenador local, y el campo de usuario igual al nombre de usuario enviado en la petición. Cuando se autentifica a sí mismo al otro ordenador, **pppd** toma la clave a enviar de la línea con el nombre de usuario igual al nombre del usuario local, y con el campo de servidor igual al nombre del ordenador remoto.

Un fichero de claves PAP sencillo puede parecerse a éste:

```
# /etc/ppp/pap-secrets
#
# user          server          secret          addrs
vlager-pap      c3po            cresspahl       vlager.vbrew.com
c3po            vlager          DonaldGNUth     c3po.lucas.com
```

La primera línea se usa para autentificarnos a nosotros mismos cuando hablemos con c3po. La segunda línea describe como un usuario llamado c3po tiene que autentificarse con nosotros.

El nombre vlager−pap de la primera columna es el nombre de usuario que nosotros mandamos a c3po. POr omisión,**pppd** tomará el nombre del ordenador local como el nombre de usuario, pero también se puede especificar un nombre diferente usando la opción `user`, seguida del nombre deseado.

Para escoger una de las entradas del fichero `pap-secrets` para la autentificación con el compañero, **pppd** tiene que saber el nombre del ordenador remoto. Como no tiene manera de averiguarlo, tiene que especificarlo en la línea de órdenes usando la palabra `remotename`, seguida por el nombre del ordenador remoto. Por ejemplo, para usar la entrada comentada anteriormente para la autentificación con c3po, tenemos que añadir la siguiente opción a la línea de órdenes del **pppd**:

```
# pppd ... remotename c3po user vlager-pap
```

En el cuarto campo (y todos los siguientes), puede especificar qué direcciones IP están permitidas para ese ordenador particular, de la misma forma que en el fichero de claves CHAP. El otro ordenador solo podrá pedir direcciones de esa lista. En el fichero de ejemplo, la entrada que c3po usará cuando llame a la linea donde c3po es el cliente, le permitirá usar su IP auténtica y no otra.

Dése cuenta de que PAP es un método de autentificación bastante débil, y se recomienda utilizar el CHAP siempre que sea posible. Por eso, no explicaremos el PAP en gran profundidad aquí; si esta interesado en utilizar el PAP, encontrará algunas características más de éste comentadas en la página del manual del `pppd(8)`.

8.9. Depurando su configuración de PPP

Por omisión, **pppd** envia cualquier mensaje de advertencia o error al demonio **syslog**. Tiene que añadir una entrada a `syslog.conf` que redirija estos mensajes a un fichero o incluso a la consola; de lo contrario, **syslog** simplemente los descarta. La entrada siguiente envia todos los mensajes a `/var/log/ppp-log/`:

```
daemon.*                /var/log/ppp-log
```

Si su configuración de PPP no funciona bien, debería mirar en este fichero de log. Si los mensajes no ayudan, también puede activar información de depuración extra usando la opción `debug`. Esta salida fuerza al **pppd** a enviar a los ficheros de log los contenidos de todos los paquetes de control enviados o recibidos a **syslog**. Todos los mensajes van entonces al demonio

Finalmente, la manera más drástica de localizar un problema es activar la depuración a nivel de núcleo invocando al **pppd** con la opción `kdebug`. Es seguida de un argumento numérico que es la suma de los valores siguientes: 1 para los mensajes de depuración genéricos, 2 para mostrar los contenidos de las tramas HDLC entrantes, y 4 para hacer que el controlador muestre las tramas HDLC salientes. Para capturar mensajes de depuración del núcleo, puede ejecutar el demonio **syslogd** que lee el fichero `/proc/kmsg`, o bien el demonio **klogd**. Cualquiera de ellos dirige los mensajes de depuración del núcleo al **syslog** del núcleo.

8.10. Configuraciones avanzadas de PPP

Mientras que configurar PPP para conectar a una red como Internet es la más común de sus aplicaciones, hay algunos de vosotros que tenéis requerimientos más avanzados. En esta sección hablaremos sobre algunas de las configuraciones más avanzadas que son posibles con PPP bajo GNU/Linux.

8.10.1. Servidor PPP

Hacer funcionar el **pppd** como servidor es solo cuestión de configurar un dispositivo terminal serie para que invoque al **pppd** con las opciones apropiadas cuando una llamada entrante es recibida. Una manera de hacer esto es crear una cuenta especial, digamos ppp, y asociarle un script o programa como shell de entrada que llame al **pppd** con esas opciones. De forma alternativa, si quiere soportar autentificacion PAP o CHAP, puede usar el programa **mgetty** para soportar su módem y explotar su característica /autoPPP/.

Para configurar un servidor usando el método de registro (login), añada una línea similar a la siguiente a su fichero `/etc/passwd`: [80]

```
ppp:x:500:200:Public PPP Account:/tmp:/etc/ppp/ppplogin
```

Si su sistema soporta shadow passwords, tambien necesita añadir una entrada al fichero `/etc/shadow`:

```
ppp:!:10913:0:99999:7:::
```

Por supuesto, el UID y GID que utilice dependen de que usuario desea que sea dueño de la conexión, y de cómo lo haya creado. Además tiene que establecer la contraseña para la cuenta mencionada usando la orden **passwd**.

El script **ppplogin** podría tener este aspecto:

```
#!/bin/sh
# ppplogin − script to fire up pppd on login
mesg n
stty −echo
exec pppd −detach silent modem crtscts
```

La orden **mesg** deshabilita la opción que permite que otros usuarios puedan escribir en el terminal (tty) usado utilizando, por ejemplo, la orden **write**. La orden **stty** desactiva el eco de caracteres. Esto es necesario, pues de otra forma todo lo que el otro ordenador envíe le será devuelto a modo de eco. La opción del **pppd** más importante de las incluidas en el script es detach, porque evita que el **pppd** se separe de la terminal controlada. Si no especificásemos esta opción, se iría a segundo plano, haciendo que el script del shell terminase. Esto provocaría que la línea serie colgase y se perdiera la conexión. La opción **silent** hace que el **pppd** espere hasta recibir un paquete del sistema llamante antes de comenzar a enviar. Esto evita la aparición de timeouts al transmitir cuando el sistema que nos llama es lento en lanzar su cliente PPP. La opción modem hace al **pppd** vigilar la línea DTR para ver si el otro sistema ha colgado. Siempre deberia activar esta opción cuando use **pppd** con un modem. La opción `crtscts` activa el control de flujo por hardware.

Además de estas opciones, se puede forzar alguna clase de autentificación, por ejemplo especificando `auth` en la línea de órdenes de **pppd**, o en el fichero de opciones globales. La página del manual también habla sobre opciones más especíificas para activar o desactivar los protocolos de autentificación individuales.

Si desea usar **mgetty**, todo lo que tiene que hacer es configurar **mgetty** para que soporte el dispositivo serie al que su módem está conectado (vea Sección 4.6.1 para más detalles), configurar **pppd** bien para autentificación PAP o CHAP con las opciones apropiadas en sus ficheros `options`, y finalmente, añadir una sección similar a la siguiente a su fichero `/etc/mgetty/login.config`:

```
# Configura mgetty para automaticamente detectar llamandas entrantes e invocar
# al demonio pppd para que se haga cargo de la conexión.
#
/AutoPPP/ −     ppp   /usr/sbin/pppd auth −chap +pap login
```

El primer campo es una parte especial mágica usada para detectar que la llamada entrante es una de tipo PPP. No debe cambiar el aspecto de esta cadena; es sensitivo a minúusculas y mayúusculas. La tercera columna en el nombre de usuario que aparece en en el listado de **who** cuando alquien a entrado en el sistema. El resto de la línea es el comando a invocar. En nuestro ejemplo, nos hemos asegurado de que la autentificación por PAP es requerida, deshabilitado CHAP, y especificado que el fichero del sistema `passwd` debe ser usado para autentificar usuarios. Esto es probablemente parecido a lo que usted querrá. Recuerde, puede especificar las opciones en el fichero `options` o en la linea de comandos si lo prefiere.

Esto es una pequeña lista de comprobación de los pasos que debe realizar y la secuencia que debe seguir para tener funcionando en su máquina un servidor PPP. Asegúrese de que cada paso funciona correctamente antes de pasar al siguiente:

1. Configure el módem para que funcione en modo de auto−respuesta. En los módems compatibles Hayes, esto se realiza mediante la orden `ATS0=3`. Si va a utilizar el demonio **mgetty**, esto no será necesario.

2. Configure el dispositivo serie con una orden tipo **getty** para que responda a las llamadas entrantes. Una variante comúnmente usada de **getty** es **mgetty**.

3.

Considere la autentificación. ¿Como se autentificarán con usted los clientes, usando PAP, CHAP, o el registro del sistema?

4. Configure **pppd** como servidor tal como se describe en esta sección.

5. Considere el encaminamiento. ¿Necesitará proporcionar una ruta de red a los clientes? El encaminamiento puede realizarse usando el script `ip-up`.

8.10.2. Llamada bajo demanda

Cuando hay tráfico IP para ser transportado a través del enlace, *la llamada en demanda* provoca la llamada de su modem y el establecimiento de una conexión con un host remoto. La llamada en demanda resulta útil cuando no puede dejar su línea telefónica permanentemente conectada a su proveedor de Internet. Por ejemplo, puede que tenga que pagar llamadas locales por tiempo de uso, así le resultará más barato tener la conexión establecida sólo cuando lo necesite y desconectada cuando no está usando Internet.

Las soluciones GNU/Linux tradicionales han usado la orden **diald**, que funcionaba bien pero era algo complicado de configurar. Las versiones 2.3.0 y posteriores del demonio PPP tienen incluído el soporte de llamada bajo demanda y una configuración muy sencilla de realizar. Debe usar un núcleo actual para poder hacer esto. Cualquiera de los núcleos posteriores al 2.0 funcionará bien.

Para configurar **pppd** para llamada bajo demanda, todo lo que tiene que hacer es añadir opciones a su fichero `/etc/options` o a la línea de órdenes de **pppd**. La tabla siguiente resume las opciones relacionadas a la llamada en demanda:

Opción	**Descripción**
`demand`	Esta opción especifica que el enlace PPP debe ser establecido en modo de llamada bajo demanda. El dispositivo de red PPP será creado, pero la orden `connect` no será usada hasta que un datagrama sea transmitido por el host local. Esta opción es obligatoria para que funcione la llamada bajo demanda.
`active-filter` *`expresión`*	Esta opción le permite especificar qué paquetes de datos van a ser considerados como tráfico activo. Cualquier tráfico que cumpla la regla especificada reiniciará el temporizador de la llamada bajo demanda, asegurando que **pppd** espera de nuevo antes de terminar el enlace. La sintaxis del filtro ha sido cogida prestada de la orden **tcpdump** El filtro predeterminado especifica todos los datagramas.
`holdoff` *`n`*	Esta opción le permite especificar la cantidad míma de tiempo, en segundos, que esperar antes de reconectar el enlace si termina. Si la conexión falla mientras **pppd** cree que está en uso activo, será reestablecido después de que este temporizador finalice. Este temporizador no se aplica a las reconexiones producidas por la no transmisión de paquetes.
`idle` *`n`*	Si esta opción está configurada, **pppd** desconectará el enlace cuando este temporizador expire. Los tiempos muertos son especificados en segundos. Cada paquete de datos nuevo activo reseteará el temporizador.

Una configuración simple de llamada bajo demanda podría ser algo como esto:

```
    demand
    holdoff 60
    idle 180
```

Esta configuración activará la llamada bajo demanda, esperará 60 segundos antes de reestablecer un conexión fallida, y terminará el enlace si pasan 180 segundos sin ningún dato activo en el enlace.

8.10.3. llamada persistente

La llamada persistente es lo que la gente que tiene conexiones permanentes a una red querrá usar. Hay una sutil diferencia entre llamada en demanda y llamada persistente. Con la llamada persistente, la conexión es automáticamente establecida tan pronto como el demonio PPP es lanzado, y el aspecto de persistencia viene a cuento siempre que la llamada telefónica que soporta el enlace se interrumpa. La llamada persistente asegura que el enlace está siempre disponible relanzando automáticamete la conexión si ésta se interrumpe.

Usted podría ser afortunado de no tener que pagar por sus llamdas telefónicas; quizás sean locales y gratuitas, o quizás su empresa es quién las paga. La opción de llamada persistente es extremadamente útil en esta situación. Si tiene que pagar por sus llamadas telefónicas, entonces tiene que tener un poco de cuidado. Si paga por sus llamadas telefónicas en base al tiempo que la utiliza, la llamada persistente es algo que casi seguro no es lo que quiere, a menos que esté seguro de que estará usando la conexión constantemente muy amenudo veinticuatro horas al día. Si paga las llamadas, pero no en base al tiempo, necesitará tener cuidado de protegerse de situaciones que puedan causar que el módem llame de forma interminable. El demonio **pppd** provee de una opción que puede ayudar a reducir el efecto de este problema.

Para activar la llamada persistente, debe incluir la opción persist en uno de los ficheros de opciones de **pppd**. Incluir esta opción es todo lo que necesita para tener al **pppd** invocando automáticamente la orden especificada en la opción connect para restablecer la conexión cuando el enlace se interrumpe. Si está preocupado por el remarcado demasiado rápido del módem (en el caso de un fallo del servidor o módem del otro extremo de la conexión), puede usar la opción holdoff para establecer el tiempo mínimo que **pppd** deberá esperar antes de intentar reconectar. Esta opción no resolverá el problema de un fallo y su consecuente gasto en llamadas de teléfono, pero al menos le servirá para reducir el impacto de uno de ellos.

Una configuración típica de llamada persistente podría parecerse a ésta:

```
    persist
    holdoff 600
```

El tiempo de espera es especificado en segundos. En nuestro ejemplo, el **pppd** espera durante cinco minutos antes de rellamar cuando una llamada ha fallado.

Es posible combinar la llamada persistente con la llamada en demanda, usando idle para interrumpir el enlace si ha estado inactivo por un período especificado de tiempo. Dudamos de que sean muchos los usuarios que quieran hacer esto, pero este escenario está descrito brevemente en la página del manual de **pppd**, por si tuviera que buscarlo.

Capítulo 9. Cortafuegos de TCP/IP

La seguridad resulta cada vez más importante tanto para las compañías como para los individuos. Intenet les ha proporcionado una poderosa herramienta para distribuir información entre ellos y para obtener información de otros, pero también les ha expuesto a peligros de los que habían estado exentos hasta entonces. La criminalidad informática, el robo de información y el daño malintencionado constituyen peligros potenciales.

Una persona no autorizada y sin escrúpulos que consiga el acceso al sistema de una computadora puede que averigüe contraseñas del sistema o que se aproveche de los errores y del comportamiento particular de ciertos programas para obtener una cuenta funcional en dicha máquina. Una vez que sea capaz de entrar en la máquina, puede que tenga acceso a información que podría resultar dañina, información tan sensible comercialmente como los planes de negocio, detalles de nuevos proyectos o las bases de datos con información de los clientes. Un daño a este tipo de datos o su modificación puede causar severos retrasos a la compañía.

La forma más segura de evitar daños de tanto alcance consiste en impedir que las personas no autorizadas puedan obtener el acceso a la máquina. Aquí es donde intervienen los cortafuegos [81].

Aviso
La construcción de cortafuegos seguros es todo un arte. Involucra un entendimiento bueno de la tecnología, y no menos importante, requiere de un entendiemiento de la filosofía que hay detrás del diseño de un cortafuegos. En este libro no se cubrirá todo lo que usted necesita; se le recomienda vivamente que realice alguna investigación adicional antes de confiar en un diseño concreto de cortafuegos, lo que incluye cualquiera que se presente aquí.

Existe bastante material sobre configuración y diseño de cortafuegos como para llenar un libro entero, y, por supuesto, ya hay algunos buenos recursos que le podría gustar consultar para incrementar su conocimiento en la materia. Dos de estos recursos son:

'Building Internet Firewalls'
: de D. Chapman y E. Zwicky (O'Reilly). Una guía que explica cómo diseñar e instalar cortafuegos para Unix, Linux y Windows NT, y cómo configurar servicios de Internet en coordinación con los cortafuegos.

'Firewalls and Internet Security'
: de W. Cheswick y S. Bellovin (Addison Wesley). Este libro cubre la filosofía del diseño e implementación de un cortafuegos.

Este capítulo se centrará en aspectos técnicos específicos de Linux. Más adelante se presentará un ejemplo de configuración del cortafuegos que debería servir como punto de partida para su propia configuración, pero, como con todos los asuntos relacionados con la seguridad, no confíe en nadie. Vuelva a comprobar otra vez el diseño, asegúrese de que lo entiende y entonces modifíquelo para ajustarlo a sus requerimientos. Para estar a salvo, esté seguro.

9.1. Métodos de ataque

Como administrador de una red, es importante que usted entienda la naturaleza de los posibles ataques a la seguridad informática. Se describirán brevemente los tipos de ataques más importantes para que usted pueda comprender mejor y de forma más precisa de qué le protegerá un cortafuegos sobre Linux. Debería realizar alguna lectura adicional para asegurarse de que está capacitado para proteger su red de otros tipos de ataques. Aquí están algunos de los más importantes métodos de ataque y las maneras de protegerse contra ellos:

Acceso no autorizado

Esto simplemente quiere decir que personas que no deberían utilizar los servicios de su computadora son capaces de conectarse y utilizarlos. Por ejemplo, personas de fuera de su compañía podrían intentar conectarse a la máquina con las cuentas de su compañía o a su servidor de NFS.

Existen varias formas de evitar este ataque especificando con cuidado quién puede tener acceso a estos servicios. Usted puede evitar el acceso a la red a todo el mundo excepto a los usuarios deseados.

Aprovechamiento de las debilidades conocidas de un programa

Algunos programas y servicios de red no fueron diseñados originalmente teniendo en cuenta una elevada seguridad y son inherentemente vulnerables a los ataques. Los servicios remotos del tipo BSD (rlogin, rexec, etc) constituyen un ejemplo.

La mejor manera de protegerse contra este tipo de ataque consiste en deshabilitar los servicios vulnerables o en encontrar alternativas. Con 'software' de código abierto resulta muchas veces posible reparar sus debilidades.

Denegación de servicio

Los ataques de denegación de servicio causan que el servicio o programa deje de funcionar o impide que otros hagan uso de ese servicio o programa. Estos ataques pueden ser realizados al nivel de red enviando datagramas cuidadosamente preparados y malintencionados de tal forma que puedan causar que las conexiones de red fallen. También pueden realizarse a nivel de aplicación, donde órdenes cuidadosamente construidas se envían contra un programa para tratar que se vuelva muy ocupado o que pare su funcionamiento.

Impedir que el tráfico de red sospechoso alcance sus máquinas y que lleguen órdenes y peticiones de programa sospechosos son las mejores formas de minimizar el riesgo de un ataque de denegación de servicio. Resulta muy útil conocer los detalles del método de ataque, por lo que debería aprender usted mismo todo lo posible de cada tipo nuevo de ataque que se haga público.

Suplantación de identidad [82]

Este tipo de ataque causa que un 'host' o aplicación simule las acciones de otro. Típicamente, el atacante se hace pasar por un 'host' inocente siguiendo el rastro de las direcciones IP contenidas en los paquetes de red. Por ejemplo, un 'exploit' [83] bien documentado del servicio de tipo BSD rlogin puede utilizar esta técnica para simular una conexión de TCP desde otro 'host' prediciendo los números de secuencia de TCP.

Para protegerse contra este tipo de ataque, verifique la autenticidad de los datagramas y órdenes. Evite el encaminamiento de datagramas con direcciones de origen no válidas. Introduzca impredicibilidad en los mecanismos de control de la conexión, como los números de secuencia de TCP y la asignación dinámica de puertos.

'Eavesdropping' [84]

Éste es el método de ataque más simple. Un 'host' se configura para "escuchar" y capturar los datos no destinados a él. Programas de fisgoneo cuidadosamente escritos pueden obtener los nombres de usuario y sus contraseñas a partir de las conexiones de red con ingresos de usuarios en el sistema. Redes de difusión como las de tipo Ethernet son especialmente vulnerables a este tipo de ataques.

Para protegerse contra este tipo de amenazas, evite el uso de tecnologías de red con difusiones e imponga el uso de encriptación de los datos.

Los cortafueros de IP resultan muy útiles para evitar o reducir los accesos no autorizados, los ataques de denegación de servicio a nivel de red, y los ataques de suplantación de identidad. No resultan muy útiles para evitar el aprovechamiento de las debilidades de los servicios de red o programas ni el ['eavesdropping'].

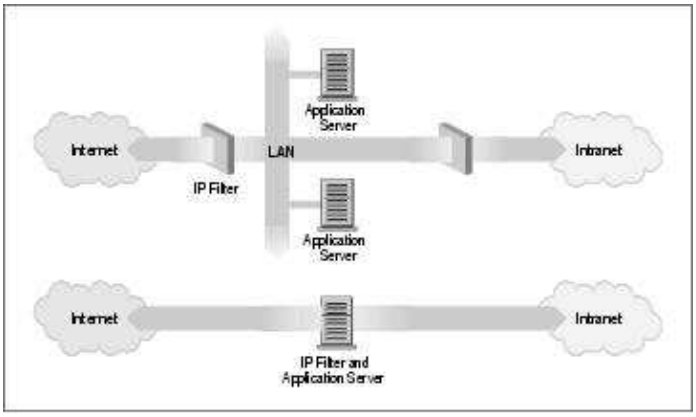
9.2. ¿Qué es un cortafuegos?

Un cortafuegos es una máquina segura y confiable que se asienta entre una red privada y una red pública.[85] La máquina cortafuegos se configura con un conjunto de reglas que determinan a qué tráfico de red se le permitirá pasar y cuál será bloqueado o rechazado. En algunas organizaciones grandes, puede que encuentre un cortafuegos localizado dentro de la red corporativa para separar áreas sensibles de la organización de otros empleados. Algunos casos de criminalidad informática acontecen dentro de la misma organización, no sólo provienen de fuera.

Se pueden construir cortafuegos en una variedad de maneras. La configuración más sofisticada involucra un número de máquinas separadas y se conoce como *red perimetral* [86]. Dos máquinas, denominadas estranguladoras [87] actúan como "filtros" para permitir pasar sólo ciertos tipos de tráfico de red, y entre estos estranguladores residen servidores de red como una pasarela de correo o un servidor intermediario [88] de 'World Wide Web'. Esta configuracón puede resultar muy segura y permite de forma fácil un amplio rango de control sobre quién puede conectarse tanto desde dentro hacia fuera cómo desde fuera hacia dentro. Este tipo de configuración debería ser el utilizado por las grandes organizaciones.

Sin embargo, típicamente los cortafuegos son máquinas únicas que sirven todas estas funciones. Esto es algo menos seguro, porque si hay alguna debilidad en la propia máquina del cortafuegos que le permita a alguien conseguir el acceso al mismo cortafuegos, la seguridad de toda la red habrá sido comprometida. Sin embargo, estos tipos de cortafuegos son más baratos y fáciles de mantener que la configuración más sofistificada descrita arriba. La Figura 9−1 ilustra los dos tipos más comunes de configuración de cortafuegos.

Figura 9−1. Las dos clases más importantes de diseño de cortafuegos



El núcleo de GNU/Linux proporciona un rango de características internas que le permiten funcionar bastante bien como un cortafuegos de IP. La implementación de red incluye código para realizar filtros a nivel de IP en numerosas formas, y proporciona un mecanismo para configurar con precisión qué tipos de reglas le gustaría imponer. El cortafuegos en GNU/Linux es suficientemente flexible como para convertirle en algo muy útil en cualquiera de las configuraciones ilustradas en la Figura 9−1. El 'software' de cortafuegos de Linux proporciona otras dos características muy útiles que se discutirán en capítulos por separado: auditoría de IP (Capítulo 10) y enmascaramiento de IP (Capítulo 11).

9.3. ¿Qué es el filtrado de IP?

El filtrado de IP es simplemente un mecanismo que decide qué tipos de datagramas de IP serán procesados normalmente y cuáles serán descartados. Por *descartados* se entiende que el datagrama se elimina y se ignora completamente, como si nunca se hubiera recibido. Usted puede aplicar muchos criterios , y en diferentes ordenamientos, para determinar qué datagramas desea filtrar; algunos ejemplos de ésto son:

•

Tipo de protocolo: TCP, UDP, ICMP, etc.

- Número de conector [89] (para TCP/UDP)
- Tipo de datagrama: SYN/ACK, datos, petición de eco de ICMP, etc.
- Dirección de origen del datagrama: de donde proviene
- Dirección de destino del datagrama: a donde se dirige

Llegado este punto, resulta muy importante comprender que el filtrado de IP es una utilidad en la capa de red. Esto significa que este mecanismo no entiende nada acerca de la aplicación que utiliza las conexiones de red, sólo sabe acerca de las conexiones mismas. Por ejemplo, usted puede denegar el acceso a usuarios a su red interna por el puerto predeterminado de telnet, pero si se apoya únicamente en el filtrado de IP, no podrá evitar que se utilice el programa de telnet en un puerto por el que usted permite el paso a través de su cortafuegos. Puede evitar este tipo de problemas haciendo uso de servidores intermediarios para cada servicio que permita que cruce su cortafuegos. Los servidores intermediarios comprenden la aplicación para la que fueron diseñados y por tanto evitan los abusos, tales como utilizar el programa de telnet para pasar a través de un cortafuegos utilizando el puerto de 'World Wide Web'. Si su cortafuegos soporta un servidor intermediario de 'World Wide Web', aquella conexión de telnet será siempre respondida por el servidor intermediario que sólo permitará que pasen peticiones HTTP. Existe un gran número de programas servidores intermediarios. Algunos son software libre y muchos otros son productos comerciales. El documento 'Firewall−HOWTO' [90] expone un subconjunto popular de aquellos, pero esto queda fuera del alcance de este libro.

El conjunto de reglas de filtrado de IP se construye a partir de muchas combinaciones de los criterios enumerados previamente. Por ejemplo, imagínese que usted quiere que los usuarios del 'World Wide Web' dentro de la red de la Cervecera Virtual no tengan acceso a ningún servicio de Internet excepto a los servidores web. Entonces configuraría su cortafuegos permitiendo el reenvío de:

- datagramas con una dirección de origen dentro de la red de la Cervecera Virtual, una dirección de destino cualquiera y con un puerto de destino igual a 80 (el de WWW)
- datagramas con dirección de destino dentro de la red de la Cervecera Virtual y un puerto de origen igual a 80 (WWW) siendo cualquiera la dirección de origen

Nótese que se han utilizado dos reglas aquí. Se tiene que permitir que nuestros datos salgan fuera, pero también que la correspondiente respuesta vuelva. En la práctica, como se verá en breve, Linux simplifica esto y nos permite especificar lo mismo en una sola orden.

9.4. Configuración de Linux como cortafuegos

Para poder construir un cortafuegos IP con Linux, es necesario disponer de un núcleo compilado con soporte de cortafuegos de IP y de la utilidad de configuración adecuada. En todos los núcleos anteriores a la serie 2.2 se usaba la utilidad **ipfwadm**. Los núcleos 2.2.x supusieron el lanzamiento de la tercera generación de cortafuegos de IP para Linux que se denominó *'IP Chains'*. 'IP chains' utiliza un programa similar a **ipfwadm** que se llama **ipchains**. Los núcleos de Linux 2.3.15 y siguientes soportan la cuarta generación de cortafuegos de IP de Linux que se denomina *netfilter*. El código de *netfilter* es el resultado de un gran rediseño del flujo en el manejo de paquetes en Linux. *Netfilter* es una criatura con múltiples caras, pues proporciona un soporte compatible hacia atrás tanto con **ipfwadm** como con **ipchains** además de una nueva orden alternativa que se llama **iptables**. En las próximas secciones se hablará de las diferencias entre los tres.

9.4.1. Núcleo configurado con cortafuegos de IP

El núcleo de Linux debe configurarse para que dé soporte a las funciones de cortafuegos de IP. Sólo hay que seleccionar las opciones adecuadas cuando se realiza un `make menuconfig` del núcleo. [91] En el Capítulo 3 se describe cómo hacerlo. En los núcleos 2.2, las siguientes opciones deberían ser seleccionadas:

```
    Networking options  --->
        [*] Network firewalls
        [*] TCP/IP networking
        [*] IP: firewalling
        [*] IP: firewall packet logging
```

En cambio, en los núcleos 2.4.0 y posteriores se deberían seleccionar estas opciones:

```
      Networking options  --->
         [*] Network packet filtering (replaces ipchains)
             IP: Netfilter Configuration  --->
                  .
                 <M> Userspace queueing via NETLINK (EXPERIMENTAL)
                 <M> IP tables support (required for filtering/masq/NAT)
                 <M>   limit match support
                 <M>   MAC address match support
                 <M>   netfilter MARK match support
                 <M>   Multiple port match support
                 <M>   TOS match support
                 <M>   Connection state match support
                 <M>   Unclean match support (EXPERIMENTAL)
                 <M>   Owner match support (EXPERIMENTAL)
                 <M>   Packet filtering
                 <M>     REJECT target support
                 <M>     MIRROR target support (EXPERIMENTAL)
                  .
                 <M>   Packet mangling
                 <M>     TOS target support
                 <M>     MARK target support
                 <M>   LOG target support
                 <M> ipchains (2.2-style) support
                 <M> ipfwadm (2.0-style) support
```

9.4.2. La utilidad ipfwadm

La utilidad **ipfwadm** (el administrador del cortafuegos de IP) es la herramienta que se utiliza para construir las reglas del cortafuegos para todos los núcleos anteriores al 2.2.0. La sintaxis de las órdenes puede resultar muy confusa porque permite realizar un amplio espectro de cosas; aquí se proporcionarán algunos ejemplos comunes que ilustrarán las variaciones más importantes dentro de ese espectro.

La utilidad **ipfwadm** se incluye en la mayoría de las distribuciones modernas de Linux, aunque quizás no por defecto. Puede que haya un paquete de 'software' específico que tenga que instalar. Si su distribución no la incluye, puede obtener el paquete con el código fuente de ftp.xos.nl dentro del directorio `/pub/linux/ipfwadm/`, y compilarla usted mismo.

9.4.3. La utilidad ipchains

Al igual que la utilidad **ipfwadm**, la utilidad **ipchains** puede resultar algo desconcertante al principio. Proporciona toda la flexibilidad de **ipfwadm** con una sintaxis simplificada, y además proporciona un mecanismo de encadenamiento que le permite gestionar múltiples conjuntos de reglas y enlazarlas conjuntamente. Se cubrirá el encadenamiento de reglas en una sección independiente cerca del final de este capítulo, porque resulta un concepto avanzado en la mayoría de las situaciones.

La orden **ipchains** aparece en la mayoría de las distribuciones de Linux basadas en los núcleos 2.2. Si desea compilarla usted mismo, puede encontrar el paquete con el código fuente en el sitio de desarrollo *http://www.rustcorp.com/linux/ipchains/*. Con el paquete del código fuente se incluye un guión [92] denominado **ipfwadm−wrapper** que imita a la orden **ipfwadm**, pero que realmente invoca la orden **ipchains**. La migración de una configuración preexistente de cortafuegos resulta menos costosa gracias a este complemento.

9.4.4. La utilidad iptables

La sintaxis de la utilidad **iptables** es bastante similar a la de **ipchains**. Los cambios consisten en mejoras y en el resultado del rediseño de la herramienta para que sea extensible a través de bibliotecas dinámicas. Al igual que en el caso de **ipchains**, se presentarán los equivalentes de los ejemplos con **iptables** de tal forma que pueda comparar y contrastar su sintaxis con la de las otras utilidades.

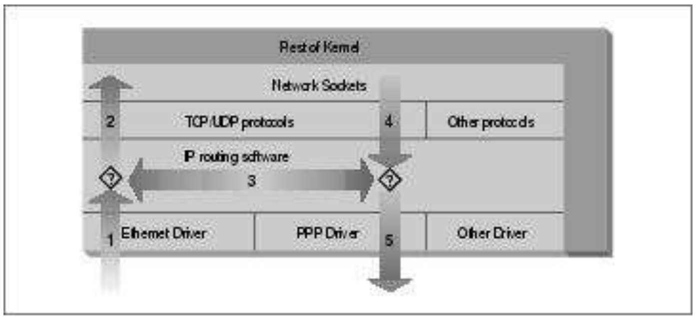
La utilidad **iptables** se incluye en el paquete de código fuente de *netfilter* que está disponible en *http://www.samba.org/netfilter/*. También estará incluído en cualquier distribución basada en la serie de núcleos 2.4.

Se hablará un poco del importante paso hacia delante que *netfilter* representa en una sección dedicada más adelante en este capítulo.

9.5. Las tres formas posibles de filtrado

Considérese cómo una máquina Unix, o de hecho cualquier máquina capaz de realizar encaminamiento de IP, procesa los datagramas de IP. Los pasos básicos, mostrados en la Figura 9−2 son:

Figura 9−2. Las etapas del procesamiento de un datagrama de IP



- Se recibe el datagrama de IP. (1)

- Se examina el datagrama de IP entrante para determinar si está destinado a un proceso de esta máquina.

-

Si el datagrama es para esta máquina, se procesa localmente.(2)

- Si no está destinado a esta máquina, se realiza una búsqueda en la tabla de encaminamiento de una ruta adecuada y el datagragama se reenvía por la interfaz adecuada o se elimina si no se puede encontrar una ruta. (3)
- Los datagramas procedentes de procesos locales se envían hacia el 'software' de encaminamiento para ser reenviados hacia la interfaz apropiada. (4)
- Se examina el datagrama de IP saliente para determinar si existe una ruta válida que escoger, si no es así, se elimina.
- Se transmite el datagrama de IP. (5)

En nuestro diagrama, el flujo 135 representa nuestra máquina encaminando datos entre un 'host' sobre nuestra red Ethernet y un 'host' alcanzable vía nuestro enlace de PPP. Los flujos 12 y 45 representan los flujos de entrada y de salida de datos de un programa de red ejecutándose en nuestro 'host' local. El flujo 432 representaría un flujo de datos vía una conexión 'loopback'. Naturalmente, los datos fluyen tanto hacia dentro como hacia fuera de los dispositivos de red. Los símbolos de interrogación del diagrama representan los puntos donde la capa de IP realiza las decisiones de encaminamiento.

El cortafuegos de IP del núcleo de Linux es capaz de aplicar filtrados en varias etapas de este proceso. Es decir, se pueden filtrar los datagramas de IP que entren en su máquina, aquellos que estén siendo reenviados a través de su máquina y aquellos que estén preparados para ser transmitidos.

En **ipfwadm** y en **ipchains**, una regla de tipo 'Input' [93] se aplica al flujo 1 del diagrama, una regla de tipo 'Forwarding' [94] al flujo 3 y una regla de tipo 'Output' [95] al flujo 5. Cuando se discuta *netfilter* más adelante se verá que los puntos de interceptación han cambiado de tal forma que una regla de tipo 'Input' se aplica ahora en el flujo 2, y una regla de tipo 'Output' en el flujo 4. Esto tiene implicaciones importantes sobre cómo se deben estructurar los conjuntos de reglas, pero los principios generales permanecen válidos para todos los tipos de cortafuegos de GNU/Linux.

Todo esto puede parecer complicado de forma innecesaria en un primer momento, pero proporciona una flexibilidad que permite construir configuraciones muy sofisticadas y poderosas.

9.6. El cortafuegos original de IP (núcleos 2.0)

La primera generación del soporte de cortafuegos de IP para GNU/Linux apareció en la serie de núcleos 1.1. Consistía en una implementación del cortafuegos ipfw de BSD por Alan Cox. El soporte de cortafuegos que apareció en la serie de núcleos 2.0 que constituye la segunda generación fue una mejora de Jos Vos, Pauline Middelink y otros.

9.6.1. Uso de ipfwadm

La orden **ipfwadm** era la herramienta de configuración para la segunda generación de cortafuegos de IP de GNU/Linux. Quizás la forma más simple de describir el uso de la orden **ipfwadm** es con un ejemplo. Para empezar, se codificará el ejemplo que se presentó antes.

9.6.1.1. Un ejemplo trivial

Supóngase que se dispone de una red en nuestra organización y que se utiliza una máquina cortafuegos basada en GNU/Linux para conectar la red a Internet. Además, supóngase que se desea que los usuarios de la red sean capaces de acceder a servidores 'web' de Internet, pero que cualquier otro tipo de tráfico no sea permitido.

Se pondrá una regla de tipo 'forwarding' para permitir que los datagramas con dirección de origen en nuestra red y un conector de destino con puerto 80 sean reenviados hacia fuera, y los correspondientes datagramas de respuesta sean reenviados de vuelta vía el cortafuegos.

Asúmase que nuestra red tiene una máscara de 24 bits (clase C) y una dirección de 172.16.1.0. La reglas que se podrían utilizar serían:

```
# ipfwadm -F -f
    # ipfwadm -F -p deny
    # ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80
    # ipfwadm -F -a accept -P tcp -S 0/0 80 -D 172.16.1.0/24
```

El argumento `-F` de la línea de órdenes significa especifica a **ipfwadm** que es una regla de tipo 'forwarding', es decir, de reenvío. La primera orden instruye a **ipfwadm** que se "desprenda" de todas las reglas de tipo 'forwarding'. Esto asegura que se trabajará con un estado conocido antes de que se añadan reglas específicas.

La segunda regla establece nuestra política predeterminada de reenvío. Se le dice al núcleo que niegue o que no permita el reenvío de datagramas de IP. Es muy importante establecer la política por omisión, porque describe qué le pasará a cualquier datagrama que no esté específicamente controlado por cualquier otra regla. En la mayoría de las configuraciones de cortafuegos, usted querrá establecer la política por defecto a 'deny' [96], como se muestra en el ejemplo, para estar seguro de que sólo el tráfico que usted específicamente permita pasar su cortafuegos sea reenviado.

La tercera y la cuarta reglas son las que implementan el requisito. La tercera orden permite que nuestros datagramas salgan, y la cuarta permite las respuestas de vuelta.

Vamos a revisar cada unos de los argumentos:

−F
: Esta es una regla de tipo 'forwarding'.

−a accept
: Añadir esta regla con la política establecida a "aceptar", lo que quiere decir que se reenviará cualquier datagrama que se ajuste a esta regla

−P tcp
: Esta regla se aplica a los datagramas de TCP (en lugar de UDP o ICMP).

−S 172.16.1.0/24
: Los primeros 24 bits de la dirección de origen deben coincidir con los de la dirección de red 172.16.1.0.

−D 0/0 80
: La dirección de destino debe tener cero bits coincidentes con la dirección 0.0.0.0. Esto en el fondo es una forma de decir "cualquier dirección". El 80 es el puerto de destino, en este caso el de WWW. También puede utilizarse cualquier entrada que aparezca en el fichero `/etc/services` para describir el puerto, de tal forma que `-D 0/0 www` habría funcionado igual de bien.

ipfwadm acepta las máscaras de red en una forma con la que puede no esté familiarizado. La notación `/nn` es una forma de describir cuántos bits de la dirección suministrada son significativos, es decir, es el tamaño de la máscara de red. Los bits se cuentan siempre de izquierda a derecha; algunos ejemplos habituales se muestran en la Tabla 9−1.

Tabla 9−1. Valores habituales de máscaras de red y bits

Máscara	Bits
255.0.0.0	8
255.255.0.0	16
255.255.255.0	24
255.255.255.128	25
255.255.255.192	26
255.255.255.224	27
255.255.255.240	28
255.255.255.248	29
255.255.255.252	30

Se mencionó antes que **ipfwadm** implementa un pequeño truco que permite que sea más fácil añadir estos tipos de reglas. Este truco consiste en el uso de la opción *−b*, que convierte a la orden en una regla bidireccional.

El modificador de bidireccionalidad nos permite unir nuestras dos reglas en una sola como sigue:

```
# ipfwadm −F −a accept −P tcp −S 172.16.1.0/24 −D 0/0 80 −b
```

9.6.1.2. Un refinamiento importante

Eche una mirada más atenta a nuestro conjunto de reglas. ¿ Puede apreciar que todavía existe un método de ataque que alguien de fuera podría utilizar para engañar a nuestro cortafuegos ?

Nuestro conjunto de reglas permite que todos los datagramas procedentes de fuera de nuestra red con un puerto de origen de 80 pasen. ¡ Esto incluiría a aquellos datagramas cuyo bit de SYN valga 1 ! El bit SYN es lo que declara a un datagrama de TCP que sea una petición de conexión. Si una persona de fuera tuviera un acceso privilegiado a un 'host', podría realizar una conexión a través de nuestro cortafuegos con cualquiera de nuestros 'hosts', dado el supuesto de que utilizará el puerto 80 en su extremo. Esto no es lo que se deseaba.

Afortunadamente, existe una solución a este problema. La orden **ipfwadm** proporciona otro modificador que permite construir reglas que coincidan con datagramas cuyo bit de SYN valga 1. Cambiemos nuestro ejemplo para incluir una regla de este tipo:

```
# ipfwadm −F −a deny −P tcp −S 0/0 80 −D 172.16.10.0/24 −y
    # ipfwadm −F −a accept −P tcp −S 172.16.1.0/24 −D 0/0 80 −b
```

El modificador `−y` hace que la regla coincida sólo si el bit SYN del datagrama vale 1. Así nuestra nueva regla dice: "Deniega cualquier datagrama destinado a nuestra regla procedente de cualquier sitio con un puerto de origen igual a 80 y bit SYN igual a 1", o "deniega cualquier petición de conexión desde 'hosts' utilizando el puerto 80"

¿Por qué se ha puesto esta regla especial *antes* de la regla principal? Las reglas de cortafuegos de IP operan de tal forma que la primera coincidencia es la regla que se utiliza. Ambas reglas coincidirían con los datagramas que queremos detener, por tanto debemos asegurarnos que se ha puesto la regla con la instrucción `deny` antes que la regla con la instrucción `accept`.

9.6.1.3. Listado de nuestras reglas

Después de haber introducido nuestras reglas, se puede pedir a **ipfwadm** que las liste con la orden:

```
# ipfwadm -F -l
```

Esta orden mostrará todas las reglas de reenvío configuradas. La salida debería parecerse a algo como esto:

```
# ipfwadm -F -l
    IP firewall forward rules, default policy: accept
    type  prot source               destination          ports
    deny  tcp  anywhere             172.16.10.0/24       www -> any
    acc   tcp  172.16.1.0/24        anywhere             any -> www
```

La orden **ipfwadm** intentará traducir el número de puerto en un nombre de servicio utilizando el fichero `/etc/services`, si es que tiene alguna entrada correspondiente.

La salida por omisión carece de algunos detalles importantes para nosotros. En la salida con el listado predeterminado no se puede ver el efecto del argumento `-y`. La orden **ipfwadm** es capaz de producir un listado más detallado si se especifica además el argumento `-e` (salida extendida). Aquí no se muestra la salida completa porque es demasiado ancha para la página, pero sí que incluye una columna para las opciones de nombre `opt` que muestra la opción `-y` que controlla los paquetes de tipo SYN:

```
# ipfwadm -F -l -e
    P firewall forward rules, default policy: accept
     pkts bytes type  prot opt  tosa tosx ifname  ifaddress   source        ...
        0     0 deny  tcp  --y- 0xFF 0x00 any     any         anywhere      ...
        0     0 acc   tcp  b--- 0xFF 0x00 any     any         172.16.1.0/24 ...
```

9.6.2. Un ejemplo más complejo

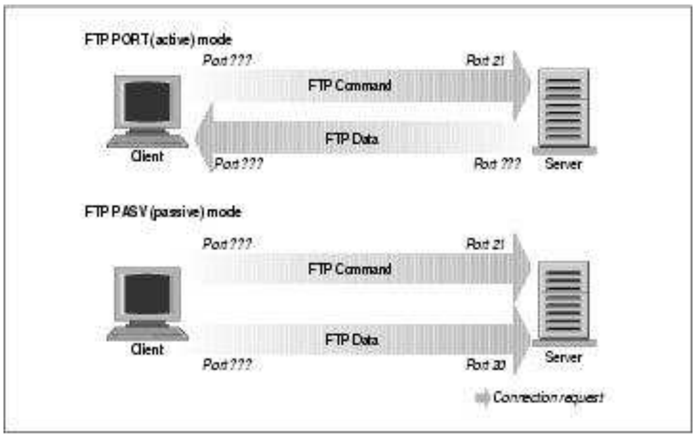
El ejemplo anterior era un ejemplo simple. No todo los servicios de red son tan simples de configurar como el servicio de WWWW; en la práctica, la configuración de un cortafuegos típico resultaría ser mucho más compleja. Vamos a examinar otro ejemplo común, esta vez FTP. Se quiere que los usuarios de la red interna puedan entrar en servidores de FTP de Internet para leer y escribir ficheros. Pero no se desea que personas de Internet puedan entrar en nuestros servidores de FTP.

Es sabido que FTP utiliza dos puertos de FTP: el puerto 20 (ftp−data) y el puerto 21 (ftp), por tanto:

```
# ipfwadm -a deny -P tcp -S 0/0 20 -D 172.16.1.0/24 -y
    # ipfwadm -a accept -P tcp -S 172.16.1.0/24 -D 0/0 20 -b
    #
    # ipfwadm -a deny -P tcp -S 0/0 21 -D 172.16.1.0/24 -y
    # ipfwadm -a accept -P tcp -S 172.16.1.0/24 -D 0/0 21 -b
```

¿Correcto? Bueno, no necesariamente. Los servidores de FTP pueden operar en dos modos diferentes: el modo pasivo y el modo activo. [97] En el modo pasivo, el servidor de FTP permanece escuchando en espera de una conexión desde el cliente. En el modo activo, es el servidor el que realmente realiza la conexión con el cliente. El modo activo es el habitual por omisión. Las diferencias se ilustran en la Figura 9−3.

Figura 9−3. Modos de un servidor de FTP



Muchos servidores de FTP realizan su conexión de datos desde el puerto 20 cuando operan en el modo activo, lo que simplifica las cosas un poco, pero, degraciadamente, no todos proceden así. [98]

Pero, ¿cómo nos afecta todo esto? Fíjese en nuestra regla del puerto 20, el puerto de datos de FTP (FTP−data). La regla, tal como se tiene en este momento, asume que la conexión será realizada por nuestro cliente al servidor. Esto funcionará si se utiliza el modo pasivo. Pero resulta muy difícil para nosotros el configurar una regla satisfactoria que permita el modo activo de FTP, porque no se puede saber de antemano qué puertos serán los utilizados. Si abrimos nuestro cortafuegos para permitir conexiones entrantes en cualquier puerto, estaríamos exponiendo nuestra red a un ataque sobre todos los servicios que acepten conexiones.

El dilema se resuelve de la forma más satisfactoria insistiendo en que nuestros usuarios operen en el modo pasivo. La mayoría de los servidores de FTP y muchos clientes de FTP funcionarán de esta forma. El cliente popular **ncftp** también soporta el modo pasivo, pero requiere un pequeño cambio de configuración para conseguir que su modo por omisión sea el pasivo. Muchos navegadores de 'World Wide Web' como el navegador de Netscape también soportan el modo pasivo de FTP, por lo que no debería ser muy difícil el encontrar el 'software' adecuado para utilizar. De forma alternativa, se puede evitar el asunto de forma completa utilizando un programa intermediario de FTP que acepten una conexión desde la red interna y establezca conexiones con las redes externas.

Cuando construya su cortafuegos, probablemente se encontrará con varios de estos problemas. Debería siempre pensar cuidadosamente cómo funciona un servico realmente para estar seguro de que ha puesto un conjunto de reglas adecuado a ese servicio. La configuración de un cortafuegos de verdad puede resultar bastante compleja.

9.6.3. Resumen de los argumentos de ipfwadm

La orden **ipfwadm** tiene muchos argumentos diferentes que están relacionados con la configuración del cortafuegos de IP. La sintaxis general es:

```
ipfwadm categoría orden parámetros [opciones]
```

Veamos cada cosa.

9.6.3.1. Categorías

Sólo puede introducirse una de estas categorías. La categoría le dice al cortafuegos qué tipo de regla de cortafuegos se está configurando:

−I
: regla de tipo 'Input'

−O

regla de tipo 'Output'

−F
regla de tipo 'Forwarding'

9.6.3.2. Órdenes

Al menos una de las siguientes órdenes debe ser introducida y se aplican sólo aquellas reglas relacionadas con la categoría introducida. La orden le dice al cortafuegos qué acción debe tomar.

−a [política]
Añade una nueva regla

−i [política]
Inserta una nueva regla

−d [política]
Borra una regla existente

−p política
Establece la política por defecto

−l
Muestra todas las reglas existentes

−f
Destruye todas las reglas existentes

Las políticas relevantes para el cortafuegos de IP y sus significados son:

accept
Permite que los datagramas coincidentes sean recibidos, reenviados o transmitidos

deny
Impide que los datagramas coincidentes sean recibidos, reenviados o transmitidos

reject
Impide que los datagramas coincidentes sean recibidos, reenviados o transmitidos y envía al 'host' que envió el datagrama un mensaje de error de ICMP.

9.6.3.3. Parámetros

Al menos uno de los siguientes parámetros debe ser introducido. Utilice los parámetros para especificar a qué datagramas se aplica esta regla:

−P protocolo
Puede ser TCP, UDP, ICMP o todos. Ejemplo:

```
-P tcp
```

−S dirección[/máscara] [puerto]
La dirección IP de origen que buscar coincidencias coná con esa regla. Se asumirá una máscara de /32

bits si no se proporciona una. Opcionalmente, puede especificar a qué puertos se aplicará esta regla. También puede especificar el protocolo utilizando el argumento `-P` que se describió más arriba. Si no se especifica el puerto o un rango de puertos, se supondrá que todos los puertos buscar coincidencias conán. Los puertos pueden especificarse por su nombre, utilizando la entrada del fichero `/etc/services` que desee. En el caso del protocolo de ICMP, el campo de puerto se utiliza para indicar el tipo de datagrama de ICMP. Pueden introducirse rangos de puertos; para ello utilice la sintaxis genérica: *`puerto inferior`*:*`puerto superior`*. Ejemplo:

```
-S 172.29.16.1/24 ftp:ftp-data
```

-D dirección[/máscara] [puerto]
: Especifica la dirección IP de destino que buscar coincidencias coná con la regla. La dirección de destino se codifica con las mismas reglas que la dirección de origen descrita previamente. Ejemplo:

```
-D 172.29.16.1/24 smtp
```

-V dirección
: Especifica la dirección del interfaz de red por el que el paquete se recibe (`-I`) o se envía (`-O`). Esto nos permite crear reglas que sólo se apliquen a ciertas interfaces de red de nuestra máquina. Ejemplo:

```
-V 172.29.16.1
```

-W nombre
: Especifica el nombre del interfaz de red. Este argumento funciona de la misma manera que el argumento `-V`, excepto que se proporciona el nombre del dispositivo en lugar de su dirección. Ejemplo:

```
-W ppp0
```

9.6.3.4. Argumentos opcionales

Estos argumentos resultan muy útiles a veces:

-b
: Utilizado para establecer el modo bidireccional. Este modificador hace que coincida el tráfico entre el origen y el destino especificados fluyendo en cualquier sentido. Esto ahorra el crear dos reglas: una para el sentido hacia delante de la conexión y otra para el sentido contrario.

-o
: Esto habilita el apunte en el registro del núcleo de información sobre los datagramas coincidentes. Cualquier datagrama que coincida con esta regla será registrado en un mensaje del núcleo. Esto resulta útil para posibilitar la detección de accesos no autorizados.

-y
: Utilizado para buscar coincidencias con datagramas de establecimiento de la conexión de TCP. Esta opción causa que la regla coincida sólo con los datagramas que intenten establecer conexiones de TCP. Únicamente los datagramas que tengan su bit SYN con un valor de uno, y su bit ACK con un valor de 0, buscar coincidencias conán. Esto resulta útil para filtrar los intentos de conexión de TCP y se ignora en el caso de otros protocolos.

-k
: Utilizado para buscar coincidencias con datagramas de acuse de recibo de TCP. Esta opción causa que la regla coincida sólo con los datagramas que sean acuse de recibos de paquetes que intentan establecer conexiones de TCP. Únicamente los datagramas que tenga su bit ACK con valor igual a 1.

Esto resulta útil para filtrar los intentos de conexión de TCP y se ignora en el caso de otros protocolos.

9.6.3.5. Tipos de datagrama de ICMP

Cada una de las órdenes de configuración del cortafuegos le permite especificar tipos de datagrama de ICMP. Al contario que los puertos de TCP y de UDP, no existe un fichero de configuración conveniente que liste los tipos de datagramas y sus significados. Los tipos de datagrama de ICMP se definen en el RFC−1700, el RFC de los números asignados. Los tipos de datagrama de ICMP aparecen también listados en uno de los ficheros de cabecera de la biblioteca estándar de C. El fichero `/usr/include/netinet/ip_icmp.h`, que pertenece al paquete con la biblioteca estándar de GNU, y que los programadores de C utilizan cuando escriben 'software' de red que utilice el protocolo de ICMP, también define los tipos de datagrama de ICMP. Para su conveniencia, se incluyen aquí en la Tabla 9−2 [99]. La interfaz de la orden **iptables** le permite especificar los tipos de ICMP por su nombre, por lo que también se muestran los nombre nemotécnicos que utiliza.

Tabla 9−2. Tipos de datagramas de ICMP

Número de tipo	Nnemónico de iptables	Descripción del tipo
0	echo−reply	Respuesta a eco
3	destination−unreachable	Destino inaccesible
4	source−quench	Disminución del tráfico desde el origen
5	redirect	Redirección
8	echo−request	Solicitud de eco
11	time−exceeded	Tiempo superado
12	parameter−problem	Problema de parámetros
13	timestamp−request	Solicitud de marca de tiempo
14	timestamp−reply	Respuesta de marca de tiempo
15	none	Solicitud de información
16	none	Respuesta de información
17	address−mask−request	Petición de máscara de dirección
18	address−mask−reply	Respuesta de máscara de dirección

9.7. Cortafuegos 'IP Chains' (núcleos 2.2)

La mayoría de los aspectos de GNU/Linux evolucionan para satisfacer las cada vez mayores demandas de sus usuarios; el cortafuegos de IP no es una excepción. La implementación del cortafuegos de IP tradicional resulta suficiente para la mayoría de las aplicaciones, pero puede resultar engorroso y poco eficiente para configurar en entornos complejos. Para resolver este problema, se desarrolló un nuevo método de configuración del cortafuegos de IP así como nuevas características relacionadas. Este nuevo método fue denominado Cortafuegos 'IP Chains'[100] y fue liberado por vez primera para uso general en el núcleo 2.2.0.

El soporte del cortafuegos 'IP Chains' fue desarrollado por Paul Russell y Michael Neuling[101]. Paul es el autor del documento sobre 'IP Chains' IPCHAINS−HOWTO.

El cortafuegos 'IP Chains' le permite desarrollar clases de reglas de cortafuegos a las que puede entonces añadir y quitar 'hosts' o redes. Una consecuencia colateral del encadenamiento de reglas de cortafuegos es que puede mejorar el rendiminento del cortafuegos en aquellas configuraciones en las que haya montones de reglas.

El cortafuegos 'IP Chains' está soportado por las series de núcleos 2.2 y también se encuentran disponibles como un parche para la series de núcleos 2.0.*. El HOWTO describe dónde obtener el parche y proporciona montones de pistas útiles sobre cómo utilizar de forma efectiva la utilidad de configuración **ipchains**.

9.7.1. Uso de ipchains

Existen dos formas de emplear la utilidad **ipchains**. La primera forma consiste en utilizar el guión de "shell" **ipfwadm−wrapper**, que es básicamente un sustituto de la orden **ipfwadm** y que llama por debajo al programa **ipchains**. Si esto es lo que desea, no siga leyendo y relea las secciones previas que describen **ipchains**, poniendo **ipfwadm−wrapper** en su lugar. Esto funcionará, pero no se garantiza que el guión se mantenga en un futuro, y en ese caso no dispondrá de las características avanzadas que el cortafuegos 'IP Chains' vaya a ofrecer.

La segunda forma de utilizar **ipchains** consiste en aprender su nueva sintaxis y modificar cualquier configuración existente que exija utilizar la nueva sintaxis en lugar de la antigua. Con algunas consideraciones cuidadosas, se dará cuenta de que puede optimizar su configuración a la vez que realiza la conversión. La sintaxis de **ipchains** es más fácil de aprender que la de **ipfwadm**, por lo que resultará una buena opción.

La orden **ipfwadm** manipulaba tres conjuntos de reglas para el propósito de configurar el cortafuegos. Con el cortafuegos 'IP Chains', podrá crear un número arbitrario de conjuntos de reglas, cada una enlazada con otra, pero siguen estando presentes siempre tres conjuntos de reglas relacionadas con la función del cortafuegos. Los conjuntos de reglas estándares son los directos equivalentes de los utilizados con **ipfwadm**, exceptuando el hecho de que ahora tiene nombre: `input`, `forward` y `output`.

Veamos primero la sintaxis general de la orden **ipchains**, después se verá como utilizar **ipchains** en lugar de **ipfwadm** sin preocuparse acerca de sus características avanzadas de encadenamiento. Se hará reutilizando nuestros ejemplos anteriores

9.7.2. Sintaxis de la orden ipchains

La sintaxis de la orden **ipchains** es bastante directa. Se contemplarán los ejemplos más importantes. La sintaxis general de la mayoría de las órdenes de **ipchains** es:

```
ipchains orden especificación_de_regla opciones
```

9.7.2.1. Órdenes

Existen diversas formas de manipular las reglas y conjuntos de reglas con la orden **ipchains**. Las relevantes para la funcionalidad de cortafuegos de IP son:

−A cadena
: Añade una o más reglas al final de la cadena especificada. Si se proporciona un nombre de 'host' como origen o destino que se resuelve a más de una dirección IP, entonces se añade una regla por cada una de las direcciones.

−I cadena numero_de_regla
: Inserta una o más reglas al principio de la cadena especificada. De nuevo, si se proporciona un nombre de 'host', se añade una regla por cada dirección que se resuelva.

−D cadena
: Elimina una o más reglas de la cadena especificada que coincida con la especificación de regla.

−D cadena número_de_regla

Elimina la regla ubicada en la posición `número_de_regla` de la cadena especificada. Las posiciones de reglas comienzan por uno en la primera regla de la cadena.

−R cadena número_de_regla

Reemplaza la regla ubicada en la posición `número_de_regla` de la cadena especificada por la especificación de regla proporcionada.

−C cadena

Comprueba el datagrama que se describe con la especificación de la regla contra la cadena especificada. Esta orden devuelve un mensaje que describe cómo se procesará el datagrama por la cadena. Esto resulta muy útil para comprobar la configuración del cortafuegos, por lo que se verán más detalles un poco más adelante.

−L [cadena]

Muestra las reglas de la cadena especificada, o de todas las cadenas si no se especifica ninguna.

−F [cadena]

Elimina todas las reglas de la cadena especificada, o de todas las cadenas si no se especifica ninguna.

−Z [cadena]

Establece a cero los contadores de datagramas y bytes de la cadena especificada, o de todas las cadenas si no se especifica ninguna.

−N cadena

Crea una nueva cadena con el nombre especificado. No puede existir una cadena con el mismo nombre. Así es como se crean las cadenas de usuario.

−X [cadena]

Elimina la cadena de usuario especificada, o todas las cadenas de usuario especificadas si no se especifica ninguna cadena. Para que esta orden tenga éxito, no deben existir referencias de ninguna otra cadena de reglas a la cadena especificada.

−P política_de_cadena

Establece la política por defecto de la cadena especificada a la política especificada. Las políticas de cortafuegos válidas son `ACCEPT`, `DENY`, `REJECT`, `REDIR`, o `RETURN`. `ACCEPT`, `DENY`, y `REJECT` tienen los mismo significados que las políticas correspondientes de la implentación tradicional del cortafuegos de IP. `REDIR` especifica que se debe redirigir de forma transparente el datagrama a un puerto del 'host' del cortafuegos. La política `RETURN` causa que el código del cortafuegos de IP vuelva a la cadena de cortafuegos que hizo la llamada a la que contiene esta regla y que continúe empezando por la regla situada tras la regla que hizo la llamada.

9.7.2.2. Parámetros de especificación de las reglas

Ciertos parámetros de **ipchains** crean una especificación de reglas al determinar qué tipos de paquetes coinciden. Si se omite algunos de esos parámetros de la especificación de una regla, se asumen sus valores por omisión.

−p [!]protocolo

Especifica el protocolo del datagrama que buscar coincidencias coná con esta regla. Los protocolos válidos son: `tcp`, `udp`, `icmp`, o `todos`. También puede especificarse un número de protocolo para buscar coincidencias con con otros protocolos. Por ejemplo, se puede utilizar el `4` para buscar coincidencias con el protocolo de encapsulamiento `ipip`. Si se proporciona el signo `!`, entonces la regla es negativa y el datagrama buscar coincidencias coná con cualquier protocolo que no sea el especificado. Si no se especifica este parámetro, se asumirá por omisión el valor `all`.

−s [!]dirección[/máscara] [!] [puerto]
: Especifica la dirección de origen y el puerto del datagrama que buscar coincidencias coná con este regla. La dirección puede proporcionarse como un nombre de 'host', un nombre de red o una dirección de IP. El argumento opcional `máscara` es la máscara de red que se utilizará y puede ser proporcionada en la forma tradicional (e.g.,/255.255.255.0) o en la forma moderna (e.g., /24). El argumento opcional `puerto` especifica el puerto de TCP o UDP, o el tipo de datagrama de ICMP que buscar coincidencias coná. Se puede proporcionar una especificación de puerto sólamente si se ha proporcionado el párametro `-p` con uno de los protocolos `tcp`, `udp`, o `icmp`. Se puede especificar los puertos en la forma de un rango, especificando los límites inferior y superior con el signo `:` como delimitador. Por ejemplo, `20:25` describe todos los puertos que van desde el 20 hasta el 25 incluyendo ambos. De nuevo, el signo `!` puede utilizarse para negar los valores.

−d [!]dirección[/máscara] [!] [puerto]
: Especificar la dirección y el puerto de destino del datagrama que buscar coincidencias coná con esta regla. La codificación de este parámetro es la misma que la del parámetro `-s`.

−j blanco
: Especifica la acción que se tomará cuando se coincida con esta regla. Puede pensarse en este parámetro como con el significado de salta a. Los blancos válidos son en principio las políticas `ACCEPT`, `DENY`, `REJECT`, `REDIR`, y `RETURN`. Se describieron sus significados más arriba. Sin embargo, también puede proporcionarse el nombre de una cadena de usuario, y será por donde el proceso continuará. Si se omite este parámetro, no se tomará ninguna acción sobre los datagramas coincidentes con la regla exceptuando la actualización de los contadores de datagrams y de bytes.

−i [!]nombre_de_interfaz
: Especifica la interfaz por la que se recibió o va a transmitirse el datagrama. De nuevo, el signo `!` invierte el resultado de la coincidencia. Si el nombre de la interfaz acaba con un signo + entonces cualquier interfaz que comience con la cadena proporcionada buscar coincidencias coná. Por ejemplo, `-i ppp+` buscar coincidencias coná con cualquier dispositivo de red de PPP y `-i ! eth+` con todas las interfaces excepto las correspondientes a dispositivos de Ethernet.

[!] −f
: Especifica que esta regla se aplica a todo excepto al primer fragmento del un datagrama fragmentado.

9.7.2.3. Opciones

Las siguientes opciones de **ipchains** son más generales por naturaleza propia. Algunas de ellas controlan características bastante esotéricas del 'software' de 'IP Chains':

−b
: Fuerza a que la orden genere dos reglas. Una ajusta el parámetro proporcionado y la otra regla añadida coincide con los parámetros en el sentido contrario.

−v
: Causa que **ipchains** sea más explícito en su salida. Proporcionará más información.

−n
: Causa que **ipchains** muestre las direcciones de IP y los números de puertos en forma de números sin intentar resoverlos contra sus correspondientes nombres.

−l

Habilita el registro del núcleo de los datagramas coincidentes. Cualquier datagrama que coincida con la regla será registrado por el núcleo utilizando su función `printk`, con lo que este registro será gestionado habitualmente por el programa **sysklogd** y escrito a un fichero de registro. Esto resulta muy útil para hacer visibles datagramas poco usuales.

−o[tamaño_máximo]
: Causa que el 'software' de 'IP Chains' copie cualquier datagrama coincidente con la regla al dispositivo netlink del espacio de usuarios. El argumento de tamaño_máximo limita el número de bytes que se pasarán desde cada datagrama al dispositivo netlink. Esta opción resulta de la mayor utilidad para los desarrolladores de 'software', pero puede que sea aprovechada por paquetes de 'software' en el futuro.

−m valor_de_marca
: Causa que los datagramas coincidentes sean *marcados* con un valor. Los valores de las marcas son números de 32 bits sin signo. En las implementaciones actuales esto no hace nada, pero en algún momento en el futuro puede que sirvan para determinar cómo otro 'software', como un código de encaminamiento, tratará al datagrama. Si un valor de una marca comienza con el signo + o con el signo −, el valor se añade o se substrae del valor actual de la marca.

−t máscara_and máscara_xor
: Le permite manipular los bits del tipo de servicio de la cabecera de IP de cualquier datagrama que coincida con esta regla. Los bits de tipo de servicio son utilizados por los encaminadores inteligentes para gestionar la prioridad de los datagramas antes de reenviarlos. El software de encaminamiento de GNU/Linux es capaz de realizar este tipo de asignación de prioridades. La *`máscara_and`* y la *`máscara_xor`* representar máscaras de bits con las que se realizarán respectivamente un AND lógico o un OR lógico con los bits del tipo de servicio del datagrama. Esto constituye una característica avanzada que se discute con más detalle en el IPCHAINS−HOWTO.

−x
: Fuerza que los números de salida de **ipchains** aparezcan con sus valores exactos sin ninguna aproximación.

−y
: Causa que la regla coincida con cualquier datagrama de TCP cuyo bit SYN valga 1 y los bits ACK y FIN lleven un valor de 0. Esto se utiliza para filtrar las peticiones de conexión de TCP.

9.7.3. Nuestro ejemplo simple revisado

Vamos de nuevo a suponer que se dispone de una red en nuestra organización y que se utiliza una máquina cortafuegos basada en GNU/Linux para permitir a nuestros usuarios el acceso a servidores de WWW en Internet, y para impedir cualquier otro tipo de tráfico.

r Si nuestra red tiene una máscara de red de 24 bits (clase C) y tiene como dirección 172.16.1.0, podrían utilizarse las siguientes reglas de **ipchains**:

```
# ipchains −F forward
    # ipchains −P forward DENY
    # ipchains −A forward −s 0/0 80 −d 172.16.1.0/24 −p tcp −y −j DENY
    # ipchains −A forward −s 172.16.1.0/24 −d 0/0 80 −p tcp −b −j ACCEPT
```

La primera de las órdenes borra todas las reglas del conjunto de reglas `forward` y el segundo establece la política predeterminada del conjunto de reglas `forward` a `DENY`. Por último, la tercera y cuartas órdenes establecen el filtrado específico que se desea. La cuarta orden permite que pasen los datagramas que provengan de o vayan a los servidores web de fuera de nuestra red, y la tercera red impide las conexiones entrantes de TCP con un puerto de origen igual a 80.

Si ahora se desea añadir reglas que permitan el modo pasivo sólo como modo de acceso a los servidores de FTP de fuera de nuestra red, se añadirían estas reglas:

```
# ipchains -A forward -s 0/0 20 -d 172.16.1.0/24 -p tcp -y -j DENY
    # ipchains -A forward -s 172.16.1.0/24 -d 0/0 20 -p tcp -b -j ACCEPT
    # ipchains -A forward -s 0/0 21 -d 172.16.1.0/24 -p tcp -y -j DENY
    # ipchains -A forward -s 172.16.1.0/24 -d 0/0 21 -p tcp -b -j ACCEPT
```

9.7.4. Listado de nuestras reglas con ipchains

Para mostrar nuestras reglas con **ipchains**, se utiliza su argumento `-L`. De igual forma que con **ipfwadm**, existen argumentos que controlan el grado de detalle de la salida. En su forma simple, **ipchains** produce una salida que se parece a ésta:

```
# ipchains -L -n
    Chain input (policy ACCEPT):
    Chain forward (policy DENY):
    target     prot opt     source              destination         ports
    DENY       tcp  -y----  0.0.0.0/0           172.16.1.0/24       80 ->   *
    ACCEPT     tcp  ------  172.16.1.0/24       0.0.0.0/0           * ->   80
    ACCEPT     tcp  ------  0.0.0.0/0           172.16.1.0/24       80 ->   *
    ACCEPT     tcp  ------  172.16.1.0/24       0.0.0.0/0           * ->   20
    ACCEPT     tcp  ------  0.0.0.0/0           172.16.1.0/24       20 ->   *
    ACCEPT     tcp  ------  172.16.1.0/24       0.0.0.0/0           * ->   21
    ACCEPT     tcp  ------  0.0.0.0/0           172.16.1.0/24       21 ->   *

    Chain output (policy ACCEPT):
```

Si no se proporciona el nombre de la cadena que se desea mostrar, **ipchains** mostrará todas las reglas de todas las cadenas. El argumento `-n` de nuestro ejemplo le dice a **ipchains** que no intente convertir ninguna dirección ni puerto en nombres. La información que presenta debería ser auto−explicativa.

Un formato de salida más explícito, que se invoca con la opción `-u`, proporciona mucho más detalle. Esta salida añade campos para los contadores de bytes y de datagramas, el tipo de servicio, los identificadores *AND* y *XOR*, el nombre de la interfaz, la marca, y el tamaño de la salida.

Todas las reglas creadas con **ipchains** tienen contadores de datagramas y bytes asociadas con ellas. Así es cómo se implementa la auditoría de IP y se discutirá con detalle en Capítulo 10. Por defecto, estos contadores se presentan de forma aproximada utilizando los sufijos `K` y `M`, que representan las unidades de mil y de millón, respectivamente. Si se proporciona el argumento `-x`, entonces se muestran los contadores en su forma completa y expandida.

9.7.5. Uso avanzado de las cadenas

Usted ya sabe que la orden **ipchains** sustituye a la orden **ipfwadm** con una sintaxis de línea de órdenes más simple y con algunas mejoras interesantes, pero sin duda alguna, usted desea saber dónde y por qué se deben utilizar las cadenas de usuario. Probablemente, también desee saber cómo utilizar los guiones de soporte que acompañan a la orden **ipchains** en su paquete de 'software'. Se investigarán a continuación estas materias y se dará respuesta a esas cuestiones.

9.7.5.1. Cadenas de usuario

Los tres conjuntos de reglas del código del cortafuegos de IP tradicional proporcionan un mecanismo para construir configuraciones de cortafuegos que eran bastante simples de entender y de gestionar en el caso de pequeñas redes con requisitos simples en cuanto a funcionalidad de cortafuegos. Cuando los requisitos de configuración no son tan simples, aparecen numerosos problemas. En primer lugar, las redes muy grandes requieren con frecuencia un número mucho mayor de reglas de cortafuegos que el pequeño que hemos visto hasta ahora; de forma inevitable, aparecen necesidades que requieren que se añadan reglas de cortafuegos para cubrir escenarios con casos especiales. Cuando el número de reglas empieza a crecer, el rendimiento del cortafuegos disminuye más y más según más y más comprobaciones tienen que realizarse sobre cada datagrama y la facilidad de gestión se convierte en un asunto importante. En segundo lugar, no es posible habilitar y deshabilitar conjuntos de reglas atómicamente; en cambio, usted se encontrará inevitablemente expuesto a ataques mientras se encuentre en medio de una reconstrucción de sus conjuntos de reglas.

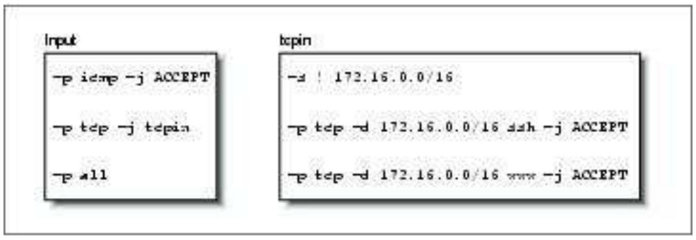
El diseño del cortafuegos 'IP Chains' ayuda a soliviantar estos problemas al permitir al administrador de la red crear conjuntos arbitrarios de reglas de cortafuegos que se pueden enlazar con los tres conjuntos de reglas predefinidas. Se puede utilizar la opción `−N` de **ipchains** para crear una nueva cadena con el nombre de ocho caracteres o menos que nos plazca. (Probablemente sea buena idea restringir el nombre a uno formado por minúsculas solamente). La opción `−j` configura la acción que se tomará cuando el datagrama coincida con la especificación de la regla. La opción `−j` especifica que si un datagrama coincide con una regla, entonces deben realizarse más comprobaciones contra una cadena definida por usuario. Se ilustrará esto con un diagrama.

Considérese las siguientes órdenes de **ipchains**:

```
ipchains −P input DENY
ipchains −N tcpin
ipchains −A tcpin −s ! 172.16.0.0/16
ipchains −A tcpin −p tcp −d 172.16.0.0/16 ssh −j ACCEPT
ipchains −A tcpin −p tcp −d 172.16.0.0/16 www −j ACCEPT
ipchains −A input −p tcp −j tcpin
ipchains −A input −p all
```

Se establece la política por defecto de la cadena de entrada a `deny`. La segunda orden crea una cadena de usuario denominada tcpin. La tercera orden añade una regla a la cadena `tcpin` que coincide con cualquier datagrama cuyo origen esté fuera de nuestra red; la regla no representa ninguna acción. Esta regla es una regla de auditoría que se discutirá con más detalle en Capítulo 10. Las dos reglas siguientes coinciden con cualquier datagrama destinado a nuestra red local tanto al puerto de `ssh` como al de `www`; los datagramas que coincidan con estas reglas son aceptados. La magia de *ipchains* empieza en la regla siguiente. Obliga al 'software' del cortafuegos a que compruebe cualquier datagrama del protocolo de TCP contra la cadena de usuario `tcpin`. Por último, se añade una regla a la cadena `input` que coincide con cualquier datagrama; esto es otra regla de auditoría. Todo esto producirá la cadenas de cortafuegos mostradas en la Figure 9−4.

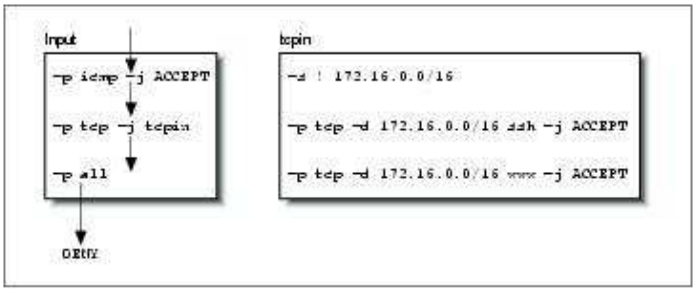
Figura 9−4. Un conjunto simple de reglas de una cadena de IP



Nuestras cadenas `input` y `tcpin` están pobladas con nuestras reglas. El procesamiento de los datagramas siempre comienza por una de las cadenas predefinidas... Veamos cómo entran en juego las cadenas de usuario siguiendo el camino de procesamiento de los diferentes tipos de datagramas.

Primero, veamos qué pasa cuando se recibe un datagrama de UDP para uno de nuestros 'hosts'. La Figura 9−5 ilustra el flujo por las reglas.

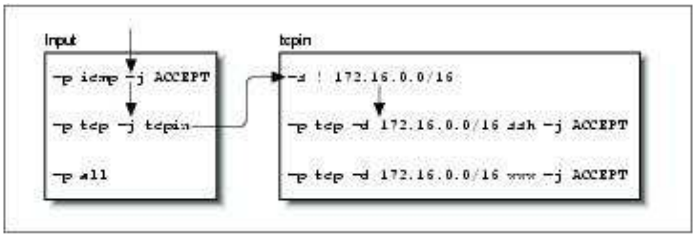
Figura 9−5. La secuencia de reglas de comprobación de un datagrama de UDP recibido



El datagrama se recibe por la cadena `input` y cae dentro de las dos reglas porque coinciden con los protocolos de ICMP y TCP, respectivamente. Coincide con la tercera regla en la cadena, pero no se especifica ningún blanco por lo que los contadores de datagramas y bytes se actualizan pero se toma ninguna otra acción. El datagrama alcanza el final de la cadena `input`, se encuentra con la política predeterminada de la cadena `input` y no se acepta.

Para ver a nuestra cadena de usuario en acción, considérese qué pasa cuando se recibe un datagrama de TCP destinado al puerto `ssh` de uno de nuestros 'hosts'. La secuencia se muestra en la Figura 9−6.

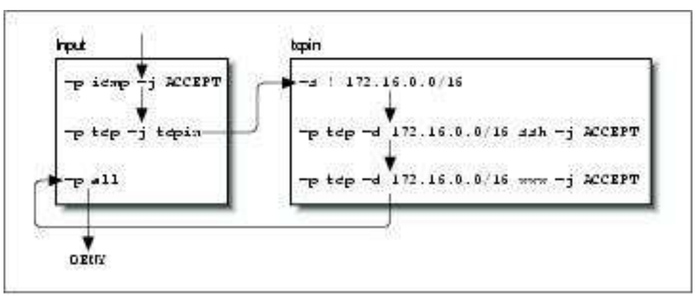
Figura 9−6. Flujo de reglas para un datagrama de TCP recibido para ssh



Esta vez, la segunda regla de la cadena `input` coincide y especifica como blanco la cadena `tcpin`, nuestra cadena de usuario. Especificar una cadena de usuario como blanco causa que se compruebe el datagrama contra las reglas de esa cadena, por lo que la siguiente regla que se comprobará será la primera regla de la cadena `tcpin`. La primera regla coincide con cualquier datagrama que tenga una dirección de origen fuera de nuestra red local y no especifica ningún blanco, por lo que también es una regla de auditoría y la comprobación pasa a la siguiente regla. La segunda regla de nuestra cadena `tcpin` sí que coincide y especifica un blanco de `ACCEPT`. Se ha llegado a un blanco tal que no se realiza más procesamiento. El datagrama se acepta.

Por último, veamos lo que pasa cuando se alcanza el final de una cadena de usuario. Para ver esto, se representará el flujo de un datagrama de TCP destinado a un puerto distinto de los dos que estamos manejando específicamente, como se muestra en la Figura 9−7.

Figura 9−7. Flujo de reglas para un datagra de TCP recibido para telnet



Las cadenas de usuario no tienen políticas por defecto. Cuando se han comprobado todas las reglas de una cadena de usuario , y ninguna coincide, el código del cortafuegos actúa como si estuviera presente una regla de `RETURN`, por lo que si no es esto lo que usted desea, debe asegurarse de proporcionar una regla al final de la cadena de usuario que tome la acción que desee. En nuestro ejemplo, la comprobación vuelve a la regla del conjunto de reglas `input` situando inmediatamente después de la que nos movió a la cadena de usuario. En algún momento, se alcanza el final de la cadena `input`, que tiene una política predeterminada y no se acepta el datagrama.

Este ejemplo era muy simple, pero sirve de ilustración. Un uso más práctico de 'IP chains' sería mucho más complejo. Un ejemplo un poco más sofisticado es el proporcionado con la siguiente lista de órdenes:

```
#
# Establece la política de reenvío por defecto a REJECT
ipchains -P forward REJECT
#
# crea nuestras cadenas de usuario
ipchains -N sshin
ipchains -N sshout
ipchains -N wwwin
ipchains -N wwwout
#
# Se asegura de que se rechazarán las conexiones provenientes por el camino incorrecto.
ipchains -A wwwin -p tcp -s 172.16.0.0/16 -y -j REJECT
ipchains -A wwwout -p tcp -d 172.16.0.0/16 -y -j REJECT
ipchains -A sshin -p tcp -s 172.16.0.0/16 -y -j REJECT
ipchains -A sshout -p tcp -d 172.16.0.0/16 -y -j REJECT
#
# se asegura que lo que alcance el final de una cadena de usuario se rechaza
ipchains -A sshin -j REJECT
ipchains -A sshout -j REJECT
ipchains -A wwwin -j REJECT
ipchains -A wwwout -j REJECT
#
# dirige los servicios de www y ssh a las cadenas de usuario relevantes
ipchains -A forward -p tcp -d 172.16.0.0/16 ssh -b -j sshin
ipchains -A forward -p tcp -s 172.16.0.0/16 -d 0/0 ssh -b -j sshout
ipchains -A forward -p tcp -d 172.16.0.0/16 www -b -j wwwin
ipchains -A forward -p tcp -s 172.16.0.0/16 -d 0/0 www -b -j wwwout
#
# Inserta nuestras reglas para buscar coincidencias con los 'hosts' en la segunda posición de
# nuestras cadenas de usuario.
ipchains -I wwwin 2 -d 172.16.1.2 -b -j ACCEPT
ipchains -I wwwout 2 -s 172.16.1.0/24 -b -j ACCEPT
ipchains -I sshin 2 -d 172.16.1.4 -b -j ACCEPT
ipchains -I sshout 2 -s 172.16.1.4 -b -j ACCEPT
ipchains -I sshout 2 -s 172.16.1.6 -b -j ACCEPT
#
```

En este ejemplo, se ha utilizado una selección de cadenas de usuario tanto para simplificar la gestión de la configuración de nuestro cortafuegos como para mejorar su eficiencia en comparación a una solución que involucrara sólo las cadenas predefinidas.

Nuestro ejemplo crea cadenas de usuario para cada uno de los servicios de `ssh` y `www` en cada sentido de la conexión. La cadena denominada `wwwout` es donde se colocan las reglas para los 'hosts' que tienen permisos para realizar conexiones de World Wide Web salientes, y `sshin` es donde se definen las reglas para los 'hosts' a los que se desea permitir las conexiones entrantes de ssh. Se asume que tenemos los requisitos de permitir o denegar a 'hosts' individuales de nuestra red la capacidad de empezar o recibir conexiones de `ssh` y `www`. La simplificación existe porque las cadenas de usuario nos permiten de forma clara agrupar las reglas para los permisos de entradas y salidas a los 'hosts' en vez de tenerlas todas revueltas. La mejora en la eficiencia se debe a que se ha reducido el número medio de comprobaciones requeridas sobre cualquier datagrama antes de que se encuentre un blanco. La ganancia de eficiencia aumenta conforme se añaden más 'hosts' Si no se hubiera utilizado cadenas de usuario, se tendría que buscar por la lista completa de reglas para determinar qué acción se toma con cada datagrama que se recibe. Incluso si se asume que cada una de las reglas de nuestra lista coincide con una proporción igual del número total de datagramas procesados, aún así estaríamos buscando en la mitad de la lista en promedio. Las cadenas de usuario nos permiten evitar la comprobación de números grandes de reglas si el datagrama que se comprueba no coincide con la regla simple en la cadena predeterminada a la que ha saltado.

9.7.5.2. Los guiones de soporte de ipchains

El paquete de 'software' de **ipchains** se proporciona con tres guiones de soporte. El primero de ellos ya se ha discutido brevemente, mientras que los dos restantes proporcian un método sencillo y conveniente de guardar y recuperar la configuración de su cortafuegos.

El guión **ipfwadm−wrapper** emula la sintaxis de la línea de órdenes de la orden **ipfwadm**, pero utiliza la orden **ipchains** para construir las reglas del cortafuegos. Esto es una forma conveniente de realizar la migración de la configuración existente de su cortafuegos o una alternativa al aprendizaje de la sintaxis de **ipchains**. El guión **ipfwadm−wrapper** se comporta de forma diferente de la orden **ipfwadm** en dos cosas: en primer lugar, porque la orden **ipchains** no soporta la especificación de una interfaz por su dirección, el guión **ipfwadm−wrapper** acepta el argumento `−V` pero intenta convertirlo en el equivalente en **ipchains** de `−W` buscando el nombre del interfaz configurado por la dirección proporcionada. El guión **ipfwadm−wrapper** le dará siempre un aviso cuando utilice la opción `−V` con el propósito de recordarle todo esto. En segundo lugar, las reglas de auditoría de los fragmentos no se traducen adecuadamente.

Los guiones **ipchains−save** e **ipchains−restore** convierten la tarea de construir y modificar la configuración del cortafuegos en una tarea mucho más simple. La orden **ipchains−save** lee la configuración actual y la escribe de forma simplificada por la salida estándar. La orden **ipchains−restore** lee datos con el formato de la salida de la orden **ipchains−save** y configura el cortafuegos de IP con esas reglas. La ventaja de utilizar estos guiones en vez de modificar directamente el guión de configuración de su cortafuegos y probar la nueva configuración consiste en la capacidad de construir dinámicamente su configuración una vez y entonces guardarla. Entonces usted puede recuperar esa configuración, modificarla, y volverla a guardar si así lo desea.

Para utilizar los guiones, se introduciría algo como:

```
ipchains−save >/var/state/ipchains/firewall.state
```

para guardar la configuración actual de su cortafuegos. Usted la recuperaría, quizás en el momento de arranque del sistema, con:

```
ipchains−restore </var/state/ipchains/firewall.state
```

El guión **ipchains−restore** comprueba si ya existe cualquiera de las cadenas de usuarios especifica en su entrada. Si se proporciona el argumento `−f`, entonces y de forma automáticaa, el guión borrará las reglas de la cadena de usuario antes de configurar las de la entrada. El comportamiento por defecto es que se le pregunte si desea saltarse esta cadena o inicializarla.

9.8. Netfilter e 'IP Tables' (Núcleos 2.4)

Mientras desarrollaba el cortafuegos 'IP Chains', Paul Russell decidió que realizar funciones de cortafuegos de IP debería ser algo menos difícil; pronto asumió como tarea simplificar los aspectos de procesamiento de datagramas en el código de cortafuegos del núcleo y produjo un esquema de filtrado que era mucho más claro y mucho más flexible. Denominó este nuevo esquema *netfilter*.

> **Nota:** En el momento de la preparación de este libro, el diseño de *netfilter* no está todavía estable. Esperamos que perdone cualquier error en la descripción de *netfilter* o de cualquiera de las herramientas de configuración asociadas que sea debido a cambios ocurridos tras la preparación de este material. Consideramos el trabajo realizado sobre *netfilter*

> suficientemente importante como para justificar la inclusión de este material, pese a que partes de él sean especulativas por sí mismas. Si tiene alguna duda, los documentos HOWTO correspondientes contendrán la información más precisa y actualizada sobre los detalles asociados con la configuración de *netfilter*.

Pero, ¿qué era lo que no estaba bien con las cadenas de IP de ipchains ? Habían aumentado de forma importante la eficiencia y la gestión de las reglas del cortafuegos. Pero la forma que tenían de procesar los datagramas eran todavía complejas, en especial en conjunción con características relacionadas con las funciones de cortafuegos como el enmascaramiento de IP (discutido en el Capítulo 11) y con otras formas de traducciones de direcciones. Parte de esta complejidad era debida a que el enmascaramiento de IP y la traducción de direcciones de red [102] fueron funciones desarrolladas independientemente del código de cortafuegos e integradas más tarde, en vez de haber sido diseñadas como partes mismas del código del cortafuegos desde el principio. Si un desarrollador deseara añadir todavía más características a la secuencia de procesamiento de datagramas, entonces se encontraría con dificultades para encontrar el lugar donde insertar el código y se habría visto obligado a realizar cambios en el núcleo.

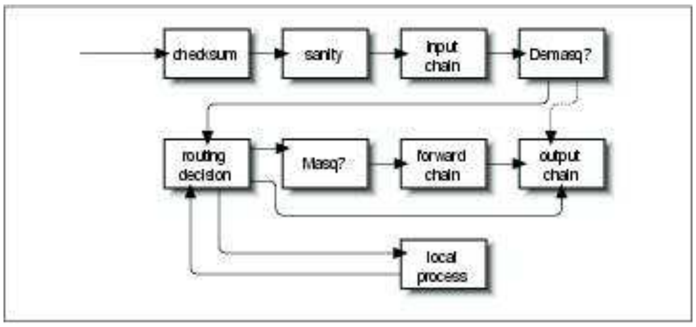
Además, había otros problemas. En concreto, la cadena input describía la entrada a la capa de red de IP tomada en conjunto. La cadena input afectaba tanto a los datagramas que estaban *destinados a* este 'host' así como los datagramas que iban a ser *encaminados*. Esto resulta contrario a la intuición porque se confundía la función de la cadena 'input' con la de la cadena 'forward', que se aplicaba sólo a los datagramas que iban a ser reenviados, pero que siempre seguía a la cadena 'input'. Se se quería tratar de forma diferente los datagramas para el propio 'host' de los que iban a ser reenviados, era necesario construir reglas complejas que excluían a unos o a otros. El mismo problema aparecía con la cadena output de salida.

Esta complejidad influía de forma inevitable en el trabajo del administrador de sistemas porque se veía reflejada en la forma en que se debían diseñar los conjuntos de reglas. Además, cualquier extensión al proceso de filtrado exigía la modificación directa del núcleo, porque todas las políticas de filtrados estaban implementadas allí y no había forma de proporcionar una interfaz transparente. *netfilter* aborda tanto la complejidad como la rigidez de las soluciones antiguas implementando un esquema genérico en el núcleo que simplifica la forma en que se procesan los datagramas y proporciona la posibilidad de extender las políticas de filtrado sin tener que modificar el núcleo.

Veamos dos de los cambios claves realizados. La Figura 9−8 ilustra cómo se procesan los datagramas en la implementación de 'IP Chains', mientras que Figura 9−9 ilustra cómo se procesan en la implementación de *netfilter*. La diferencias claves consisten en la eliminación de la función de enmascaramiento del código central y de un cambio en la localización de las cadenas de entrada y de salida. En acompañamiento a estos cambios, se creó una herramienta de configuración nueva y extensible que se denominó **iptables**.

En 'IP Chains' la cadena de entrada se aplica a todos los datagramas recibidos por el 'host', independientemente de si están destinados al 'host' local o de si serán encaminados a otro 'host'. En *netfilter*, la cadena 'input' de entrada se aplica *sólamente* a los datagramas destinados al 'host' local, y la cadena 'forward' de reenvío se aplica sólo a los datagramas destinados a *otro* 'host'. De forma similar, en 'IP chains', la cadena 'output' de salida se aplica a todos los datagramas que abadonen el 'host' local, independientemente de si el datagrama se genera en el 'host' local o ha sido encaminado desde otro 'host'. En *netfilter*, la cadena 'output' de salida se aplica *sólamente* a los datagramas generados en este 'host' y no se aplica a los datagramas que están siendo encaminados provenientes de otro 'host'. Este cambio por sí solo ofrece una enorme simplificación de muchas configuraciones de cortafuegos.

Figura 9−8. Procesamiento de datagramas en 'IP Chains'

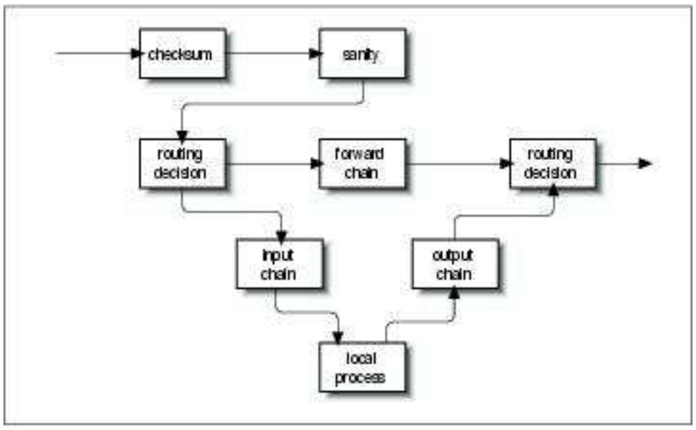


En la Figura 9−8, los componentes etiquetados como demasq y masq son componentes separados del núcleo que son responsables del procesamiento de los datagramas enmascarados entrantes y salientes. Estos componentes han sido reimplementados como módulos de *netfilter*.

Considérese el caso de una configuración para la que la política por defecto para cada una de las cadenas 'input', 'forward' y 'output' es `deny`. En 'IP Chains', se necesitarían seis reglas para permitir cualquier sesión a través del 'host' cortafuegos; dos para cada una de las cadenas 'input, 'forward' y 'output' (una cubriría el camino en un sentido y la otra en el sentido contrario). Puede imaginarse cómo esto puede llegar a resultar extremadamente complejo y difícil de gestionar cuando se mezclan sesiones que pueden ser encaminadas y sesiones que podrían conectarse al 'host' local sin que deban ser encaminadas. 'IP chains' le permite crear cadenas que le simplificarían esta tarea un poco, pero su diseño no resulta evidente y requiere de un cierto nivel de experiencia.

En la implementación de *netfilter* con **iptables**, esta complejidad desaparece completamente. Para que se pueda encaminar por un 'host' cortafuegos un servicio que se desea prohibir que termine en el propio 'host', sólo se necesitan dos reglas: una para un sentido y otra para el contrario ambas en la cadena 'forward'. Esto es la forma obvia de diseñar reglas de cortafuegos, y servirá para simplificar enormemente el diseño de las configuraciones del cortafuegos.

Figura 9−9. Cadena de procesamientos de datagramas en 'netfilter'



El documento PACKET−FILTERING−HOWTO ofrece una lista detallada de los cambios que se han realizado, por lo que aquí nos vamos a centrar en los aspectos más prácticos.

9.8.1. Compatibilidad hacia atrás con ipfwadm e ipchains

La flexibilidad notoria de *netfilter* en GNU/Linux queda ilustrada por su habilidad para emular las interfaces **ipfwadm** e **ipchains**. La emulación hace un poco más sencilla la transición a la nueva generación del software de cortafuegos.

Los dos módulos de *netfilter* del núcleo denominados `ipfwadm.o` e `ipchains.o` proporcionan la compatibilidad hacia atrás para **ipfwadm** e **ipchains**. Sólo puede cargarse uno de estos módulos a la vez, y utilizarlo sólo si el módulo `ip_tables.o` no está cargado. Cuando se ha cargado el módulo apropiado, entonces *netfilter* funciona exactamente de la misma forma que la anterior implementación del cortafuegos.

netfilter imita la interfaz de **ipchains** con las siguientes órdenes:

```
    rmmod ip_tables
    modprobe ipchains
    ipchains ...
```

9.8.2. Uso de iptables

La utilidad **iptables** se utiliza para configurar las reglas de filtrado de *netfilter*. Su sintaxis se apoya fuertemente en la de la orden **ipchains**, pero difiere en un aspecto muy importante: es *extensible*. Esto quiere decir que su funcionalidad puede extenderse sin tener que recompilar. Consigue este truco utilizando bibliotecas compartidas. Hay extensiones estándares de las que se explorarán algunas dentro de un momento.

Antes de que se pueda utilizar la orden **iptables**, se debe cargar el módulo del núcleo de *netfilter* que proporciona el soporte para ello. La forma más fácil de hacerlo es con la orden **modprobe**:

```
    modprobe ip_tables
```

La orden **iptables** se utiliza para configurar tanto el filtrado de IP como la traducción de direcciones de red. Para facilitar esto, existen dos tablas de reglas denominadas *filter* y *nat*. Por defecto, se asume la tabla 'filter' salvo que se especifique la opción `−t`. También se proporciona cinco cadenas predefinidas. Las cadenas `INPUT` y `FORWARD` están disponibles para la tabla `filter`, las cadenas `PREROUTING` y `POSTROUTING` están disponbiles para la tabla `nat` , y la cadena `OUTPUT` está disponible para ambas tablas. En este capítulo se discutirá solamente la tabla *filter*. Se contemplará la tabla *nat* en el Capítulo 11

La sintaxis general de la mayoría de las órdenes de **iptables** es:

```
iptables orden especificación_de_regla extensiones
```

Veamos alguna de las opciones con detalle, y después se revisarán algunos ejemplos.

9.8.2.1. Órdenes

Existen varias formas de manipular las reglas y los conjuntos de reglas con la orden **iptables**. Las relevantes para la función de cortafuegos de IP son:

−A cadena
: Añade una o más reglas al final de la cadena especificada. Si se proporciona un nombre de 'host' tanto como origen como destino y se resuelve a más de una dirección IP, se añadirá una regla por cada una de esas direcciones.

−I cadena número_de_regla
: Inserta una o más reglas al comienzo de la cadena especificada. De nuevo, si se proporciona un nombre de 'host' en la especificación de la regla, se añadirá una regla por cada una de las direcciones que se resuelvan.

−D cadena
: Borra de la cadena especificada una o más reglas que coincidan con la especificación de regla de la cadena especificada.

−D cadena número_de_regla

Borra la regla que ocupa la posición `número_de_regla` en la cadena especificada. Las posiciones de reglas comienzan en el 1 para la primera regla de la cadena.

−R cadena número_de_regla
: Reemplaza la regla que ocupa la posición `número_de_regla` en la cadena especificada por la regla proporcionada en la especificación.

−C cadena
: Comprueba el datagrama descrito por la especificación de la regla contra la cadena especificada. Esta orden devolverá un mensaje que describe cómo el datagrama procesa la cadena. Resulta de la mayor utilidad para las pruebas de la configuración del cortafuegos por lo que se contemplará con más detalle más adelante.

−L [cadena]
: Muestra las reglas de la cadena especificada, o de todas las cadenas si no se especifica ninguna.

−F [cadena]
: Borra todas las reglas de la cadena especificada, o de todas las cadenas si no se especifica ninguna.

−Z [cadena]
: Pone a cero los contadores de datagramas y bytes en todas las reglas de la cadena especificada, o de todas las cadenas si no se especifica ninguna.

−N cadena
: Crea una nueva cadena con el nombre especificado. No puede existir antes una cadena del mismo nombre. Así es cómo se crean las cadenas de usuario.

−X [cadena]
: Borra la cadena de usuario especificada, o todas las cadenas de usuario si no se especifica ninguna. Para que este comando tenga éxito, no deben existir referencias a la cadena especificada desde cualquier otra cadena de reglas.

−P cadena política
: Establece la política por defecto de la cadena especificada a la política especificada. Las políticas válidas de cortafuegos son `ACCEPT`, `DROP`, `QUEUE`, y `RETURN`. `ACCEPT` permite pasar a los datagramas. `DROP` causa que el datagrama sea descartado. `QUEUE` causa que el datagrama sea pasado al espacio de usuario para posterior procesamiento. La política `RETURN` causa que el código del cortafuegos de IP vuelva a la cadena que llamó a la que contenía esta regla, y que continúe con la regla situada tras la regla desde la que se hizo la llamada.

9.8.2.2. Parámetros de especificación de reglas

Existe una serie de parámetros de **iptables** que constituyen la especificación de una regla. Donde se requiera la especificación de una regla, se debe proporcionar algunos de esos parámetros o se asumirá sus valores por defecto.

−p [!]protocolo
: Especifica el protocolo del datagrama que buscar coincidencias coná con esta regla. Los nombres válidos de protocolos son `tcp`, `udp`, `icmp`, o un número, si se conoce el número del protocolo de IP.[103] Por ejemplo, podría utilizarse un `4` para buscar coincidencias con el protocolo de encapsulamiento `ipip`. Si se proporciona el signo `!`, entonces se niega la regla y el datagrama buscar coincidencias coná con cualquier protocolo diferente del especificado. Si no se proporciona este

parámetro, se asume por omisión la coincidencia con todos los protocolos.

−s [!]dirección[/máscara]
: Especifica la dirección de origen del datagrama que buscar coincidencias coná con esta regla. Se puede proporcionar la dirección como un nombre de 'host', como un nombre de red o como una dirección de IP. El parámetro opcional `máscara` es la máscara de red que se utilizará y puede proporcionarse tanto en la forma tradicional (e.g., /255.255.255.0) como en la forma moderna (e.g., /24).

−d [!]dirección[/máscara]
: Especifica la dirección de destino del datagrama que buscar coincidencias coná con esta regla. La codificación de este parámetro es la misma que la del parámetro `−s`.

−j blanco
: Especifica qué acción se tomará cuando se coincida con esta regla. Puede pensarse en este parámetro como con el significado de salta a. Los blancos válidos son `ACCEPT`, `DROP`, `QUEUE`, y `RETURN`. Se describieron sus significados más arriba. Sin embargo, también puede proporcionarse el nombre de una cadena de usuario, que sería por donde el proceso continuaría. También puede proporcionarse el nombre de un blanco complementado con el de una extensión. Se hablará acerca de las extensiones en breve. Si se omite este parámetro, no se realizará ninguna acción sobre los datagramas coincidentes, excepto la actualización de los contadores de datagramas y bytes de esta regla.

−i [!]nombre_de_interfaz
: Especifica la interfaz por la que se recibió el datagrama. De nuevo, el signo `!` invierte el resultado de la coincidencia. Si el nombre de la interfaz acaba con un signo +entonces cualquier interfaz que comience con la cadena proporcionada buscar coincidencias coná. Por ejemplo, `−i ppp+` buscar coincidencias coná con cualquier dispositivo de red de PPP y `−i ! eth+` con todas las interfaces excepto las correspondientes a dispositivos de Ethernet.

−o [!]nombre_de_interfaz
: Especifica la interfaz por la que se enviará el datagrama. Este argumento tiene la misma codificación que el argumento `−i`.

[!] −f
: Especifica que esta regla se aplica al segundo y restantes fragmentos de un datagrama fragmentado, y no al primer fragmento.

9.8.2.3. Opciones

Las siguientes opciones de **iptables** son más generales por naturaleza propia. Algunas de ellas controlan características bastante esotéricas del software de *netfilter*.

−v
: Hace que **iptables** sea más explícito en su salida. Proporcionará más información.

−n
: Hace que **iptables** muestre las direcciones de IP y los números de puertos en forma de números sin intentar resoverlos contra sus correspondientes nombres.

−x
: Fuerza que los números de salida de **iptables** parezcan con sus valores exactos sin ninguna aproximación.

− −line−number

Causa que se muestren los números de línea en los listados de los conjuntos de reglas. El número de línea corresponderá con la posición de la regla dentro de la cadena.

9.8.2.4. Extensiones

Se dijo antes que la utilidad **iptables** es extensible a través de módulos de bibliotecas compartidas. Existen alguna extensiones estándares que proporciona algunas de las características que **ipchains** proporcionaba. Para utilizar una extensión, se debe especificar su nombre con el argumento `−m` *`nombre`* de **iptables**. La lista siguiente muestra las opciones `−m` y `−p` que establecen el contexto de la extensión, y las opciones proporcionadas por esa extensión.

9.8.2.4.1. Extensiones de TCP: utilizadas con `−m tcp −p tcp`

− −sport [!] [puerto[:puerto]]
: Especifica el puerto que debe utilizar el origen del datagrama para buscar coincidencias con esta regla. Se pueden especificar los puertos en la forma de un rango, especificando los límites inferior y superior con un signo **:** como delimitador. Por ejemplo, `20:25` describe todos los puertos que van desde el 20 hasta el 25 incluyendo ambos. De nuevo, el signo `!` puede utilizarse para negar los valores.

− −dport [!] [puerto[:puerto]]
: Especifica el puerto que el datagrama de destino utilizará para buscar coincidencias con este regla. Este argumento se codifica de forma idéntica a la opción `− −sport`.

− −tcp−flags [!] máscara comp
: Especifica que esta regla debe buscar coincidencias con cuando los indicadores de TCP del datagrama coincidan con los especificados por *`máscara`* y *`comp`*. *`máscara`* es una lista separada por comas de los indicadores que deben examinarse en la comprobación. *`comp`* es una lista separada por comas de indicadores cuyo valores han de ser todos 1 para que la regla coincida. Los indicadores válidos son: *SYN*, *ACK*, *FIN*, *RST*, *URG*, *PSH*, *ALL* o *NONE*. Esto constituye una opción avanzada: consúltese una buena descripción del protocolo de TCP, como el documento RFC−793, para la explicación del significado y la implicación de cada uno de estos indicadores. El signo `!` niega la regla.

[!] − −syn
: Especifica que la regla buscar coincidencias conná sólo con los datagramas cuyo bit `SYN` valga 1 y cuyos bits `ACK` y `FIN` valgan ambos 0. Los datagramas con estos valores de los indicadores se utilizan para abrir las conexiones de TCP, por tanto esta opción puede ser utilizada para gestionar las solicitudes de conexión. Esta opción es una abreviatura de:

```
− −tcp−flags SYN,RST,ACK SYN
```

Cuando se utilice el operador de negación, la regla buscar coincidencias conná con todos los datagramas cuyos bits `SYN` y `ACK` no valgan 1 simultáneamente.

9.8.2.4.2. Extensions de UDP: utilizadas con −m udp −p udp

− −sport [!] [port[:port]]
: Especifica el puerto que debe utilizar el origen del datagrama para buscar coincidencias con esta regla. Se pueden especificar los puertos en la forma de un rango, especificando los límites inferior y superior con un signo **:** como delimitador. Por ejemplo, `20:25` describe todos los puertos que van desde el 20 hasta el 25 incluyendo ambos. De nuevo, el signo `!` puede utilizarse para negar los valores.

– –dport [!] [port[:port]]
: Especifica el puerto que el datagrama de destino utilizará para buscar coincidencias con esta regla. Este argumento se codifica de forma idéntica a la opción `– -sport`.

9.8.2.4.3. Extensiones de ICMP: utilizadas con `–m icmp -p icmp`

– –icmp–type [!] nombre_de_tipo
: Especifica el tipo de mensaje de ICMP que buscar coincidencias coná con esta regla. Puede especificarse el tipo tanto por su número como por su nombre. Algunos nombres válidos son: `echo–request`, `echo–reply`, `source–quench`, `time–exceeded`, `destination–unreachable`, `network–unreachable`, `host–unreachable`, `protocol–unreachable`, y `port–unreachable`.

9.8.2.4.4. Extensiones de MAC: utilizadas con `–m mac`

– –mac–source [!] address
: Especifica la dirección Ethernet del 'host' transmisor que coincide con esta regla. Esto sólo tiene sentido en una regla de la cadena de entrada 'input' o de reenvío 'forward' porque se trasmitirá cualquier datagrama que pase la cadena de salida 'output'.

9.8.3. Nuestro ejemplo simple revisado otra vez

Para implementar nuestro ejemplo simple con *netfilter*, simplemente se podría cargar el módulo `ipchains.o` para que se comporte como la versión **ipchains**. En cambio, se volverá a realizar una implementación con **iptables** para ilustrar lo similar que es.

De nuevo, supóngase que se dispone en nuestra organización de una red y que se está utilizando una máquina cortafuegos basada en GNU/Linux para permiir a nuestros usuarios que sean capaces de acceder a servidores WWW de Internet, y para impedir el paso de cualquier otro tipo de tráfico.

Si nuestra red tiene una máscara de red de 24 bits (clase C) y tiene como dirección 172.16.1.0, entonces se podría utilizar las siguientes reglas de **iptables**:

```
# modprobe ip_tables
    # iptables –F FORWARD
    # iptables –P FORWARD DROP
    # iptables –A FORWARD –m tcp –p tcp –s 0/0 --sport 80 –d 172.16.1.0/24 /
    --syn –j DROP
    # iptables –A FORWARD –m tcp –p tcp –s 172.16.1.0/24 --sport /
    80 –d 0/0 –j ACCEPT
    # iptables –A FORWARD –m tcp –p tcp –d 172.16.1.0/24 --dport 80 –s 0/0 –j /
    ACCEPT
```

En este ejemplo, las órdenes de **iptables** se interpretan exactamente como sus equivalentes de **ipchains**. La diferencia más importante es que debe cargarse el módulo `ip_tables.o`. Nótese que **iptables** no soporta la opción `–b`, por lo que debe proporcionarse una regla para cada sentido.

9.9. Manipulación de los bits de TOS

Los bits del campo de tipo de servicio (TOS [104]) son un conjunto de cuatro indicadores de un bit de la cabecera de IP. Si uno de estos indicadores de bit vale 1, un encaminador puede manipular el datagrama de forma diferente del caso en el que ningún indicador valiera 1. Cada uno de los cuatro bits tiene un propósito diferente y sólo uno de los bits de TOS puede valer 1 al mismo tiempo, es decir, las combinaciones no están permitidas. Estos indicadores de bit se denominan de "tipo de servicio" porque permiten que la aplicación que transmite los datos informe a la red del tipo de servicio de red que requiere.

Las clases de servicios de red disponibles son:

Demora mínima
: Se utiliza cuando se le da la máxima importancia al tiempo de viaje de un datagrama del 'host' de origen al 'host' de destino (demora). Por ejemplo, un suministrador de red podría estar utilizando tanto conexiones de red de fibra como por satélite. Los datos transportados por las conexiones por satélite tienen que viajar más lejos y su demora entre los mismos extremos será por lo general mayor que la de las conexiones de red terrestres. Un suministrador de red podría elegir asegurarse que los datagramas con este tipo de servicio no se transporten por satélite.

Rendimiento máximo
: Se utiliza cuando el volumen de datos transmitidos en cualquier período de tiempo es importante. Existen numerosos tipos de aplicaciones de red para las que el tiempo de demora no es muy importante pero el rendimiento sí que lo es; por ejemplo, las transferencias de ficheros en bloque. Un suministrador de red podría elegir encaminar los datagramas con este tipo de servicio vía rutas de demora alta, pero de gran ancho de banda, como las conexiones por satélite.

Fiabilidad máxima
: Se utiliza cuando es importante tener alguna certeza de que los datos llegarán al destino sin necesidad de una retransmisión. El protocolo IP puede transportarse sobre un número variado de medios de transmisión de bajo nivel. Mientras que SLIP y PPP son adecuados para protocolos de enlace de datos [105], no son tan fiables para transportar IP como otras redes, como las redes X.25. Un suministrador de red podría tener disponible una red alternativa, que ofreciera alta fiabilidad para transportar IP y que se utilizaría cuando se eligiera este tipo de servicio.

Coste mínimo
: Se utiliza cuando resulta importante minimizar el coste de los datos transmitidos. El alquiler de ancho de banda de un satélite para una transmisión transoceánica cuesta generalmente menos que el alquier de espacio de un cable de fibra óptica sobre la misma distancia, por lo que los suministradores de red pueden elegir proporcionar ambos y cobrar de forma diferente según sea el que se utilice. En este escenario, el bit de tipo de servicio de coste mínimo puede ocasionar que los datagramas sean encaminados vía la ruta de menor coste por satélite.

9.9.1. Establecimiento de los bits de TOS con ipfwadm o ipchains

Las órdenes **ipfwadm** e **ipchains** gestionan los bits de TOS prácticamente de la misma forma. En ambos casos, se especifica una regla que coincide con los datagramas que contengan un conjunto particular de bits de TOS, y en ambos se utiliza el argumento `-t` para especificar los cambios que se desean realizar.

Los cambios se especifican utilizando máscaras de dos bits. Se realiza un AND lógico con la primera de estas máscaras de bits y el campo de las opciones de IP, mientras que se realiza un OR exclusivo con la segunda. Por si puede parecer complicado, en un momento se darán las recetas que se necesitan para habilitar cada uno de los tipos de servicio.

Las máscaras de bit se especifican utilizando valores hexadecimales de ocho bits. Tanto **ipfwadm** como **ipchains** utilizan la misma sintaxis para el argumento:

```
-t máscara_and máscara_xor
```

Afortunadamente, pueden utilizarse los mismo argumentos de máscara cada vez que se desee establececer un tipo de servicio particular, lo que ahorra el tener que averiguarlas cada vez. En la Tabla 9−3 se presentan junto con algunas sugerencias de uso.

Tabla 9−3. Sugerencias de uso de las máscaras de bits de TOS

TOS	ANDmask	XORmask	Uso sugerido
Demora mínima	`0x01`	`0x10`	ftp, telnet, ssh
Rendimiento máximo	`0x01`	`0x08`	ftp−data, www
Fiabilidad máxima	`0x01`	`0x04`	snmp, dns
Coste mínimo	`0x01`	`0x02`	nntp, smtp

9.9.2. Establecimiento de los bits de TOS con iptables

La herramienta **iptables** permite especificar reglas que capturen sólo los datagramas con los bits de TOS coincidentes con los valores prederminados por la opción `−m tos`, y que establezcan los bits de TOS de los datagramas de IP que coincidan con una regla con el blanco `−j TOS`. Pueden establecerse los bits de TOS sólamente en las cadenas `FORWARD` y `OUTPUT`. La coincidencia y el establecimiento ocurren de forma independiente. Se puede configurar todo tipo de reglas interesante. Por ejemplo, puede configurarse una regla que descarte todos los datagramas con una cierta combinación de bits de TOS, o una regla que establezca los bits de TOS sólamente de los datagramas que provengan de unos ciertos 'hosts'. La mayoría de las veces, se utilizarán reglas que contengan tanto la coincidencia como el establecimiento para así realizar traducciones de bits de TOS, como ya se podía hacer con **ipfwadm** o **ipchains**.

En lugar de la configuración complicada de dos máscaras de **ipfwadm** e **ipchains**, **iptables** utiliza la estrategia más simple de especificar de forma plana los bits de TOS que deben buscar coincidencias con, o los bits de TOS que deben establecerse. Además, en lugar de tener que recordar y utilizar el valor hexadecimal, puede especificarse los bits de TOS utilizando los nombres mnemotécnicos mostrados en la tabla siguiente.

La sintaxis general utilizada para buscar coincidencias con los bits de TOS se parece a:

```
−m tos −−tos nombre_mnemotécnico [otros_args] −j objetivo
```

La sintaxis general utilizada para establecer los bits de TOS se parece a:

```
[otros_args] −j TOS −−set nombre_mnemotécnico
```

Recuérdese que las dos posibilidades generalmente se utilizarán juntas, pero que también pueden ser utilizadas de forma independiente si se quiere una configuración que así lo requiera.

Mnemónico	Hexadecimal
Normal−Service[a]	0x00
Minimize−Cost[b]	0x02
Maximize−Reliability[c]	0x04
Maximize−Throughput[d]	0x08
Minimize−Delay[e]	0x10

Notas:
a. N. del T.: "servicio normal"
b. N. del T.: "minimizar los costes"
c. N. del T.: "maximizar la fiabilidad"
d. N. del T.: "maximizar el rendimiento"
e. N. del T.: "minimizar la demora"

9.10. Comprobación de una configuración del cortafuegos

Después de haber diseñado una configuración de cortafuegos adecuada, es importante comprobar que efectivamente se obtiene lo que se deseaba. Una forma de hacerlo consiste en utilizar un 'host' de prueba fuera de nuestra red para que intente atravesar su cortafuegos: esto puede llegar a resultar farragoso y lento, y se estaría limitado a la comprobación únicamente de aquellas direcciones que realmente puedan usarse.

Un método más rápido y sencillo está disponible con la implementación del cortafuegos de GNU/Linux. Permite generar pruebas y ejecutarlas contra el cortafuegos como si se estuviera haciendo la prueba con datagramas reales. Todas las variedades de software del cortafuegos del núcleo de GNU/Linux, **ipfwadm**, **ipchains**, e **iptables**, dan soporte a este tipo de comprobaciones. La implementación involucra el uso de la orden de *comprobación* relevante.

El procedimiento general de comprobación es como sigue:

1. Diseñe y configure su cortafuegos utilizando **ipfwadm**, **ipchains**, o **iptables**.

2. Diseñe una serie de comprobaciones que determinen si su cortafuegos está realmente funcionando como deseaba. Puede utilizar cualquier dirección de origen o destino para realizar estas comprobaciones, por lo que escoja algunas combinaciones de direcciones que deberían ser aceptadas y otras que deberían ser rechazadas. Si se va a permitir o prohibir a un cierto rango de direcciones, es una buena idea comprobar las direcciones situadas justo en los límites del rangouna dirección justo dentro del rango y otra justo fuera del rango. Esto le ayudará a asegurarse de que configuró correctamente los rangos, pues a veces resulta muy fácil especificar de forma incorrecta las máscaras de su configuración. Si se filtra por número de protocolo y puerto, sus comprobaciones también deberían comprobar todas las combinaciones importantes de estos parámetros. Por ejemplo, si se desea aceptar TCP sólo en ciertas circunstancias, compruebe que se rechazan los datagramas de tipo UDP.

3. Desarrolle reglas de **ipfwadm**, **ipchains**, o **iptables** para implementar cada comprobación. Probablemente merezca la pena escribir todas estas reglas en un guión de tal forma que pueda hacer y rehacer la comprobación fácilmente a la vez que va corrigiendo los errores o cambiando el diseño. Las comprobaciones utilizan casi la misma sintaxis que las especificaciones de reglas, pero los argumentos tiene significados ligeramente diferentes. Por ejemplo, el argumento de dirección de origen de una especificación de una regla especifica la dirección de origen que deberían tener los datagramas coincidentes con esta regla. En cambio, el argumento de dirección de origen en la sintaxis de comprobación específica la dirección de origen del datagrama de prueba que se generará. En el caso de **ipfwadm**, debe utilizarse la opción `c` para especificar que la orden es una comprobación, mientras que en el caso de **ipchains** e **iptables**, se debe utilizar la opción `C`. En todos los casos *siempre* se deben especificar la dirección de origen, la dirección de destino, el protocolo y la interfaz que se utilizará como prueba. El resto de argumentos, como los números de puertos y los valores de los bits de TOS, son opcionales.

4. Ejecute cada orden de comprobación y anote el resultado. El resultado de cata comprobación consistirá en una única palabra que indicará el blanco final del datagrama después de haber cruzado la configuración del cortafuegos; es decir, dónde terminará el proceso. Para el caso de **ipchains** e **iptables**, pueden comprobarse las cadenas de usuario además de las predefinidas.

5.

> Compare la salida de cada comprobación contra el resultado deseado. Si encuentra alguna discrepancia, necesitará analizar su conjunto de reglas para determinar dónde cometió el error. Si escribió sus órdenes de pruebas en un fichero de guión, entonces podrá reejecutar la comprobación de forma fácil después de haber corregido cualquier error de la configuración del cortafuegos. Se considera buena práctica borrar por completo sus conjuntos de reglas y reconstruirlas desde cero, en vez de estar realizando cambios dinámicamente. Esto le ayudará a asegurarse de que la configuración activa que está comprobando refleja el conjunto de órdenes de su guión de configuración.

Veamos lo que podría ser la transcripción de una comprobación manual de nuestro ejemplo simple con **ipchains**. Recuerde que la red local del ejemplo era 172.16.1.0 con una máscara de red de 255.255.255.0, y que se permitían las conexiones de TCP hacia servidores web de fuera de la red. La cadena de reenvío 'forward' no permitía pasar nada más. Empiece con una transmisión que sabemos debería funcionar, una conexión desde el 'host' local a un servidor web de fuera:

```
# ipchains −C forward −p tcp −s 172.16.1.0 1025 −d 44.136.8.2 80 −i eth0
accepted[106]
```

Fíjese en los argumentos que se han proporcionado y en la forma en que se utilizan para describir el datagrama. La salida de la orden indica que se aceptó el datagrama para su reenvío, que es lo que se esperaba.

Ahora hagamos otra prueba, esta vez con una dirección de origen que no pertenece a nuestra red, y, que por tanto, debería ser rechazada.

```
# ipchains −C forward −p tcp −s 172.16.2.0 1025 −d 44.136.8.2 80 −i eth0
denied[107]
```

Realice algunas comprobaciones más, con el mismo nivel de detalle que la primera comprobación pero con diferentes protocolos. También deberían ser rechazados:

```
# ipchains −C forward −p udp −s 172.16.1.0 1025 −d 44.136.8.2 80 −i eth0
denied
# ipchains −C forward −p icmp −s 172.16.1.0 1025 −d 44.136.8.2 80 −i eth0
denied
```

Pruebe con otro puerto de destino, de nuevo esperando que sea rechazado:

```
# ipchains −C forward −p tcp −s 172.16.1.0 1025 −d 44.136.8.2 23 −i eth0
denied
```

Tendrá que recorrer un largo camino para estar tranquilo pues tendrá que diseñar una serie de comprobaciones muy exhaustiva. Aunque a veces esto resulta tan complicado como el diseño de la propia configuración del cortafuegos, también es la mejor forma de saber si su diseño proporciona la seguridad que esperaba.

9.11. Un ejemplo de configuración del cortafuegos

Se han discutido los fundamentos de la configuración del cortafuegos. Veamos ahora qué aspecto tendría una configuración real del cortafuegos.

Se ha diseñado la configuración de este ejemplo con vistas a que sea fácilmente extensible y personalizable. Se proporcionan tres versiones. la primera se implementa con la orden **ipfwadm** (o el guión **ipfwadm−wrapper**), la segunda utiliza **ipchains**, y la tercera **iptables**. En el ejemplo no se intenta aprovechar las posibilidades de las cadenas de usuario, pero le mostrará las similitudes y diferencias entre las sintaxis de las viejas y las nuevas herramientas de configuración:

```
#!/bin/bash
##########################################################################
# VERSIÓN PARA IPFWADM
# Esta configuración está pensada como ejemplo de configuración de
# un cortafuegos sobre un 'host' único que no hospede él mismo ningún
# servicio
```

```
##########################################################################

# SECCIÓN CONFIGURABLE POR EL USUARIO

# El nombre y la localización de la utilidad ipfwadm. Utilice
# ipfwadm-wrapper para los núcleos 2.2.*.
IPFWADM=ipfwadm

# Ruta del ejecutable de ipfwadm.
PATH="/sbin"

# El espacio de direcciones de nuestra red interna y el dispositivo
# de red que la soporta.
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Las direcciones de fuera y el dispositivo de red que la soporta.
ANYADDR="0/0"
ANYDEV="eth1"

# Los servicios de TCP que deseamos permitir que pasen - un "" vacío
# significa todos los puertos
# nota: separados por espacios
TCPIN="smtp www"
TCPOUT="smtp www ftp ftp-data irc"

# Los servicios de UDP que deseamos permitir que pasen - un "" vacío
# significa todos los puertos
# nota: separados por espacios
UDPIN="domain"
UDPOUT="domain"

# Los servicios de ICMP que deseamos permitir que pasen - un "" vacío
# significa todos los tipos
# referencia para los números de los tipos: /usr/include/netinet/ip_icmp.h
# nota: separados por espacios
ICMPIN="0 3 11"
ICMPOUT="8 3 11"

# Registro; descomente la siguiente línea para habilitar el registro
# de los datagramas rechazados por el cortafuegos
# LOGGING=1

# FIN DE LA SECCIÓN CONFIGURABLE POR EL USUARIO
###########################################################################
# Borra las reglas de la cadena de entrada
$IPFWADM -I -f

# Por defecto, queremos denegar el acceso a los intentos de entrada
$IPFWADM -I -p deny

# SUPLANTACIÓN DE IDENTIDAD
# No se debería aceptar ningún datagrama proveniente de fuera con una
# direccción de origen coincidente con una de las nuestras, por
# eso las rechazamos.
$IPFWADM -I -a deny -S $OURNET -W $ANYDEV

# 'SMURF'
# No se permiten difusiones dirigidas de ICMP a nuestra red para evitar
# los ataques del estilo denominado 'Smurf'.
$IPFWADM -I -a deny -P icmp -W $ANYDEV -D $OURBCAST

# TCP
# Aceptaremos todos los datagramas de TCP que pertenezcan a una
```

```
# conexión ya existente (i.e. cuyo bit de ACK valga 1)
# en el caso de los puertos de TCP que estamos permitiendo.
# Esto debería capturar más del 95% de todos los paquetes válidos de TCP.
$IPFWADM -I -a accept -P tcp -D $OURNET $TCPIN -k -b

# TCP - CONEXIONES ENTRANTES
# Aceptaremos únicamente las solicitudes de conexión desde
# fuera sólamente en los puertos de TCP permitidos.
$IPFWADM -I -a accept -P tcp -W $ANYDEV -D $OURNET $TCPIN -y

# TCP - CONEXIONES SALIENTES
# Aceptaremos todas las conexiones salientes de TCP hacia los puertos
# de TCP permitidos.
$IPFWADM -I -a accept -P tcp -W $OURDEV -D $ANYADDR $TCPOUT -y

# UDP - ENTRADA
# Aceptaremos la entrada de datagramas UDP por puertos permitidos
$IPFWADM -I -a accept -P udp -W $ANYDEV -D $OURNET $UDPIN

# UDP - SALIDA
# Aceptaremos la salida de datagramas hacia los puertos permitidos.
$IPFWADM -I -a accept -P udp -W $OURDEV -D $ANYADDR $UDPOUT

# ICMP - ENTRADA
# Aceptaremos la entrada de los datagramas de ICMP de los tipos permitidos.
$IPFWADM -I -a accept -P icmp -W $ANYDEV -D $OURNET $ICMPIN

# ICMP - SALIDA
# Aceptaremos la salida de los datagramas de ICMP de los tipos permitidos.
$IPFWADM -I -a accept -P icmp -W $OURDEV -D $ANYADDR $ICMPOUT

# CASO POR DEFECTO y REGISTRO
# Todos los restantes datagramas caen dentro de la regla por defecto
# y son eliminados. Serán registrados si más arrriba se ha configurado
# la variable LOGGING.
#
if [ "$LOGGING" ]
then
    # Registra los paquetes de TCP descartados
    $IPFWADM -I -a reject -P tcp -o

    # Registra los paquetes de UDP descartados
    $IPFWADM -I -a reject -P udp -o

    # Registra los paquetes de ICMP descartados
    $IPFWADM -I -a reject -P icmp -o
fi
#
# fin.
```

Ahora se vuelve a implementar el ejemplo con la orden **ipchains**:

```
#!/bin/bash
##########################################################################
# VERSIÓN PARA IPCHAINS
# Este configuración está pensada como ejemplo de configuración de
# un cortafuegos sobre un 'host' único que no hospede él mismo ningún
# servicio
##########################################################################

# SECCIÓN CONFIGURABLE POR EL USUARIO

# El nombre y la localización de la utilidad ipchains.
IPCHAINS=ipchains

# Ruta del ejecutable de ipchains.
```

```
PATH="/sbin"

# El espacio de direcciones de nuestra red interna y el dispositivo
# de red que la soporta.
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Las direcciones de fuera y el dispositivo de red que la soporta.
ANYADDR="0/0"
ANYDEV="eth1"

# Los servicios de TCP que deseamos permitir que pasen - un "" vacío
# significa todos los puertos
# nota: separados por espacios
TCPIN="smtp www"
TCPOUT="smtp www ftp ftp-data irc"

# Los servicios de UDP que deseamos permitir que pasen - un "" vacío
# significa todos los puertos
# nota: separados por espacios
UDPIN="domain"
UDPOUT="domain"

# Los servicios de ICMP que deseamos permitir que pasen - un "" vacío
# significa todos los tipos
# referencia para los números de los tipos: /usr/include/netinet/ip_icmp.h
# nota: separados por espacios
ICMPIN="0 3 11"
ICMPOUT="8 3 11"

# Registro; descomente la siguiente línea para habilitar el registro
# de los datagramas rechazados por el cortafuegos
# LOGGING=1

# FIN DE LA SECCIÓN CONFIGURABLE POR EL USUARIO
###########################################################################
# Borra las reglas de la cadena de entrada
$IPCHAINS -F input

# Por defecto, queremos denegar el acceso a los intentos de entrada
$IPCHAINS -P input deny

# SUPLANTACIÓN DE IDENTIDAD
# No se debería aceptar ningún datagrama proveniente de fuera con una
# direccción de origen coincidente con una de las nuestras, por
# eso las rechazamos.
$IPCHAINS -A input -s $OURNET -i $ANYDEV -j deny

# 'SMURF'
# No se permiten difusiones dirigidas de ICMP a nuestra red para evitar
# los ataques del estilo denominado 'Smurf'.
$IPCHAINS -A input -p icmp -w $ANYDEV -d $OURBCAST -j deny

# Deberíamos aceptar fragmentos, esto se debe explicitar en ipchains.
$IPCHAINS -A input -f -j accept

# TCP
# Aceptaremos todos los datagramas de TCP que pertenezcan a una
# conexión ya existente (i.e. cuyo bit de ACK valga 1)
# en el caso de los puertos de TCP que estamos permitiendo.
# Esto debería capturar más del 95% de todos los paquetes válidos de TCP.
$IPCHAINS -A input -p tcp -d $OURNET $TCPIN ! -y -b -j accept

# TCP - CONEXIONES ENTRANTES
```

```
# Aceptaremos únicamente las solicitudes de conexión desde
# fuera en los puertos de TCP permitidos.
$IPCHAINS -A input -p tcp -i $ANYDEV -d $OURNET $TCPIN -y -j accept

# TCP - CONEXIONES SALIENTES
# Aceptaremos todas las conexiones salientes de TCP hacia los puertos
# de TCP permitidos.
$IPCHAINS -A input -p tcp -i $OURDEV -d $ANYADDR $TCPOUT -y -j accept

# UDP - ENTRADA
# Aceptaremos la entrada de los datagramas de UDP por puertos permitidos
$IPCHAINS -A input -p udp -i $ANYDEV -d $OURNET $UDPIN -j accept

# UDP - SALIDA
# Aceptaremos la salida de datagramas hacia los puertos permitidos.
$IPCHAINS -A input -p udp -i $OURDEV -d $ANYADDR $UDPOUT -j accept

# ICMP - ENTRADA
# Aceptaremos la entrada de los datagramas de ICMP de los tipos permitidos
$IPCHAINS -A input -p icmp -w $ANYDEV -d $OURNET $ICMPIN -j accept

# ICMP - SALIDA
# Aceptaremos la salida de los datagramas de ICMP de los tipos permitidos.
$IPCHAINS -A input -p icmp -i $OURDEV -d $ANYADDR $ICMPOUT -j accept

# CASO POR DEFECTO y REGISTRO
# Todos los restantes datagramas caen dentro de la regla por defecto
# y son eliminados. Serán registrados si más arriba se ha configurado
# la variable LOGGING.
#
if [ "$LOGGING" ]
then
    # Registra los paquetes de TCP descartados
    $IPCHAINS -A input -p tcp -l -j reject

    # Registra los paquetes de UDP descartados
    $IPCHAINS -A input -p udp -l -j reject

    # Registra los paquetes de ICMP descartados
    $IPCHAINS -A input -p icmp -l -j reject
fi
#
# fin.
```

En el ejemplo con **iptables**, se ha pasado autilizar el conjunto de reglas `FORWARD` por la diferencia de significado del conjunto de reglas `INPUT` en la implementación de *netfilter*. Esto tiene implicaciones; significa que ninguna de las reglas protege el 'host' mismo del cortafuegos. Para imitar con precisión el ejemplo con **ipchains**, se replicaría cada una de las reglas de la cadena `INPUT`. En aras de la claridad, en su lugar se ha decidido eliminar todos los datagramas entrantes provenientes desde el lado de fuera de la interfaz

```
#!/bin/bash
##########################################################################
# VERSIÓN PARA IPTABLES
# Este configuración está pensada como ejemplo de configuración de
# un cortafuegos sobre un 'host' único que no hospede él mismo ningún
# servicio
##########################################################################

# SECCIÓN CONFIGURABLE POR EL USUARIO

# El nombre y la localización de la utilidad iptables.
IPTABLES=iptables

# Ruta del ejecutable de iptables.
PATH="/sbin"
```

```
# El espacio de direcciones de nuestra red interna y el dispositivo
# de red que la soporta.
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Las direcciones de fuera y el dispositivo de red que la soporta.
ANYADDR="0/0"
ANYDEV="eth1"

# Los servicios de TCP que deseamos permitir que pasen - un "" vacío
# significa todos los puertos
# nota: separados por espacios
TCPIN="smtp,www"
TCPOUT="smtp,www,ftp,ftp-data,irc"

# Los servicios de UDP que deseamos permitir que pasen - un "" vacío
# significa todos los puertos
# nota: separados por espacios
UDPIN="domain"
UDPOUT="domain"

# Los servicios de ICMP que deseamos permitir que pasen - un "" vacío
# significa todos los tipos
# referencia para los números de los tipos: /usr/include/netinet/ip_icmp.h
# nota: separados por espacios
ICMPIN="0,3,11"
ICMPOUT="8,3,11"

# Registro; descomente la siguiente línea para habilitar el registro
# de los datagramas rechazados por el cortafuegos
# LOGGING=1

# FIN DE LA SECCIÓN CONFIGURABLE POR EL USUARIO
###########################################################################
# Borra las reglas de la cadena de entrada
$IPTABLES -F FORWARD

# # Por defecto, queremos denegar el acceso a los intentos de entrada
$IPTABLES -P FORWARD deny

# Rechaza todos los datagramas destinados a este host y recibidos
# desde fuera.
$IPTABLES -A INPUT -i $ANYDEV -j DROP

# SUPLANTACIÓN DE IDENTIDAD
# No se debería aceptar ningún datagrama proveniente de fuera con una
# direccción de origen coincidente con una de las nuestras, por
# eso las rechazamos.
$IPTABLES -A FORWARD -s $OURNET -i $ANYDEV -j DROP

# 'SMURF'
# No se permiten difusiones dirigidas de ICMP a nuestra red para evitar
# los ataques del estilo denominado 'Smurf'.
$IPTABLES -A FORWARD -m multiport -p icmp -i $ANYDEV -d $OURNET -j DENY

# Deberíamos aceptar fragmentos, esto se debe explicitar en iptables.
$IPTABLES -A FORWARD -f -j ACCEPT

# TCP
# Aceptaremos todos los datagramas de TCP que pertenezcan a una
# conexión ya existente (i.e. cuyo bit de ACK valga 1)
# en el caso de los puertos de TCP que estamos permitiendo.
# Esto debería capturar más del 95% de todos los paquetes válidos de TCP.
```

```
    $IPTABLES -A FORWARD -m multiport -p tcp -d $OURNET --dports $TCPIN /
        ! --tcp-flags SYN,ACK ACK -j ACCEPT
    $IPTABLES -A FORWARD -m multiport -p tcp -s $OURNET --sports $TCPIN /
        ! --tcp-flags SYN,ACK ACK -j ACCEPT

    # TCP - CONEXIONES ENTRANTES
    # Aceptaremos únicamente las solicitudes de conexión desde
    # fuera en los puertos de TCP permitidos.
    $IPTABLES -A FORWARD -m multiport -p tcp -i $ANYDEV -d $OURNET $TCPIN /
        --syn -j ACCEPT

    ## TCP - CONEXIONES SALIENTES
    # Aceptaremos todas las conexiones salientes de TCP hacia los puertos
    # de TCP permitidos
    $IPTABLES -A FORWARD -m multiport -p tcp -i $OURDEV -d $ANYADDR /
        --dports $TCPOUT --syn -j ACCEPT
    # UDP - ENTRADA
    ## Aceptaremos la entrada y vuelta de los datagramas de UDP por puertos
    #permitidos.
    $IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -d $OURNET /
        --dports $UDPIN -j ACCEPT
    $IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -s $OURNET /
        --sports $UDPIN -j ACCEPT
    # UDP - SALIDA
    # Se aceptarán la salida de los datagramas de UDP hacia los puertos
    permitidos y su vuelta.
    $IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -d $ANYADDR /
        --dports $UDPOUT -j ACCEPT
    $IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -s $ANYADDR /
        --sports $UDPOUT -j ACCEPT
    # ICMP - ENTRADA
    # Aceptaremos la entrada de los datagramas de ICMP de los tipos permitidos
    $IPTABLES -A FORWARD -m multiport -p icmp -i $ANYDEV -d $OURNET /
        --dports $ICMPIN -j ACCEPT
    # ICMP - SALIDA
    # Aceptaremos la salida de los datagramas de ICMP de los tipos permitidos.
    $IPTABLES -A FORWARD -m multiport -p icmp -i $OURDEV -d $ANYADDR /
        --dports $ICMPOUT -j ACCEPT
    # CASO POR DEFECTO y REGISTRO
    # Todos los restantes datagramas caen dentro de la regla por defecto
    # y son eliminados. Serán registrados si más arriba se ha configurado
    # la variable LOGGING.
    #
    if [ "$LOGGING" ]
    then
        # Registra los paquetes de TCP descartados
        $IPTABLES -A FORWARD -m tcp -p tcp -j LOG
        # Registra los paquetes de UDP descartados
        $IPTABLES -A FORWARD -m udp -p udp -j LOG
        # Registra los paquetes de ICMP descartados
        $IPTABLES -A FORWARD -m udp -p icmp -j LOG
    fi
    #
    # fin.
```

En muchas situaciones simples, para utilizar el ejemplo todo lo que necesitará será editar la sección superior del fichero denominado SECCIÓN CONFIGURABLE POR EL USUARIO para especificar qué protocolos y tipos de datagramas desea que se les permita su entrada y su salida. Para el caso de configuraciones más complejas, se necesitará también editar la sección inferior. Recuerde que el ejemplo es simple, por tanto examínelo cuidadosamente para asegurarse de que hace lo que usted desea cuando lo implemente.

Capítulo 10. Contabilidad IP

En el mundo de servicio comercial Internet de hoy, está volviéndose cada vez más importante saber cuántos datos está transmitiendo y recibiendo en sus conexiones de red. Si es un Provedor de Servicios Internet (ISP) y cobra a sus clientes por volumen, esto será esencial para su negocio. Si es un cliente de un Provedor de Servicios Internet que cobra por el volumen de datos, encontrará muy útil recoger sus propios datos para asegurar la exactitud de sus cargos de Internet.

Hay otros usos de la contabilidad de red que no tienen nada que ver con dinero y facturas. Si usted administra un servidor que ofrece varios tipos diferentes de servicios de red, podría ser útil para usted, saber exactamente cuántos datos está generándose por cada uno. Esta clase de información puede ayudarle en la toma de decisiones, como qué hardware comprar o cuántos servidores ejecutar.

El núcleo Linux proporciona una funcionalidad que le permite coleccionar todo tipo de información útil sobre el tráfico de red que ve. Esta funcionalidad es llamada *Contabilidad IP*.

10.1. Configurando el núcleo para contabilidad IP

La característica de contabilidad IP de Linux se relaciona muy estrechamente al software de cortafuego. Los lugares en que necesita coleccionar datos de contabilidad son los mismos lugares en los que estaría interesado realizando filtrado con cortafuegos: dentro y fuera de un puesto en la red, y en el software que hace asignación de ruta de datagramas. Si no ha leído la sección de cortafuegos, ahora es probablemente un buen momento para hacerlo, puesto que estaremos usando algunos de los conceptos descritos en Capítulo 9.

Para activar la característica de contabilidad IP, debe ver primero si su núcleo Linux está configurado para ello. Revise y vea si el fichero `/proc/net/ip_acct` existe. Si es así, su núcleo ya soporta contabilidad IP. Si no es así, debe compilar un núcleo nuevo, asegurándose que responde Y a las opciones en las series de núcleos 2.0 y 2.2:

```
Networking options  --->
    [*] Network firewalls
    [*] TCP/IP networking
     ...
    [*] IP: accounting
```

o en la serie de núcleos 2.4:

```
Networking options  --->
    [*] Network packet filtering (replaces ipchains)
```

10.2. Configurando Contabilidad IP

Debido a que la contabilidad IP se relaciona estrechamente con el cortafuegos de IP, la misma herramienta fue designada para configurarla, de modo que **ipfwadm**, **ipchains** o **iptables** sean utilizados para configurar la contabilidad IP. La sintaxis de órdenes es muy similar a la de las reglas del cortafuegos, así que no nos centraremos en eso, pero discutiremos qué puede descubrir sobre la naturaleza de su tráfico de red utilizando esta característica.

La sintaxis general para contabilidad IP con **ipfwadm** es:

```
# ipfwadm -A [sentido] [orden] [parámetros]
```

El argumento sentido es nuevo. Esto se codifica simplemente como `entrada (in)`, `salida (out)`, o `ambos (both)`. Estas trayectorías son desde la perspectiva de la propia máquina GNU/Linux. `entrada (in)` se refiere a datos que entran a la máquina desde una conexión de red y `salida (out)` se refiere a datos que están transmitiéndose por este nodo en una conexión de red. El sentido `ambos (both)` es la suma de ambas trayectorias, entrante y saliente.

La sintaxis general para la orden **ipchains** e **iptables** es:

```
# ipchains -A cadena especificación-de-regla
# iptables -A cadena especificación-de-regla
```

Las órdenes **ipchains** e **iptables** permiten especificar el sentido de una manera más consistente con las reglas de cortafuegos. El cortafuegos de cadenas IP[108] no le permite configurar una regla que agrege ambos sentidos, pero permite configurar reglas en la cadena `forward` que la antigua implementación no hacía. Veremos la diferencia que produce, en algunos ejemplos un poco mas adelante.

Las órdenes son bastante iguales a las reglas de cortafuegos, excepto que las políticas de reglas no se aplican aquí. Podemos agregar, insertar, eliminar y listar las reglas de contabilidad. En el caso de **ipchains** e **iptables**, todas las reglas válidas son reglas de contabilidad, y cualquier orden que no especifica la opción *−j* sólo realiza recuento.

Las reglas de especificación de parámetros para contabilidad IP son las mismas que aquellas usadas para cortafuegos IP. Éstas son las que nosotros usamos para definir precisamente qué tráfico de la red deseamos contabilizar y sumar.

10.2.1. Contabilidad por Dirección

Trabajemos con un ejemplo para ilustrar como usaríamos la contabilidad IP.

Imagine que tenemos un encaminador basado en Linux que sirve a dos departamentos en la Cervecería Virtual. El encaminador tiene dos dispositivos Ethernet, `eth0` y `eth1`, cada uno de los cuales sirve a un departamento; y un dispositivo PPP, `ppp0`, que nos conecta vía un enlace serie de alta velocidad al campus principal de la Universidad Groucho Marx.

También imaginemos que para propósitos de faturación queremos conocer el total de tráfico generado por cada uno de los departamentos a lo largo del enlace serie, y para propósitos de administración queremos conocer el total de tráfico generado entre los dos departamentos.

La siguiente tabla muestra las interfaces y direcciones que usaremos en nuestro ejemplo:

interfaz	dirección	máscara de red
eth0	172.16.3.0	255.255.255.0
eth1	172.16.4.0	255.255.255.0

Para responder a la pregunta, ¿ Cuántos datos genera cada departamento en el enlace PPP ?, podríamos usar una regla parecida a:

```
# ipfwadm -A both -a -W ppp0 -S 172.16.3.0/24 -b
# ipfwadm -A both -a -W ppp0 -S 172.16.4.0/24 -b
```

o:

```
# ipchains -A input -i ppp0 -d 172.16.3.0/24
# ipchains -A output -i ppp0 -s 172.16.3.0/24
# ipchains -A input -i ppp0 -d 172.16.4.0/24
# ipchains -A output -i ppp0 -s 172.16.4.0/24
```

y con **iptables**:

```
# iptables -A FORWARD -i ppp0 -d 172.16.3.0/24
# iptables -A FORWARD -o ppp0 -s 172.16.3.0/24
# iptables -A FORWARD -i ppp0 -d 172.16.4.0/24
# iptables -A FORWARD -o ppp0 -s 172.16.4.0/24
```

La primera mitad de cada una de estas reglas dice, Cuente todos los datos viajando en cualquier dirección por la interfaz llamada ppp0 con una dirección origen o destino (recuerde la función de la bandera *−b* en

ipfwadm e **iptables**) de `172.16.3.0/24.` La segunda mitad de cada conjunto de reglas es la misma, pero para la segunda red Ethernet en nuestro sitio.

Para responder a la segunda pregunta , ¿ Cuántos datos viajan entre los dos departamentos ?, necesitamos una regla como esta:

```
# ipfwadm -A both -a -S 172.16.3.0/24 -D 172.16.4.0/24 -b
```

o:

```
# ipchains -A forward -s 172.16.3.0/24 -d 172.16.4.0/24 -b
```

o:

```
# iptables -A FORWARD -s 172.16.3.0/24 -d 172.16.4.0/24
# iptables -A FORWARD -s 172.16.4.0/24 -d 172.16.3.0/24
```

Estas reglas contarán todos los datagramas con una dirección origen perteneciente a una de las redes de departamento y una dirección destino perteneciente a la otra.

10.2.2. Contabilidad por el Puerto de Servicio

Bien, supongamos que también queremos una mejor idea de qué tipo de tráfico exactamente está transportándose por nuestro enlace PPP. Por ejemplo, nosotros podríamos querer saber cuánto del enlace están consumiendo los servicios FTP, SMTP, y Web.

Un guión de reglas para permitirnos coleccionar esta información podría parecerse a:

```
#!/bin/sh
# Coleccionar estadísticas de volumen FTP, SMTP y WWW para los datos
# transportados en nuestro enlace PPP utilizando ipfwadm
#
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 ftp ftp-data
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 smtp
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 www
```

o:

```
#!/bin/sh
# Coleccionar estadísticas de volumen FTP, SMTP y WWW para los datos
# transportados en nuestro enlace PPP utilizando ipchains
#
ipchains -A input -i ppp0 -p tcp -s 0/0 ftp-data:ftp
ipchains -A output -i ppp0 -p tcp -d 0/0 ftp-data:ftp
ipchains -A input -i ppp0 -p tcp -s 0/0 smtp
ipchains -A output -i ppp0 -p tcp -d 0/0 smtp
ipchains -A input -i ppp0 -p tcp -s 0/0 www
ipchains -A output -i ppp0 -p tcp -d 0/0 www
```

o:

```
#!/bin/sh
# Coleccionar estadísticas de volumen FTP, SMTP y WWW para los datos
# transportados en nuestro enlace PPP utilizando iptables
#
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport ftp-data:ftp
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport ftp-data:ftp
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport smtp
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport smtp
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport www
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport www
```

Hay un par de rasgos interesantes a esta configuración. Primeramente, hemos especificado el protocolo. Cuando especificamos puertos en nuestras reglas, también debemos especificar un protocolo porque TCP y UDP proveen conjuntos separados de puertos. Ya que todos estos servicios están basados en TCP, lo hemos especificado como el protocolo. Segundo, tenemos especificado dos servicios `ftp` y `ftp-data` en un comando. **ipfwadm** permite establecer puertos simples, rango de puertos o una lista arbitraria de puertos. La orden **ipchains** permite cualesquiera, puertos simples o rango de puertos, que es lo que hemos usado aquí. La sintaxis `ftp-data:ftp` significa puertos ftp–data (20) hasta ftp (21), y es como nosotros codificamos rangos de puertos en ambos: **ipchains** e **iptables**. Cuando usted tiene una lista de puertos en una regla de contabilidad, eso significa que cualquier dato recibido para alguno de los puertos en la lista, provocará que el dato sea sumado a los totales de esa entrada. Recordando que el servicio FTP utiliza dos puertos, el de órdenes y el de transferencia de datos, los hemos añadido a la vez para sumar el tráfico de FTP. Finalmente, especificamos la dirección origen como `0/0`, que es la notación especial que coincide con todas las direcciones y es requerida por ambas órdenes **ipfwadm** e **ipchains** para especificar los puertos.

Podemos extendernos un poco en el segundo punto para darnos una vista diferente de los datos en nuestro enlace. Ahora imaginemos que nosotros clasificamos tráfico FTP, SMTP, y del Web como tráfico esencial, y todo el otro tráfico como no esencial. Si nosotros estuviéramos interesados en ver la proporción del tráfico esencial al tráfico no esencial, podríamos hacer algo como:

```
# ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 ftp ftp-data smtp www
# ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 1:19 22:24 26:79 81:32767
```

Si ya ha examinado su fichero `/etc/services`, observará que la segunda regla cubre todos los puertos excepto (`ftp`, `ftp-data`, `smtp`, y `www`).

¿Cómo hacemos esto con las órdenes **ipchains** o **iptables**, puesto que ellas permiten sólo un argumento en la especificación de puerto ?. Podemos aprovecharnos en contabilidad, de las cadenas definidas por usuario tan fácil como en las reglas del cortafuegos. Considere el siguiente acercamiento:

```
# ipchains -N a-essent
# ipchains -N a-noness
# ipchains -A a-essent -j ACCEPT
# ipchains -A a-noness -j ACCEPT
# ipchains -A forward -i ppp0 -p tcp -s 0/0 ftp-data:ftp -j a-essent
# ipchains -A forward -i ppp0 -p tcp -s 0/0 smtp -j a-essent
# ipchains -A forward -i ppp0 -p tcp -s 0/0 www -j a-essent
# ipchains -A forward -j a-noness
```

Aquí creamos dos cadenas definidas por usuario, una llamada `a-essent`, donde capturamos datos de contabilidad para servicios esenciales y otra llamada `a-noness`, donde capturamos datos de contabilidad para servicios no esenciales. Entonces agregamos a nuestra cadena *forward* las reglas que coinciden con nuestros servicios esenciales y saltan a la cadena `a-essent`, donde tenemos justamente una regla que acepta todos los datagramas y los cuenta. La última regla en nuestra cadena *forward* es una regla que salta a nuestra cadena `a-noness`, donde otra vez tenemos solamente una regla que acepta todos los datagramas y los cuenta. La regla que salta a la cadena `a-noness` no será alcanzada por ninguno de nuestros servicios esenciales, puesto que ellos se habrán aceptado en su propia cadena. Nuestras cuentas para servicios esenciales y no esenciales estarán por consiguiente disponibles en las reglas dentro de esas cadenas. Éste es simplemente un acercamiento que podría tomar; hay otros. Nuestra implementación **iptables** del mismo acercamiento se parecería a:

```
# iptables -N a-essent
# iptables -N a-noness
# iptables -A a-essent -j ACCEPT
# iptables -A a-noness -j ACCEPT
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport ftp-data:ftp -j a-essent
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport smtp -j a-essent
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport www -j a-essent
# iptables -A FORWARD -j a-noness
```

Esto parece bastante simple. Desafortunadamente, hay un pequeño pero inevitable problema al intentar efectuar contabilidad por el tipo de servicio. Recordará que discutimos el rol que desempeña la MTU en redes TCP/IP en un capítulo anterior. La MTU define el datagrama más largo que se transmitirá en un dispositivo de red. Cuando un datagrama se recibe por un encaminador que es más grande que el MTU de la interfaz que necesita al retransmitirlo, el encaminador realiza un truco llamado *fragmentación*. El encaminador fragmenta el datagrama largo en piezas pequeñas no más largos que la MTU de la interfaz y entonces transmite éstas piezas. El encaminador construye nuevas cabeceras para poner delante de cada una de estas piezas, y éstas son las que la máquina remota usa para reconstruir el dato original. Desafortunadamente, durante el proceso de fragmentación el puerto se pierde para todos menos para el primer fragmento. Esto significa que la contabilidad IP no puede contar adecuadamente datagramas fragmentados. Puede contar fiablemente sólo el primer fragmento o datagramas no fragmentados. Hay un pequeño truco permitido por **ipfwadm** que asegura que mientras nosotros no podamos saber exactamente desde qué puerto el segundo y siguientes fragmentos vienen, podemos todavía contarlos. Una temprana versión del software de contabilidad Linux asignó a los fragmentos un número de puerto falso, 0xFFFF, que podríamos contar. Para asegurarnos que capturamos el segundo y posteriores fragmentos, podemos usar una regla como ésta:

```
# ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 0xFFFF
```

La implementación de cadenas IP tiene una solución ligeramente más sofisticada, pero el resultado es muy similar. Usando la orden **ipchains** usaríamos en cambio:

```
# ipchains -A forward -i ppp0 -p tcp -f
```

y con **iptables** usaríamos:

```
# iptables -A FORWARD -i ppp0 -m tcp -p tcp -f
```

Éstos no nos dirán el puerto original para estos datos, pero por lo menos podemos ver cuanto de nuestros datos son fragmentos, y seremos capaces de contabilizar el volumen de tráfico que ellos consumen.

En núcleos 2.2 podemos seleccionar una opción del núcleo en tiempo de compilación, que niega este problema completo si su máquina GNU/Linux está actuando como el único punto de acceso para una red. Si habilita la opción `IP: Desfragmentar siempre` cuando compila su núcleo, todos los datagramas recibidos serán reensamblados por el encaminador Linux antes de encaminar y retransmitir. Esta operación es realizada antes que el software de contabilidad y cortafuegos miren el datagrama, y así no tendrá trato con ningún fragmento. En núcleos 2.4 usted puede compilar y cargar el módulo *netfilter* `forward-fragment`.

10.2.3. Contabilidad de Datagramas ICMP

El protocolo ICMP no usa número de puerto de servicio y es por eso un poco más dificultoso coleccionar detalles. ICMP usa un número de tipos diferentes de datagramas. Muchos de éstos son inofensivos y normales, mientras otros sólo deben observarse bajo circunstancias especiales. A veces las personas con mucho tiempo en sus manos intentan maliciosamente deteriorar el acceso de un usuario a la red, generando grandes cantidades de mensajes ICMP. Esto es comúnmente denominado *saturamiento ping*[109]. Aun cuando la contabilidad IP no puede hacer nada para prevenir este problema (¡ Aunque el cortafuegos IP puede ayudar !) podemos al menos colocar reglas de contabilidad en un lugar que nos muestre si alguien lo ha estado intentando.

ICMP no usa los puertos como lo hacen TCP y UDP. En cambio ICMP tiene mensajes tipo ICMP. Podemos construir reglas de contabilidad para cada tipo de mensaje ICMP. Para hacer esto, colocamos el mensaje ICMP y el número del tipo en lugar del puerto en la orden de contabilidad **ipfwadm**. Listamos los tipos de mensaje ICMP en Sección 9.6.3.5 refíerase a él si usted necesita recordar cuáles son.

Una regla de contabilidad IP para coleccionar información sobre el volumen de datos ping que está enviándose a usted o que usted está generando podría verse como:

```
# ipfwadm -A both -a -P icmp -S 0/0 8
# ipfwadm -A both -a -P icmp -S 0/0 0
# ipfwadm -A both -a -P icmp -S 0/0 0xff
```

o, con **ipchains**:

```
# ipchains -A forward -p icmp -s 0/0 8
# ipchains -A forward -p icmp -s 0/0 0
# ipchains -A forward -p icmp -s 0/0 -f
```

o, con **iptables**:

```
# iptables -A FORWARD -m icmp -p icmp --sports echo-request
# iptables -A FORWARD -m icmp -p icmp --sports echo-reply
# iptables -A FORWARD -m icmp -p icmp -f
```

La primera regla colecciona información sobre datagramas Petición de eco ICMP (petición ping) [110], y la segunda regla colecciona información sobre datagramas Respuesta de eco ICMP (respuesta ping). La tercera regla colecciona información sobre fragmentos de datagrama ICMP. Este es un truco similar al descrito para fragmentos de datagramas TCP y UDP.

Si usted especifica la dirección origen y/o destino en sus reglas, puede seguir la pista de dónde están viniendo los ping, tales como si ellos se originan dentro o fuera de su red. Una vez que ha determinado de dónde están viniendo los datagramas pillos, usted puede decidir si quiere poner reglas de cortafuegos en un sitio para evitarlos o tomar alguna otra acción, como avisar al dueño de la red agraviante para avisarles del problema, o quizás incluso, acción legal si el problema es un acto malévolo.

10.2.4. Contabilidad por Protocolo

Imaginemos ahora que estamos interesados en conocer cuánto tráfico en nuestro enlaces es TCP, UDP, e ICMP. Usaríamos reglas como las siguientes:

```
# ipfwadm -A both -a -W ppp0 -P tcp -D 0/0
# ipfwadm -A both -a -W ppp0 -P udp -D 0/0
# ipfwadm -A both -a -W ppp0 -P icmp -D 0/0
```

o:

```
# ipchains -A forward -i ppp0 -p tcp -d 0/0
# ipchains -A forward -i ppp0 -p udp -d 0/0
# ipchains -A forward -i ppp0 -p icmp -d 0/0
```

o:

```
# iptables -A FORWARD -i ppp0 -m tcp -p tcp
# iptables -A FORWARD -o ppp0 -m tcp -p tcp
# iptables -A FORWARD -i ppp0 -m udp -p udp
# iptables -A FORWARD -o ppp0 -m udp -p udp
# iptables -A FORWARD -i ppp0 -m icmp -p icmp
# iptables -A FORWARD -o ppp0 -m icmp -p icmp
```

Con estas reglas situadas, todo el tráfico fluyendo por la interfaz `ppp0` será analizado para determinar si es TCP, UDP, o tráfico de IMCP y los contadores apropiados serán actualizados para cada uno. El ejemplo con **iptables** divide el flujo entrante del flujo saliente como lo exige su sintaxis.

10.3. Utilizando los resultados de contabilidad IP

Está muy bien estar recolectando esta información, pero ¿ cómo hacemos realmente para conseguir verlos ? Para ver los datos de contabilidad coleccionados y las reglas de contabilidad configuradas, usamos nuestras órdenes de configuración de cortafuegos, pidiéndole listar nuestras reglas. Los contadores de byte y paquetes para cada una de nuestras reglas son listadas en la salida.

Las órdenes **ipfwadm**, **ipchains**, e **iptables** difieren en cómo se manejan los datos de contabilidad, asi que trataremos estos independientemente.

10.3.1. Listando datos de contabilidad con ipfwadm

Los medios más básicos de listar nuestros datos de contabilidad con la orden **ipfwadm** son utilizados así:

```
# ipfwadm -A -l
IP accounting rules
 pkts bytes dir prot source              destination         ports
 9833 2345K i/o all  172.16.3.0/24      anywhere            n/a
56527   33M i/o all  172.16.4.0/24      anywhere            n/a
```

Esto nos dirá el número de paquetes enviados en cada dirección. Si usamos el formato de salida extendida, con la opción `-e` (no mostrada aquí porque la salida es muy ancha para una página), nosotros también proporcionamos las opciones y los nombres de la interfaz aplicables. Muchos de estos campos en la salida serán autoexplicativos, pero puede que los siguientes no:

dir
: El sentido en que la regla aplica. Aquí se esperan los valores `in`, `out`, o `i/o`, significando ambos sentidos.

prot
: Los protocolos a los cuales la regla se aplica.

opt
: Una forma codificada de las opciones que usamos al invocar **ipfwadm**.

ifname
: El nombre de la interfaz a que la regla se aplica.

ifaddress
: La dirección de la interfaze a que la regla se aplica.

De modo predeterminado, **ipfwadm** despliega el contador de bytes y paquetes en una forma reducida, redondeado al mil más cercano (K) o millón (M). Podemos pedirle que despliegue los datos recolectados en unidades exactas usando la opción extendida como sigue:

```
# ipfwadm -A -l -e -x
```

10.3.2. Listando datos de contabilidad con ipchains

La orden **ipchains** no desplegará nuestros datos de contabilidad (contador de byte y paquetes) a menos que le proporcionemos el argumento `-v`. Los recursos simples para listar nuestros datos de contabilidad con **ipchains** se utilizan así:

```
# ipchains -L -v
```

Nuevamente, así como con **ipfwadm**, podemos desplegar el contador de bytes y paquetes en unidades usando el modo de salida extendida. La orden **ipchains** usa el argumento `-x` para esto.

```
# ipchains -L -v -x
```

10.3.3. Listando datos de contabilidad con iptables

El comando **iptables** se comporta muy similarmente a la orden **ipchains**. Otra vez, debemos usar `-v` cuando listemos nuestras reglas para ver los contadores de contabilidad. Para listar nuestros datos de contabilidad, podemos utilizar:

```
# iptables -L -v
```

Tal como para la orden **ipchains**, podemos usar el argumento `-x` para mostrar la salida en formato extendido con cifras unitarias.

10.4. Restableciendo contadores

Los contadores de contabilidad IP pueden desbordarse si usted los deja mucho tiempo. Si se desbordan, tendrá dificultad determinando el valor que realmente representan. Para evitar este problema, debe leer los datos de contabilidad periódicamente, registrarlos, y entonces restablecer los contadores a cero para comenzar a recolectar información de contabilidad para el próximo período de recuento.

La órdenes **ipfwadm**, **ipchains** e **iptables** le proporcionan recursos para hacer esto bastante simple:

```
# ipfwadm -A -z
```

o:

```
# ipchains -Z
```

o:

```
# iptables -Z
```

Usted puede incluso combinar el listado junto a la acción de puesta a cero para asegurarse que ningún dato de contabilidad es perdido en medio:

```
# ipfwadm -A -l -z
```

o:

```
# ipchains -L -Z
```

o:

```
# iptables -L -Z -v
```

Estas órdenes primero listarán los datos de contabilidad y entonces inmediatamente pondrá a cero los contadores y empezará la cuenta de nuevo. Si está interesado en coleccionar y utilizar esta información regularmente, usted probablemente querrá poner esta orden en una secuencia de órdenes que registre la salida y lo guarde en alguna parte, y ejecutar la secuencia de órdenes periódicamente utilizando la utilidad **cron**

10.5. Vaciando las reglas

Una último orden que puede ser útil, le permite vaciar todas las reglas de contabilidad que haya configurado. Esto es bastante útil cuando quiere alterar radicalmente su conjunto de reglas sin reiniciar el sistema.

El argumento `-f` en combinación con la orden **ipfwadm** vaciará todas las reglas del tipo que usted especifique. **ipchains** apoya el argumento `-F`, que hace lo mismo:

```
# ipfwadm -A -f
```

o:

```
# ipchains -F
```

o:

```
# iptables -F
```

Esto vacía todas sus reglas de contabilidad IP configuradas, quitándolas todas y salvándolo de tener que quitar cada una de ellas individualmente. Observe que vaciar las reglas con **ipchains** no provoca que cualquier cadena definida por el usuario sea quitada, sólo las reglas dentro de ellas.

10.6. Colección pasiva de datos de contabilidad

Un último truco que podría gustarle considerar: si su máquina Linux está conectada a una Ethernet, puede aplicar reglas a todos los datos del segmento, no sólo aquello que es transmitido por él o destinado para él. Su máquina escuchará pasivamente todos los datos en el segmento y los contará.

Usted debe primero desactivar el reenvío IP en su máquina Linux para que no intente encaminar los datagramas que recibe. [111] En los núcleos 2.0.36 y 2.2, ésto es una cuestión de:

```
# echo 0 >/proc/sys/net/ipv4/ip_forward
```

Debe habilitar entonces el modo promiscuo en su interfaz Ethernet utilizando la orden **ifconfig**. Ahora puede colocar reglas de contabilidad que le permitan recoger información sobre los datagramas que fluyen a lo largo de su Ethernet sin involucrar en absoluto a su GNU/Linux en la ruta.

Capítulo 11. Enmascaramiento IP yTraducción de Direcciones de Red

No hace falta tener una memoria excelente para recordar los días en los que sólo las grandes compañías se podían permitir disponer de un cierto número de máquinas conectadas por una red local. Frente a aquello, hoy los precios de la tecnología de red han bajado y bajado hasta producir dos consecuencias: La primera, que las redes locales sean algo común, presentes incluso en entornos domésticos. Es seguro que muchos de los lectores, usuarios de GNU/Linux, tendrán en su casa dos o más computadoras conectadas por algún tipo de ethernet. La segunda, que los recursos de red, y de forma especial las direcciones IP, hayan llegado a ser algo escasos y, aunque no están lejanos los tiempos en que eran gratuitos, sean ahora objeto de compraventa.

La mayor parte de la gente que disponga de una LAN deseará también disfrutar de una conexión a Internet que todas las máquinas de su red puedan utilizar al mismo tiempo. Las reglas del encaminamiento IP son muy estrictas respecto a la forma de manejar esta situación. Las soluciones tradicionales a este problema hubieran pasado por solicitar un conjunto de direcciones IP, probablemente un rango de clase C, dar a cada máquina de la LAN una dirección del rango asignado, y utilizar un entutador para conectar la LAN a Internet.

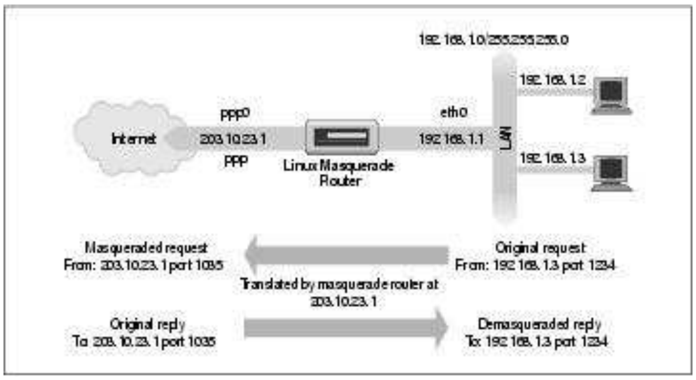
En el actual escenario de una Internet mercantilizada, esa solución saldría bastante cara. En primer lugar habría que pagar por el rango de direcciones asignado, en segundo lugar habría que pagar con toda probabilidad al Proveedor de Servicios de Internet (ISP) por el privilegio de disponer de una ruta hacia la red local en sus máquinas, de tal forma que el resto de Internet supiera cómo llegar a ellas. Esto puede sonar posible para algunas empresas, pero en una instalación doméstica los costes no estarían justificados.

Afortunadamente GNU/Linux proporciona una solución al problema, solución que utiliza un componente de un grupo de funcionalidades avanzadas de red llamadas en conjunto Traducción de Direcciones de Red (NAT). NAT es un conjunto de procedimientos para modificar las direcciones IP contenidas en las cabeceras de los datagramas IP mientras éstos viajan (al vuelo). Puede sonar extraño, pero mostraremos que se trata de la solución ideal al problema real para muchos que acabamos de plantear. 'IP masquerade' es el nombre que recibe un tipo de traducción de direcciones de red que permite que todas las máquinas de una red privada utilicen Internet contando con una única conexión (y una única dirección IP).

El enmascaramiento IP (en inglés «IP masquerading») permite utilizar un rango de direcciones privadas (reservadas) en la red local y que el encaminador GNU/Linux se encargue de hacer al vuelo ciertas traducciones de direcciones IP y puertos. Cuando le llega un datagrama IP de alguna máquina de la red local, se fija en el protocolo de nivel superior encapsulado en el mismo («UDP», «TCP», «ICMP», etc...) y modifica el datagrama para que parezca que fue generado por el propio encaminador (y recuerda qué ha sido modificado). A continuación saca el datagrama a Internet donde aparece generado por la única dirección IP pública del encaminador. Cuando la máquina destino recibe el datagrama cree que se ha originado en la máquina GNU/Linux, y responde a su dirección de Internet. Cuando el encaminador GNU/Linux recibe un datagrama en su interfaz de red conectada a Internet, busca en su tabla de conexiones enmascaradas en curso para ver si el datagrama pertenece a alguna máquina de la LAN y, si es así, deshace la traducción que hizo en el primer datagrama y reenvía este datagrama de respuesta a la máquina local.

En Figura 11−1 aparece un ejemplo sencillo.

Figura 11−1. Un escenario de enmascaramiento IP típico



Tenemos una pequeña red ethernet en la que utilizamos uno de los rangos de direcciones reservadas. La red dispone de un encaminador con enmascaramiento, una máquina GNU/Linux, por supuesto, que proporciona acceso a Internet. Una de las máquinas de la red (192.168.1.3) desea establecer una conexión con el host remoto 209.1.106.178 en el puerto 8888. El equipo encamina su datagrama por el encaminador con enmascaramiento, que identifica la petición de conexión como requiriente de los servicios de enmascaramiento. El encaminador entonces acepta el datagrama y reserva un número de puerto (1035) para este menester, sustituye la dirección IP y número de puerto de la máquina origen del datagrama por los suyos propios, y transmite el datagrama al host destino. El host destino cree que ha recibido una petición de conexión de la máquina GNU/Linux enmascaradora, y genera un datagrama de respuesta. La máquina enmascaradora, al recibir ese datagrama, halla la asociación en su tabla de enmascaramiento y deshace la sustitución que llevó a cabo en el primer datagrama. Entonces transmite el datagrama de respuesta a la máquina origen.

La máquina local cree que se está comunicando directamente con el host remoto. El host remoto no sabe nada de la existencia de la máquina local y cree que ha establecido una conexión con la máquina GNU/Linux enmascaradora. La máquina GNU/Linux enmascaradora sabe que las otras dos máquinas están hablando entre sí y en qué puertos, y realiza las traducciones de direcciones y puertos necesarias para que la comunicación tenga lugar.

Todo lo anterior puede parecer un poco confuso, y puede que lo sea, pero funciona y es verdadermente simple de poner a punto. Así que no se preocupe si aún no comprende todos los detalles.

11.1. Efectos Laterales y Beneficios Accesorios

La funcionalidad de enmascaramiento IP viene acompañada de su propio conjunto de efectos laterales, algunos son útiles y algunos pueden acabar siendo un problema.

Ninguna de las máquinas en la red detrás del encaminador enmascarador son jamás vistas directamente desde Internet. Consecuentemente, sólamente se necesita una dirección IP válida y rutable para permitir que todas las máquinas establezcan conexiones hacia Internet. Esto tiene un lado no tan bueno: ninguna de esas máquinas es visible desde Internet, y por lo tanto no se puede conectar directamente a ellas desde Internet. La única máquina visible en una red enmascarada es el propio encaminador enmascarador. Se trata de algo importante cuando se piensa en servicios como el correo o el FTP. Resulta de utilidad decidir qué servicios deberían ser provistos por la máquina enmascaradora y para cuáles debería actuar como proxy o tratar de algún otro modo especial.

Segundo, dado que ninguna de las máquinas enmascaradas son visibles, se encuentran relativamente protegidas de ataques del exterior. Eso puede simplificar (o eliminar) la necesidad de puesta a punto de funcionalidades de cortafuegos en la máquina enmascaradora. No se debe confiar demasiado en ésto, puesto que la red local estará únicamente tan segura como lo esté la máquina enmascaradora. Así, si la seguridad es un punto importante, se debería utilizar un cortafuegos para protegerla.

Tercero, el enmascaramiento IP tendrá cierto impacto negativo en el rendimiento de su red. En un escenario típico ese impacto negativo será probablemente insignificante. Si se tiene un gran número de sesiones enmascaradas activas puede ocurrir que se perciba cierta sobrecarga en la máquina enmascaradora que afecte negativamente al rendimiento de la red. El enmascaramiento IP implica un incremente considerable en el proceso que requiere cada datagrama comparado con el normalmente exigiría. Si piensa utilizar un 386SX16 como encaminador enmascarador para una conexión telefónica a Internet puede resultar, pero no espere demasiado si quiere usarlo como encaminador en su red corporativa a velocidades Ethernet.

Por último, ciertos servicios de red simplemente no funcionarán a través de enmascaramiento, o, al menos, no sin un poco de ayuda. Típicamente se trata de servicios que dependen de conexiones entrantes para funcionar, como ciertos tipos de Canales de Comunicación Directa (DCC), ciertas funciones del IRC, o algunos tipos de servicios de audio y vídeo multicast. Algunos de esos servicios disponen de módulos del kernel especialmente desarrollados para proporcionar una solución, y de ellos hablaremos dentro de un momento. Para otros es posible que no se encuentre soporte, así que se debe tener en cuenta que el enmascaramiento no es la solución adecuada en todas las situaciones.

11.2. Configuración del Núcleopara enmascaramiento IP

Para usar la función del enmascaramiento IP el núcleo debe ser compilado precisamente con soporte de enmascarmiento. Se deben seleccionar las siguientes opciones al configurar un núcleo de la serie 2.2:

```
    Networking options  --->
        [*] Network firewalls
        [*] TCP/IP networking
        [*] IP: firewalling
        [*] IP: enmascaramiento
        --- Protocol-specific masquerading support will be built as modules.
        [*] IP: ipautofw masq support
        [*] IP: ICMP masquerading
```

Nótese que partes del soporte de enmascaramiento están disponibles únicamente como módulos. Esto significa que habrá que ejecutar `make modules` además del habitual `make bzImage` cuando se compile el núcleo.

Los núcleos de la serie 2.4 no presentan el soporte de enmascaramiento Ip como una opción de la compilación. En su lugar, se debe seleccionar la opción del filtrado de paquetes de red:

```
    Networking options  --->
        [M] Network packet filtering (replaces ipchains)
```

En los núcleos de la serie 2.2 cierto número de módulos de asistencia para ciertos protocolos se crean durante la compilación del núcleo. Algunos protocolos comienzan estableciendo una conexión hacia fuera y entonces reciben una conexión desde fuera en otro puerto. Normalmente eso no podría ser enmascarado, puesto que no hay forma para la máquina GNU/Linux de asociar la segunda conexión con la primera sin meterse dentro del propio protocolo. Los módulos asistentes hacen justamente eso: examinan los datagramas y permiten que el enmascaramiento funcione para los protocolos que soportan. Protocolos que de otra forma serían imposibles de enmascarar. Los protocolos soportados son:

Módulo	Protocolo
ip_masq_ftp	FTP
ip_masq_irc	IRC
ip_masq_raudio	RealAudio
ip_masq_cuseeme	CU−See−Me
ip_masq_vdolive	Para VDO Live
ip_masq_quake	Quake, de IdSoftware

Esos módulos deben ser cargados manualmente mediante la orden **insmod**. Nótese que no pueden ser cargados por el demonio **kerneld**. Cada uno de esos módulos acepta como argumento el puerto en el que debe escuchar. Para el módulo RealAudio se podría poner:[112]

```
# insmod ip_masq_raudio.o ports=7070,7071,7072
```

Los puertos especificados dependen del protocolo. El mini−HOWTO del enmascaramiento IP (IP masquerade mini−HOWTO), escrito por Ambrose Au, trata con más detalle los módulos asistentes y cómo configurarlos.[113]

El paquete *netfilter* contiene módulos que realizan funciones similares. Por ejemplo, para que se hagan cargo del seguimiento de las sesiones FTP, se deben cargar los módulos `ip_conntrack_ftp` y `ip_nat_ftp.o`.

11.3. Configuración del enmascaramiento IP

Después de leer los capítulos sobre cortafuegos y auditoría IP, probablemente no sea sorprendente que los programas **ipfwadm**, **ipchains**, e **iptables** se utilicen para configurar también las reglas de enmascaramiento IP.

Las reglas de enmascaramiento son una clase especial de reglas de filtrado. Sólamente se puede enmascarar datagramas que se reciban por una interfaz y que vayan a ser encaminados por otra. Una regla de enmascaramiento se construye de forma parecida a una regla de encaminamiento de cortafuegos, pero incluyendo unas opciones especiales que le dicen al núcleo que debe enmascarar el datagrama IP. El programa **ipfwadm** utiliza la opción *−m*, **ipchains** utiliza `−j MASQ`, e **iptables**, `−j MASQUERADE` para indicar que los datagramas que cumplan las condiciones especificadas por la regla deben ser enmascarados.

Veamos un ejemplo. Una estudiante de informática de la Universidad Groucho Marx tiene unas cuantas máquinas en casa interconectadas en una pequeña red local ethernet. Ha decidido utilizar uno de los rangos privados de direcciones IP para su red. Comparte el lugar con otras estudiantes, todas las cuales tienen interés en usar Internet. Dado que las residentes no gozan de una economía especialmente boyante, no se plantean costearse una conexión permanente a Internet. En su lugar utilizan una conexión PPP telefónica. A todas ellas las gustaría poder compartir la conexión para chatear en el IRC, navegar por la Web y bajarse ficheros por FTP directamente a cada una de sus computadoras. El enmascaramiento IP es la respuesta.

Nuestra estudiante primeramente configura una máquina GNU/Linux para que se encargue del enlace telefónico y para que actúe como encaminador de la red local. La dirección IP que le asignan cuando se conecta telefónicamente no es importante. Configura el encaminador GNU/Linux para que haga enmascaramiento IP y usa uno de los rangos privados de direcciones IP para la red local: `192.168.1.0`. Se asegura de que todas las computadoras de la red tengan como ruta por defecto una que apunte al encaminador GNU/Linux.

Todo lo que se necesita para que el enmascaramiento funcione en ese escenario son las siguientes invocaciones del programa **ipfwadm**:

```
# ipfwadm −F −p deny
# ipfwadm −F −a accept −m −S 192.168.1.0/24 −D 0/0
```

o, utilizando **ipchains**:

```
# ipchains −P forward −j deny
# ipchains −A forward −s 192.168.1.0/24 −d 0/0 −j MASQ
```

o con **iptables**:

```
# iptables −t nat −P POSTROUTING DROP
# iptables −t nat −A POSTROUTING −o ppp0 −j MASQUERADE
```

Ahora simpre que alguna de las máquinas de la red local trate de acceder a un servicio en una máquina remota, sus datagramas serán automáticamente enmascarados por el encaminador enmascarador GNU/Linux. La primera regla en cada uno de los ejemplos impide que la máquina GNU/Linux encamine cualquier otro datagrama y proporciona cierta seguridad.

Para listar las reglas de enmascaramiento que se hayan creado, se utiliza la opción `-l` para el programa **ipfwadm**, exactamente como se dijo ya al referirnos a los cortafuegos.

Para listar la regla que creamos previamente se utiliza:

```
# ipfwadm -F -l -e
```

tras lo que debe aparecer algo como:

```
# ipfwadm -F -l -e
IP firewall forward rules, default policy: accept
 pkts bytes type  prot opt  tosa tosx ifname  ifaddress
    0     0 acc/m all  ---- 0xFF 0x00 any     any
```

La `/m` indica que se trata de una regla de enmascaramiento.

Para listar las reglas de enmascaramiento con **ipchains**, se utiliza la opción **-L**. Si listamos las reglas creadas previamente con **ipchains**, la salida sería algo como:

```
# ipchains -L
    Chain input (policy ACCEPT):
    Chain forward (policy ACCEPT):
    target     prot opt     source                destination           ports
    MASQ       all  ------  192.168.1.0/24        anywhere              n/a

    Chain output (policy ACCEPT):
```

Cualquier regla en la que como *objetivo* aparezca `MASQ` es una regla de enmascaramiento.

Finalmente, para listar las reglas mediante **iptables** hay que usar:

```
# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy DROP)
target     prot opt source               destination
MASQUERADE  all  --  anywhere             anywhere           MASQUERADE

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Las reglas de enmascaramiento aparecen aquí también con `MASQUERADE` como *objetivo*.

11.3.1. Configuración de parámetros temporales

Cuando se establece una nueva conexión, el software de enmascaramiento IP crea una asociación en memoria entre cada una de las máquinas implicadas. Tales asociaciones se pueden ver en cualquier momento inspeccionando el fichero `/proc/net/ip_masquerade`. Las asociaciones caducan tras un período de inactividad.

Los valores de máximo tiempo de inactividad se pueden configurar mediante **ipfwadm**. La sintaxis es:

```
ipfwadm -M -s <tcp> <tcpfin> <udp>
```

con **ipchains** sería:

```
ipchains -M -S <tcp> <tcpfin> <udp>
```

La implementación de *iptables* incluye unos valores por defecto mucho mayores que no permite cambiar.

Cada uno de esos valores representa un contador usado por el software de enmascaramiento IP y están dados en segundos. La siguiente tabla muestra un resumen de los contadores y sus significados:

Nombre	Descripción
tcp	Tiempo de caducidad de la sesión TCP. Cuánto puede estar inactiva una conexión TCP antes de que la correspondiente asociación sea eliminada.
tcpfin	Tiempo de caducidad de TCP tras un FIN. Cuanto permanece una asociación tras la desconexión de la correspondiente conexión TCP.
udp	Caducidad de sesión UDP. Tiempo máximo de inactividad de una «conexión» UDP antes de que la asociación correspondiente sea eliminada.

11.4. Manipulación del Servicio de Nombres

El manejo de las resoluciones de nombres desde las máquinas de una red local enmascarada ha representado tradicionalmente un problema. Hay dos maneras de encajar el DNS en un entorno con enmascaramiento. Se puede instruir a cada máquina para que utilice el mismo servidor DNS que utilice el encaminador GNU/Linux y dejar que el enmascaramiento IP haga su trabajo. Otra alternativa es ejecutar un servidor de nombres de caché en la máquina GNU/Linux y hacer que cada una de las máquinas en la LAN tenga configurada a la máquina GNU/Linux como su servidor DNS. Aunque se trata de una configuración más agresiva, es seguramente mejor porque reduce el volumen de tráfico DNS hacia el enlace con Internet y es ligeramente más rápido para la mayor parte de las peticiones, dado que serán respondidas utilizando la caché. La desventaja de esta configuración es que es más compleja. Sección 6.3.4, En el capítulo 6 se explica cómo configurar un servidor de nombres de sólo−caché.

11.5. Más sobre la traducción de direcciones de red

El software *netfilter* es capaz de realizar muchos tipos de traducción de direcciones de red. El enmascaramiento IP es una ejemplo simple de ello.

Es posible, por ejemplo, configurar reglas NAT que traduzcan sólo ciertas direcciones o rangos de direcciones y dejen las demás intactas, o que traduzcan rangos de direcciones en rangos en vez de en direcciones individuales (que es lo que hace el enmascaramiento). Se puede de hecho utilizar el programa **iptables** para crear reglas NAT que conviertan casi cualquier cosa, con combinaciones de condiciones que utilicen atributos estándares tales como dirección origen o destino, tipo de protocolo, puerto, etc.

La traducción de la dirección origen de un datagrama es conocida como «Source NAT» o `SNAT`, en la documentación de *netfilter*. La traducción de la dirección destino de un datagrama es conocida como «Destination NAT» o `DNAT`. La traducción del puerto TCP o UDP es conocida como `REDIRECCION`. `SNAT`, `DNAT`, y `REDIRECT` son *objetivos* que se pueden usar con **iptables** para construir reglas más complejas y sofisticadas.

El tema del NAT y su utilización necesitaría al menos un capítulo completo. ... ¡y es problable que incluso un libro! Lamentablemente no dispongo de espacio en este libro para tratarlo en mayor profundidad. Si se desea más información sobre los posibles usos del NAT, es conveniente leer el IPTABLES−HOWTO (o su versión en castellano: IPTABLES−COMO).

Capítulo 12. Características Importantesde Redes

Después de establecer con éxito IP y el traductor de direcciones, debe fijarse en los servicios que quiere suministrar a través de la red. Este capítulo cubre la configuración de unas sencillas aplicaciones de red, incluyendo el servidor**inetd** y los programas de la familia **rlogin**. También trataremos brevemente el interfaz de Llamada a Procedimiento Remoto (Remote Procedure Call), sobre el cuál servicios como Network File System (NFS) y Network Information System (NIS) se basan. Las configuraciones de NFS y NIS, sin embargo, som más complejas y están descritas en otros capítulos, así como lo están tanto el correo electrónico y las noticias de red.

Por supuesto, no queremos cubrir todas las aplicaciones de red en este libro. Si quiere instalar alguna que no esté cubierta aquí, como **talk**, **gopher**, o **http**, por favor consulte las páginas del manual del servidor para más detalles.

12.1. El Super Servidor inetd

Los programas que proporcionan servicios de aplicación a través de la red se llaman *demonios*[114]. Un demonio es un programa que abre un puerto, comúnmente un puerto de algún servicio bien conocido, y espera conexiones entrantes en él. Si ocurre una, el demonio crea un proceso hijo que acepta la conexión, mientras que el proceso padre continúa escuchando más peticiones. Este mecanismo funciona bien, pero tiene unas pocas desventajas; al menos una instancia de cada posible servicio que se quiera proporcionar, debe estar activa en memoria a todas horas. Además, la rutina software que hacen la escucha y la gestión del puerto tiene que ser replicada en cada uno de los demonios de red.

Para superar estas ineficiencias, muchas instalaciones Unix ejecutan un demonio de red especial, el cual debe ser considerado como un super servidor. Este demonio crea sockets en nombre de cada uno de los servicios y escucha en todos ellos simultáneamente. Cuando una conexión entrante es recibida en cualquiera de esos sockets, el super servidor acepta la conexión y replica el servicio especificado para ese puerto, pasando el socket a gestionarse a través del proceso hijo. El servidor entonces, vuelve a la escucha.

startup El super servidor más común se llama **inetd**, el Demonio de Internet[115]. Se inicia en tiempo de arranque del sistema y toma la lista de servicios que ha de gestionar de un fichero de inicialización llamado `/etc/inetd.conf`. Además de estos servidores, hay un número de servicios triviales realizados por **inetd** llamados *servicios internos*. Se incluyen **chargen**, el cuál simplemente genera una cadena de caracteres, y **daytime**, el cuál devuelve la idea del sistema de la hora del día.

Una entrada en este fichero consiste en una sola línea compuesta de los siguientes campos:

```
servicio tipo protocolo espera usuario servidor línea_de_órdenes
```

Cada uno de los campos se describe en la siguiente lista:

servicio
: Da el nombre del servicio. El nombre del servicio tiene que ser traducido a un número de puerto buscándolo en el fichero `/etc/services`. Este fichero se describirá más tarde en este capítulo, en la sección Sección 12.3.

tipo
: Especifica la clase de socket, o un socket de flujo stream (para protocolos orientados a la conexión) o un socket de datagrama dgram (para protocolos orientados a datagramas). Los servicios basados en TCP deberían entonces utilizar siempre sockets de flujo, stream, mientras que servicios basados en UDP deberían utilizar sockets de datagramas, dgram.

protocolo

Nombra el protocolo de transporte usado por el servicio. Debe ser un nombre válido de protocolo que se encuentre en el fichero `protocols`, expuesto más adelante.

espera
: Esta opción se aplica sólo a sockets dgram. Puede ser wait[116] o nowait[117]. Si se especifica wait, **inetd** ejecuta sólo un servidor para el puerto especificado. El otro modo, continúa escuchando inmediatamente en el puerto después de ejecutar el servicio.

  Esto es usado para servidores de hilo único[118] que leen todos los datagramas entrantes hasta que no llegan más, y después terminan. Muchos servidores RPC son de este tipo y tienen que ser especificados como wait. El tipo opuesto, multi−hilo, [119] permite la ejecución concurrente de un número ilimitado de instancias. Estos servidores deben especificarse como nowait.

  Los sockets de flujo, stream, deben usar siempre nowait.

usuario
: Esto es el identificador de registro del usuario[120] que será propietario del procesos mientras se esté ejecutando. Éste será muchas veces el usuario root, pero algunos servicios pueden usar cuentas distintas. Es una buena idea aplicar el principio del mínimo privilegio aquí, lo que significa que usted no debe ejecutar órdenes bajo cuentas privilegiadas si el programa no requiere esto para su correcto funcionamiento. Por ejemplo, el servidor de noticias NNTP ejecutado como news, mientras sirve noticias puede ser un riesgo de seguridad (como **tftp** o **finger**) que son muchas veces ejecutados como nobody.

servidor
: Proporciona el camino completo del programa servidor a ser ejecutado. Los servicios internos se marcan con la palabra clave internal.

línea_de_órdenes
: Esta es la línea de órdenes que se va a pasar al servidor. Comienza con el nombre del servidor a ejecutar y puede incluir cualquier argumento que se le necesiten pasar. Si está usando encapsulación TCP [121], especificará el camino completo al servidor aquí. Si no, entonces especificará el nombre del servidor como quiera que aparezca en un listado de procesos. Hablaremos acerca de encapsulación TCP brevemente.

  Este campo está vacío para los servicios internos.

Un ejemplo del fichero `inetd.conf` se expone en Ejemplo 12−1. El servicio **finger** está comentado así que no está disponible. Esto se hace a menudo por razones de seguridad, porque puede ser usado por atacantes para obtener nombres y otros detalles de los usuarios de su sistema.

Ejemplo 12−1. Un ejemplo del fichero /etc/inetd.conf

```
#
# inetd services
ftp      stream tcp nowait root  /usr/sbin/ftpd    in.ftpd −l
telnet   stream tcp nowait root  /usr/sbin/telnetd in.telnetd −b/etc/issue
#finger  stream tcp nowait bin   /usr/sbin/fingerd in.fingerd
#tftp    dgram  udp wait  nobody /usr/sbin/tftpd   in.tftpd
#tftp    dgram  udp wait  nobody /usr/sbin/tftpd   in.tftpd /boot/diskless
#login   stream tcp nowait root  /usr/sbin/rlogind in.rlogind
#shell   stream tcp nowait root  /usr/sbin/rshd    in.rshd
#exec    stream tcp nowait root  /usr/sbin/rexecd  in.rexecd
#
#       inetd internal services
```

```
#
daytime  stream tcp nowait root internal
daytime  dgram  udp nowait root internal
time     stream tcp nowait root internal
time     dgram  udp nowait root internal
echo     stream tcp nowait root internal
echo     dgram  udp nowait root internal
discard  stream tcp nowait root internal
discard  dgram  udp nowait root internal
chargen  stream tcp nowait root internal
chargen  dgram  udp nowait root internal
```

El demonio **tftp** se muestra comentado también. **tftp** implementa el *Trivial File Transfer Protocol* (TFTP), el cual permite transferir cualquier fichero con permisos de lectura globales desde su sistema sin verificación de contraseña. Esto es especialmente perjudicial para el fichero `/etc/passwd`, e incluso más cuando no usan claves shadow.

TFTP se usa normalmente por clientes sin disco duro y por Terminales X para descargar su código desde un servidor de arranque. Si necesita ejecutar **tftpd** por esta razón, asegúrese de limitar su alcance a aquellos directorios desde los cuáles los clientes obtendrán los ficheros; deberá añadir esos nombres de directorio a la línea de órdenes de **tftpd**. Esto se muestra en la segunda línea de **tftp** en el ejemplo.

12.2. La Facilidad de Control de Acceso tcpd

Debido a que abrir un computador al acceso de red implica muchos riesgos de seguridad, las aplicaciones están diseñadas para protegerse contra algunos tipos de ataques. Algunas características de seguridad, sin embargo, pueden tener fallos (más drásticamente demostrados por el gusano de Internet RTM, que explotaba un agujero en algunos programas, incluyendo versiones antiguas del demonio de correo Sendmail), o no distinguen entre hosts seguros cuyas peticiones de un servicio particular deberán ser aceptadas y hosts inseguros cuyas peticiones deberán ser rechazadas. Ya hemos visto brevemente los servicios **finger** y **tftp**. Los administradores de red deberían querer limitar el acceso de estos servicios a hosts de confianza únicamente, lo que es imposible con la configuración usual, por la que **inetd** ofrece este servicio o a todos los clientes o a ninguno.

Una herramienta útil para gestionar el acceso de hosts específicos es **tcpd**, a menudo llamado el demonio encapsulador [122].[123] Para los servicios TCP quiera monitorizar o proteger, se invoca en vez del programa servidor. **tcpd** verifica si el host remoto tiene permitido usar ese servicio, y sólo si esto tiene éxito ejecutará el programa servidor real. **tcpd** también deja registradas las peticiones en el demonio **syslog** . Nótese que no funciona para servicios basados en UDP.

Por ejemplo, para encapsular el demonio **finger**, debe cambiar la correspondiente línea en `inetd.conf` de esta forma:

```
# demonio finger desencapsulado
finger    stream tcp nowait bin    /usr/sbin/fingerd in.fingerd
```

a esta:

```
# demonio finger encapsulado
finger  stream  tcp     nowait  root    /usr/sbin/tcpd   in.fingerd
```

Sin añadir ningún control de acceso, al cliente le parecerá como la configuración usual de **finger**, excepto que todas las peticiones son registradas en la facilidad *auth* de **syslog**.

Dos ficheros llamados `/etc/hosts.allow` y `/etc/hosts.deny` implementan el control de acceso. Contienen entradas que permiten y deniegan acceso a ciertos servicios y hosts. Cuando **tcpd** gestiona una petición para un servicio como **finger** desde un host cliente llamado biff.foobar.com, se busca en `hosts.allow` y `hosts.deny` (en este orden) una entrada que coincida tanto con el servicio como con el host cliente. Si se encuentra la entrada correspondiente en `hosts.allow`, se autoriza el acceso y **tcpd** no consulta el fichero `hosts.deny`. Si no se encuentra una coincidencia en el fichero `hosts.allow`, pero se encuentra en el `hosts.deny`, la petición es rechazada cerrando la conexión. La petición es aceptada si no hay correspondencias en ninguno de los ficheros.

Las entradas en los ficheros de acceso tienen un aspecto como este:

```
lista_de_servicios: lista_de_hosts [:órdenes_de_shell]
```

lista_de_servicios es una lista de nombres de servicio de `/etc/services`, o la palabra clave ALL. Para hacer coincidir a todos los servicios excepto **finger** y **tftp**, se usa ALL EXCEPT `finger, tftp`.

lista_de_hosts es una lista de nombres de hosts, direcciones IP, o las palabras clave ALL, LOCAL, UNKNOWN o PARANOID. ALL coincide con cualquier host, mientras que LOCAL coincide con nombres de host que no contienen un punto.[124] UNKNOWN coincide con cualquier host cuya búsqueda de nombre o dirección falle. PARANOID coincide con cualquier host cuyo nombre de host no se resuelva de vuelta a su dirección IP. [125] Un nombre que empice por un punto hace coincidir todos los hosts cuyo dominio es igual a este nombre. Por ejemplo, .foobar.com coincide biff.foobar.com, pero no nurks.fredsville.com. Un patrón que termine con un punto coincide con cualquier host cuya dirección IP comience con el patrón proporcionado, así que 172.16. coincide con 172.16.32.0, pero no con 172.15.9.1. Un patrón de la forma *n.n.n.n/m.m.m.m* es tratado como una dirección IP y la máscara de red, así que podemos especificar nuestro ejemplo anterior como 172.16.0.0/255.255.0.0 en su lugar. Finalmente, cualquier patrón que empiece por el carácter / permite especificar un fichero que se presume que contiene una lista de nombres de host o patrones de direcciones IP, con cualquiera de los cuáles se permite la coincidencia. Así que un patrón que se parezca a */var/access/trustedhosts* causaría que el demonio **tcpd** lea este fichero, verificando si alguna de sus líneas coincide con el host que está conectándose.

Para denegar acceso a los servicios **finger** y **tftp** a todos menos al los hosts locales, ponga lo siguiente en `/etc/hosts.deny` y deje vacío `/etc/hosts.allow`:

```
    in.tftpd, in.fingerd: ALL EXCEPT LOCAL, .su.dominio
```

El campo opcional *orden_de_shell* puede contener una orden del intérprete de órdenes [126] para ser invocada cuando la entrada coincida. Esto es útil para establecer trampas que puedan exponer a los atacantes potenciales. El siguiente ejemplo crea un fichero de registros que lista al usuario y al host que se conecta, y si el host no es **vlager.vbrew.com** se concatenará la salida de un **finger** a ese host:

```
    in.ftpd: ALL EXCEPT LOCAL, .vbrew.com : \
          echo "request from %d@%h:" >> /var/log/finger.log; \
          if [ %h != "vlager.vbrew.com:" ]; then \
              finger -l @%h >> /var/log/finger.log \
          fi
```

Los argumentos %h y %d son expandidos por **tcpd** como el nombre del host del cliente y el nombre del servicio, respectivamente. Por favor dirígase a la página de manual `hosts_access(5)` para más detalles.

12.3. Los Ficheros de Servicios Y Protocolos

Los números de puerto en los que ciertos servicios estándar se ofrecen se definen en el RFC Assigned Numbers. Para permitir a los programas servidores y clientes convertir nombres de servicio a estos números, al menos parte del listado se mantiene en cada host; esto se almacena en un fichero llamado `/etc/services`. Una entrada se crea del siguiente modo:

```
servicio puerto/protocolo   [alias]
```

Aquí, *servicio* especifica el nombre de servicio, *puerto* define el puerto en el que el servicio se ofrece y *protocolo* define qué protocolo de transporte se usa. Comúnmente, el último campo es o *udp* o *tcp*. Es posible que un servicio sea ofrecido para más de un protocolo, así como ofrecer diferentes servicios en el mismo puerto mientras que los protocolos son diferentes. El campo *alias* le permite especificar nombres alternativos para el mismo servicio.

Normalmente, usted no tiene que cambiar el fichero de servicios que viene junto con el software de red de su sistema GNU/Linux. Sin embargo, le damos un pequeño extracto de ese fichero en Ejemplo 12−2.

Ejemplo 12−2. Un fichero /etc/services de ejemplo

```
# El fichero de servicios:
#
# servicios bien conocidos
echo           7/tcp                  # Eco
echo           7/udp                  #
discard        9/tcp  sink null       # Descartar
discard        9/udp  sink null       #
daytime       13/tcp                  # Hora
daytime       13/udp                  #
chargen       19/tcp  ttytst source   # Generador de caracteres
chargen       19/udp  ttytst source   #
ftp-data      20/tcp                  # Protocolo FTP (Datos)
ftp           21/tcp                  # Protocolo FTP (Control)
telnet        23/tcp                  # Protocolo de Terminales Virtuales
smtp          25/tcp                  # Protocolo de Correo SMTP
nntp         119/tcp  readnews        # Protocolo de Noticias NNTP
#
# UNIX services
exec         512/tcp                  # BSD rexecd
biff         512/udp  comsat          # notificación de correo
login        513/tcp                  # registro remoto (remote login)
who          513/udp  whod            # who y uptime remotos
shell        514/tcp  cmd             # órden remota, sin uso de contraseña
syslog       514/udp                  # sistema remoto de registros del sistema
printer      515/tcp  spooler         # colas de impresión remotas
route        520/udp  router routed   # protocoo de información de encaminamiento
```

Fíjese que el servicio **echo** se ofrece en el puerto 7 para TCP y UDP, y que el puerto 512 se usa para 2 servicios diferentes: ejecución remota (**rexec**) usando TCP, y el demonio **COMSAT**, el cual notifica a los usuarios que tienen correo nuevo, sobre UDP (vea **xbiff(1x)**).

Como el fichero de servicios, la biblioteca de red necesita una manera de traducir nombres de protocolopor ejemplo, esos usados en el fichero de servicios a los números de protocolo entendidos por la capa IP en otros hosts. Esto se hace mirando el nombre en el fichero `/etc/protocols`. Éste contiene una entrada por línea, cada una conteniendo el nombre del protocolo y el número asociado. Tener que tocar este fichero es incluso más inverosímil que hacerlo con `/etc/services`. Un ejemplo del fichero se proporciona en Ejemplo 12−3.

Ejemplo 12−3. Un fichero /etc/protocols de ejemplo

```
#
# Protocolos de Internet (IP)
#
ip      0       IP              # protocolo de internet, número de pseudo protocolo
icmp    1       ICMP            # protocolo ICMP
igmp    2       IGMP            # Protocolo IGMP
tcp     6       TCP             # Protocolo de control de transmisión (TCP)
udp     17      UDP             # Protocolo UDP
raw     255     RAW             # Interfaz RAW IP
```

12.4. Llamada a Procedimiento Remoto

El mecanismo general para las aplicaciones cliente−servidor se proporciona por el paquete *Remote Procedure Call* (RPC). RPC fue desarrollado por Sun Microsystems y es una colección de herramientas y funciones de biblioteca. Aplicaciones importantes construidas sobre RPC son NIS, Sistema de Información de Red[127] (descrito en Capítulo 13), y NFS, Sistema de Ficheros de Red[128] (descrito en Capítulo 14), ambos se describen en este libro.

Un servidor RPC consiste en una colección de procedimientos que un cliente puede solicitar por el envío de una petición RPC al servidor junto con los parámetros del procedimiento. El servidor invocará el procedimiento indicado en nombre del cliente, entregando el valor de retorno, si hay alguno. Para ser independiente de la máquina, todos los datos intercambiados entre el cliente y el servidor se convierten al formato *External Data Representation* [129] (XDR) por el emisor, y son reconvertidos a la representación local por el receptor. RPC confía en sockets estandard UDP y TCP para transportar los datos en formato XDR hacia el host remoto. Sun amablemente a puesto RPC en el dominio público; se describe en una serie de RFCs.

A veces las mejoras en una aplicación RPC introducen cambios incompatibles con la interfaz de llamada a procedimientos. Por supuesto, simplemente cambiando el servidor hará que no funcionen todas las aplicaciones que todavía esperen el comportamiento original. Por lo tanto, los programas RPC tienen números de versión asignados, casi siempre empezando por 1, y con cada nueva versión de la interfaz RPC, este contador se incrementa. A menudo, un servidor puede ofrecer varias versiones simultáneamente; entonces los clientes indican a través del número de versión en la petición que implementación del servicio quieren usar.

La comunicación entre servidores RPC y clientes es un tanto peculiar. Un servidor RPC ofrece una o más colecciones de procedimientos; cada conjunto se llama un *programa* y es idenficado de forma única por un *número de programa*. Una lista que relaciona nombres de servicio con números de programa se mantiene usualmente en `/etc/rpc`, un extracto del cual se ve en Ejemplo 12−4.

Ejemplo 12−4. Una muestra de fichero /etc/rpc

```
#
# /etc/rpc - servición miscaláneos basados en RPC
#
portmapper      100000  portmap sunrpc
rstatd          100001  rstat rstat_svc rup perfmeter
rusersd         100002  rusers
nfs             100003  nfsprog
ypserv          100004  ypprog
mountd          100005  mount showmount
ypbind          100007
walld           100008  rwall shutdown
yppasswdd       100009  yppasswd
bootparam       100026
ypupdated       100028  ypupdate
```

En redes TCP/IP , los autores de RPC se enfrentan al problema del mapeo de números de programa con servicios genéricos de red. Diseñaron cada servidor para proveer ambos puertos TCP y UDP para cada programa y cada versión. Generalmente, las aplicaciones RPC usan UDP cuando envían datos, y vuelven a TCP sólo cuando los datos a transferir no caben en un solo datagrama UDP.

Por supuesto, los programas cliente necesitan averiguar a qué puerto se refiere un número de programa. Usar un fichero de configuración para esto podría ser demasiado inflexible; debido a que las aplicaciones RPC no usan puertos reservados, no hay garantía de que un puerto originalmente usado por nuestra aplicación de base de datos, no haya sido tomado por cualquier otro proceso. Por lo tanto, las aplicaciones RPC toman cualquier puerto que puedan obtener y lo registran con un programa especial llamado el *demonio* *portmapper*[130]. El mapeador de puertos actúa como un intermediario para todos los servidores RPC ejecutándose en su máquina. Un cliente que desea contactar con un servicio con un número de programa dado primero pregunta al mapeador de puertos en el host del servidor, el cuál devuelve el número de puerto TCP y UDP en donde el servicio puede ser alcanzado.

Este método introduce un solo punto de fallo, similar a como el demonio **inetd** hace para los servicios estándar de Berkeley. Sin embargo, este caso es aún un poco peor porque cuando el mapeador de puertos muere, toda la información de los puertos RPC se pierde; esto a menudo significa que debe reiniciar todos los servidores RPC manualmente o reiniciar la máquina.

En Linux, el mapeador de puertos se llama `/sbin/portmap`, o a veces `/usr/sbin/rpc.portmap`. Una vez que se cerciora de que se inicia desde sus guiones de inicio de red, el mapeador de puertos no requiere ninguna configuración.

12.5. Configurando el Registro y Ejecución Remotos

Es a menudo muy útil ejecutar una orden en un host remoto y que la entrada o la salida de esa orden pueda leerse o escribirse a través de una conexión de red.

Los programas tradicionales para ejecutar órdenes en hosts remotos son **rlogin**, **rsh** y **rcp**. Vimos un ejemplo de la orden **rlogin** en Capítulo 1 en la sección Sección 1.2.1. vimos brevemente cuestiones de seguridad asociadas con esto en Sección 1.5.1 y sugerimos **ssh** como alternativa. El paquete **ssh** proporciona unos reemplazos llamados **slogin**, **ssh**, y **scp**.

Cada una de estas órdenes genera un intérprete de órdenes en el host remoto y permite al usuario ejecutar órdenes. Por supuesto, el cliente necesita tener una cuenta en el host remoto donde la orden va a ser ejecutada. Así, todas estas órdenes usan un proceso de autentificación. Las órdenes *r* usan un simple intercambio de nombre de usuario y contraseña entre los hosts sin encriptación, de este modo cualquiera que esté escuchando puede fácilmente interceptar las contraseñas. El conjunto de órdenes **ssh** proporcionan un nivel de seguridad más alto: usan una técnica llamada **Criptografía de Clave Pública**, la cual proporciona autentificación y encriptación entre los hosts para asegurar que tanto contraseñas como datos de la sesión sean interceptados por otros hosts.

Es posible relajar la comprobación de la autentificación para ciertos usuarios todavía más. Por ejemplo, si usted tiene que registrarse en otras máquinas de su red frecuentemente, usted puede querer ser admitido sin tener que teclear su contraseña cada vez. Esto era posible con las órdenes *r*, pero las órdenes **ssh** le permiten hacer esto algo más sencillo. Esto no es una gran idea porque significa que si una cuenta de una máquina es violada, se puede ganar el acceso a otras cuentas que el usuario ha configurado para registrarse sin password, pero esto es muy conveniente y la gente quiere usarlo.

Hablemos acerca de quitar las órdenes *r* y usar **ssh** para trabajar en su lugar.

12.5.1. Desactivando las Órdenes r

Comencemos retirando las órdenes **r** si están instaladas. La forma más fácil de desactivar las órdenes **r** antiguas es comentando (o borrando) sus entradas en el fichero `/etc/inetd.conf`. Las entradas relevantes se parecen a algo como esto:

```
# Shell, login, exec y talk como protocolos BSD.
shell    stream  tcp     nowait  root  /usr/sbin/tcpd /usr/sbin/in.rshd
login    stream  tcp     nowait  root  /usr/sbin/tcpd /usr/sbin/in.rlogind
exec     stream  tcp     nowait  root  /usr/sbin/tcpd /usr/sbin/in.rexecd
```

Puede comentarlas anteponiendo el carácter # al principio de cada línea, o borrando las líneas completamente. Recuerde, necesitará reiniciar el demonio **inetd** para que este cambio tenga efecto. Idealmente, debería retirar los mismos ejecutables también.

12.5.2. Instalando y Configurando ssh

OpenSSH es una versión libre del conjunto de programas ssh; el porte para GNU/Linux se puede encontrar en http://violet.ibs.com.au/openssh/ y en muchas distribuciones modernas de GNU/Linux.[131] No explicaremos aquí la compilación; se incluyen buenas instrucciones en el código fuente. Si usted puede instalarlo desde un paquete precompilado, es probablemente inteligente hacerlo así.

Hay dos partes en una sesión **ssh**. Hay un cliente **ssh** que usted necesita configurar y ejecutar en el host local y un demonio **ssh** que debe ejecutarse en el host remoto.

12.5.2.1. El Demonio ssh

El demonio **sshd** es el programa que escucha conexiones de red desde clientes **ssh**, gestiona la autentificación, y ejecuta las órdenes requeridas por el cliente. Hay un fichero de configuración principal llamado `/etc/ssh/sshd_config` y un fichero especial que contiene una clave usada por los procesos de autentificación y encriptación para representar la parte del host. Cada host y cada cliente tienen su propia clave.

Una utilidad llamada **ssh−keygen** se proporciona para generar un clave aleatoria. Esto comúnmente se usa una vez en la instalación para generar la clave del host, la cual el administrador de sistema guarda en un fichero llamado `/etc/ssh/ssh_host_key`. Las claves pueden ser de cualquier longitud de 512 bits o mayores. Por omisión, **ssh−keygen** genera claves de 1024 bits de longitud, y la mayoría de la gente usa lo predeterminado. Para generar una clave aleatoria, debe invocar la orden **ssh−keygen** así:

```
# ssh-keygen -f /etc/ssh/ssh_host_key
```

Se le pedirá que introduzaca una frase de paso. Sin embargo, las claves host no deben usar frase de paso, en este caso pulse la tecla return para dejarla en blanco. La salida del programa será algo así:

```
Generating RSA keys:  ......oooooO..............................oooooO
Key generation complete.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_key
Your public key has been saved in /etc/ssh/ssh_host_key.pub
The key fingerprint is:
1024 3a:14:78:8e:5a:a3:6b:bc:b0:69:10:23:b7:d8:56:82 root@moria
```

Puede encontrar al final que los dos ficheros han sido creados. El primero se llama la clave privada, el cual debe mantenerse en secreto y estará en `/etc/ssh/ssh_host_key`. El segundo se llama la clave pública y es el que puede compartir; estará en `/etc/ssh/ssh_host_key.pub`.

Armados con las claves para la comunicación **ssh**, necesita crear un fichero de configuración. las órdenes **ssh** son muy potentes y el fichero de configuración puede contener muchas opciones. Expondremos un ejemplo sencillo para que empiece; debe dirigirse a la documentación de **ssh** para activar otras características. El siguiente código muestra un fichero de configuración seguro y mínimo de **sshd** . El resto de las opciones de configuración se detallan en las páginas del manual de **sshd**(8) :

```
# /etc/ssh/sshd_config
#

# Las direcciones IP que escuchan conexiones entrantes. 0.0.0.0 significa todas las
# direcciones locales
ListenAddress 0.0.0.0
```

```
# El puerto TCP que escucha conexiones entrantes. Por omisión el 22.
Port 22

# El nombre del fichero clave del host.
HostKey /etc/ssh/ssh_host_key

# La longitud de la clave en bits.
ServerKeyBits 1024

# ¿Debemos permitir registros (login) del root por ssh?
PermitRootLogin no

# ¿Debe el demonio ssh verificar que el directorio inicial (home) del usuario y los permisos d
# sean seguros antes de permitir el registro (login)?
StrictModes yes

# ¿Debemos permitir el método antiguo de autentificación  ~/.rhosts y /etc/hosts.equiv?
RhostsAuthentication no
# ¿Debemos permitir autenticación pura RSA?
RSAAuthentication yes
# ¿Debemos permitir autenticación por contraseña?
PasswordAuthentication yes

# ¿Debemos permitir /etc/hosts.equiv combinado con autentificación host por RSA?
RhostsRSAAuthentication no
# ¿Ignorar los ficheros ~/.rhosts?
IgnoreRhosts yes
# ¿Permitimos registros (logins) a cuentas con contraseñas vacías?
PermitEmptyPasswords no
```

Es importante estar seguro de que los permisos de los ficheros de configuración son correctos para asegurar que se mantiene el sistema de seguridad. Use las siguientes órdenes:

```
# chown -R root:root /etc/ssh
# chmod 755 /etc/ssh
# chmod 600 /etc/ssh/ssh_host_key
# chmod 644 /etc/ssh/ssh_host_key.pub
# chmod 644 /etc/ssh/sshd_config
```

La etapa final de la administración del demonio **sshd** es ejecutarlo. Normalmente necesitará crear un fichero `rc` para ello o añadirlo a uno existente, de este modo se ejecutará automáticamente en el arranque. El demonio corre solo y no necesita ninguna entrada en el fichero `/etc/inetd.conf` . El demonio debe correr como usuario `root` . La sintaxis es simple:

```
/usr/sbin/sshd
```

El demonio **sshd** automáticamente se ejecutará en segundo plano. Ahora esta listo para aceptar conexiones *ssh*.

12.5.2.2. El cliente ssh

Existen variedad de programas clientes **ssh**: **slogin**, **scp** y **ssh**. Cada uno lee el mismo fichero de configuración, normalmente llamado `/etc/ssh/ssh_config`. Cada uno de ellos también lee ficheros de configuración desde el directorio `.ssh` en el directorio inicial (home) del usuario que lo esté ejecutando. El más importante de estos ficheros es el `.ssh/config`, el cual puede contener opciones que sustituirán a las especificadas en el fichero `/etc/ssh/ssh_config`, el fichero `.ssh/identity`, el cual contiene la propia clave privada del usuario, y el correspondiente fichero `.ssh/identity.pub`, conteniendo la clave pública propia del usuario. Otros ficheros importantes son `.ssh/known_hosts` y `.ssh/authorized_keys`; hablaremos de ellos después en Sección 12.5.2.3. Primero, vamos a crear el fichero de configuración global y el fichero de claves de usuario.

El fichero `/etc/ssh/ssh_config` es muy similar al de configuración del servidor. Otra vez, tenemos muchas características que usted puede configurar, pero una configuración minima puede ser como la expuesta en Ejemplo 12−5. El resto de las opciones de configuración están detalladas en la página de manual **sshd(8)**. Puede añadir secciones que coincidan con hosts específicos o grupos de hosts. El parámetro a la declaración `Host` puede ser cualquiera de los nombres completos de un host o una especificación de carácter comodín, como hemos usado en nuestro ejemplo, para que coincidan todos los hosts. Podemos crear una entrada que usada, por ejemplo, `Host *.vbrew.com` haga coincidir cualquier host en el dominio `vbrew.com`.

Ejemplo 12−5. Ejemplo De Fichero de Configuración del Cliente ssh

```
# /etc/ssh/ssh_config

# Opciones predeterminadas a usar cuando se conecte a un host remoto
Host *
  # ¿Comprimir los datos de sesión?
  Compression yes
  # .. usando qué nivel de compresión? (1 − rápida/escasa, 9 − lenta/mucha)
  CompressionLevel 6

  # ¿Usar  rsh si la conexión segura falla?
  FallBackToRsh no

  # ¿Debemos mandar mensajes para mantener la conexión (keep−alive)? Util si se usa enmascaram
  KeepAlive yes

  # ¿Intentar autentificación RSA?
  RSAAuthentication yes
  # ¿Intentar autentificación RSA en combinación con autentificación .rhosts?
  RhostsRSAAuthentication yes
```

Mencionamos en la sección de configuración de servidor que cada host y cada usario tiene una clave. La clave de usuario se guarda en su fichero `~/.ssh/indentity` . Para generar la clave, se usa la misma orden **ssh−keygen** que usamos para generar la clave de host, excepto que esta vez no necesita especificar el nombre del fichero donde usted guarda la clave. **ssh−keygen** tiene predeterminada la localización correcta, pero le pregunta que introduzca un nombre de fichero en el caso que usted no quiera éste. Es útil algunas veces para tener ficheros de identidad diferentes, así que **ssh** permite esto. Como antes, **ssh−keygen** le preguntará que introduzca una frase de paso. Las frases de paso añaden otro nivel de seguridad y son una buena idea. Su frase de paso no será impresa en pantalla cuando usted la teclee.

Aviso
No hay forma de recuperar una frase de paso si la olvida. Cercióorese de que será algo que usted recordará, pero como toda contraseña, elija algo que no sea obvio, como nombres propios o su nombre. Para que una frase de paso sea efectiva, debe tener entre 10 y 30 caracteres de longitud y no debe ser prosa inglesa simple. Pruebe incluir algunos caracteres no usuales. Si usted pierde su frase de paso, deberá generar una clave nueva.

Debería preguntar a cada uno de sus usuarios si han ejecutado la orden **ssh−keygen** para asegurarse de que sus ficheros de claves se han generado correctamente. El **ssh−keygen** creará sus directorios `~/.ssh/` para ellos con los permisos apropiados y creará su clave privada y pública en `.ssh/identity` y `.ssh/identity.pub`, respectivamente. Un ejemplo de sesión se muestra aquí:

```
$ ssh−keygen
Generating RSA keys:  .......oooooO.............................
Key generation complete.
Enter file in which to save the key (/home/maggie/.ssh/identity):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/maggie/.ssh/identity.
```

```
Your public key has been saved in /home/maggie/.ssh/identity.pub.
The key fingerprint is:
1024 85:49:53:f4:8a:d6:d9:05:d0:1f:23:c4:d7:2a:11:67 maggie@moria
$
```

Ahora **ssh** esta listo para ejecutarse.

12.5.2.3. Utilización de ssh

Ahora tenemos la orden **ssh** y sus programas asociados instalados y listos para ejecutarse. Veamos rápidamente como se ejecutan.

Primero, probaremos un registro (login) remoto a un host. Podemos usar el programa **slogin** de la misma forma que usamos el programa **rlogin** en nuestro ejemplo anterior en el libro. La primera vez que esperamos conectarnos a un host, el cliente **ssh** recuperará la clave publica del host y le preguntará si confirma esta identidad instándole con una versión reducida de la clave pública llamada **huella dactilar**[132].

El administrador del host remoto le debería haber proporcinado previamente estas huellas dactilares, las cuáles usted debe añadir a su fichero `.ssh/known_hosts` . Si el administrador remoto no le ha dado las claves apropiadas, usted puede conectarse al host remoto, pero **ssh** le advertirá que no tiene una clave y le pedirá que acepte una ofrecida por el host remoto. Asumiendo que usted está seguro que nadie le engaña con DNS spoofing y que usted de hecho está hablando con el host correcto, conteste yes. La clave se guarda automáticamente en su `.ssh/known_hosts` y no se le preguntará otra vez. Si, en un futuro intento de conexión, la clave pública recuperada desde este host no coincide con la que hay guardada, se le advertirá, porque esto representa un agujero de seguridad potencial.

La primera vez que conectamos con un host remoto veremos algo como esto:

```
$ slogin vchianti.vbrew.com
The authenticity of host 'vchianti.vbrew.com' can't be established.
Key fingerprint is 1024 7b:d4:a8:28:c5:19:52:53:3a:fe:8d:95:dd:14:93:f5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'vchianti.vbrew.com,172.16.2.3' to the list of/
    known hosts.
maggie@vchianti.vbrew.com's password:
Last login: Tue Feb  1 23:28:58 2000 from vstout.vbrew.com
$
```

Se le pedirá una clave, debe contestar con la clave de la cuenta remota, no con la local. Esta clave no tendrá eco por pantalla cuando la introduzca.

Sin ningún argumento especial, **slogin** intentará utilizar el mismo identificador de usuario que en la máquina local. Puede cambiar esto usando el argumento `-l` , dando un nombre de registro alternativo en el host remoto. Esto que lo que hicimos en nuestro ejemplo anterior en el libro.

Podemos copiar ficheros hacia y desde un host remoto usando el programa **scp**. Su sintaxis es similar al convencional **cp** con la excepción que debe especificar un nombre de host antes del fichero, significando que el camino del fichero está en el host especificado. El siguiente ejemplo ilustra la sintaxis de **scp** copiando un fichero local llamado `/tmp/fred` al `/home/maggie/` del host remoto **chianti.vbrew.com**:

```
$ scp /tmp/fred vchianti.vbrew.com:/home/maggie/
maggie@vchianti.vbrew.com's password:
fred                 100% |*****************************| 50165   00:01 ETA
```

De nuevo, se le pedirá una clave. La orden **scp** muestra el progreso de la copia por omisión. Puede copiar un fichero desde un host remoto con la misma facilidad; simplemente especificando su nombre de host y ruta como origen y la ruta local como destino. También se puede copiar un fichero desde un host remoto a otro host remoto, pero habitualmente no necesitará hacer eso, porque todos los datos viajan a través de su host.

Puede ejecutar órdenes en host remotos usando la orden **ssh**. De nuevo, su sintaxis es muy simple. Tengamos nuestro usuario **`maggie`** recuperando el directorio raíz del host remoto **vchianti.vbrew.com**. Ella hará algo como esto:

```
$ ssh vchianti.vbrew.com ls -CF /
maggie@vchianti.vbrew.com's password:
bin/    console@  dos/     home/    lost+found/  pub@   tmp/  vmlinuz@
boot/   dev/      etc/     initrd/  mnt/         root/  usr/  vmlinuz.old@
cdrom/  disk/     floppy/  lib/     proc/        sbin/  var/
```

Puede utilizar **ssh** con tuberías y entubar entradas/salidas de programas desde o hacia como cualquier otra orden, excepto que la entrada o la salida son dirigidas hacia o desde el host remoto mediante conexión **ssh**. Aquí tenemos un ejemplo de como puede utilizar esta característica en combinación con la orden **tar** para copiar un directorio entero con subdirectorios y ficheros desde un host remoto al host local:

```
$ ssh vchianti.vbrew.com "tar cf - /etc/" | tar xvf -
maggie@vchianti.vbrew.com's password:
etc/GNUstep
etc/Muttrc
etc/Net
etc/X11
etc/adduser.conf
..
..
```

Hacemos notar que la orden se debe ejecutar con comillas para clarificar qué se está pasando como un argumento a la orden **ssh** y qué debe usar el intérprete de órdenes local. Esta orden ejecuta la orden **tar** en el host remoto para archivar el directorio `/etc/` y escribir en la salida estándar. Hemos entubado una instancia de la orden **tar** ejecutando en nuestro host local en modo extracción leyendo desde la entrada estándar.

De nuevo, se pide una clave. ¡Ahora puede ver por qué le animamos a configurar **ssh** para que no se le pida las claves todo el tiempo! Vamos ahora a configurar nuestro cliente local **ssh** de modo que no nos pida la clave cuando conectemos al host vchianti.vbrew.com. Mencionamos antes el fichero `.ssh/authorized_keys`; aquí es donde se va a usar. El fichero `.ssh/authorized_keys` contiene las claves *públicas* de cada cuenta de usuario remota que queremos registrar automáticamente. Puede establecer registros automáticos copiando el contenido del `.ssh/identity.pub` desde la cuenta *remota* en nuestro fichero local `.ssh/authorized_keys`. Es vital que los permisos de fichero de `.ssh/authorized_keys` permitan sólo que usted pueda leer y escribir; cualquiera puede robar y usar las claves para registrarse en esas cuentas remotas. Para asegurar que los permisos sean correctos, cambie `.ssh/authorized_keys`, como sigue:

```
$ chmod 600 ~/.ssh/authorized_keys
```

Las claves públicas son una larga *sencilla* línea de texto plano. Si usa copiar y pegar para duplicar la clave en su fichero local, asegúrese de borrar cualquier carácter de final de línea que se pueden haber introducido de esta manera. El fichero `.shh/uathorized_keys` puede contener muchas de estas claves, cada una en una línea propia.

La suite de herramientas **ssh** es muy potente y tiene muchas otras características y opciones que pueden ser interesantes de explorar. Por favor consulte las páginas del manual y otros documentos que se proporcionan con los paquetes para más información.

Capítulo 13. El Sistema de Información de Red (NIS)

Cuando se usa una red de área local, la meta final suele ser proporcionar un entorno que haga la red transparente a los usuarios. Un paso importante es mantener los datos vitales, como la información de las cuentas de usuario, sincronizados a lo largo de todas las máquinas. Esto proporciona a los usuarios la libertad de moverse de máquina en máquina sin el inconveniente de tener que recordar contraseñas diferentes y copiar datos de una máquina a otra. Los datos que están almacenados centralmente no necesitan ser replicados mientras exista un medio de acceder a ellos desde un nodo conectado a la red. Al almacenar centralmente la información administrativa importante, se consigue asegurar la consistencia de esos datos, aumentar la flexibilidad a los usuarios permitiéndoles moverse de nodo a nodo de manera transparente, y hacerle la vida mucho más fácil al administrador, al tener que mantener sólo una copia individual de la información.

Anteriormente hemos discutido un ejemplo importante de este concepto que se utiliza en Internetel Sistema de Nombres de Dominio (DNS). DNS sirve un rango limitado de información, siendo la más importante la correspondencia entre el nombre de nodo y la dirección IP. Para otros tipos de información, no existe un servicio especializado así. Por otra parte, si usted sólo administra una pequeña LAN sin conectividad a Internet, no parece que merezca la pena configurar DNS.

Ésta es la razón por la que Sun desarrolló el *Sistema de Información de Red* (NIS). NIS proporciona prestaciones de acceso a bases de datos genéricas que pueden utilizarse para distribuír, por ejemplo, la información contenida en los ficheros `passwd` y `groups` a todos los nodos de su red. Esto hace que la red parezca un sistema individual, con las mismas cuentas en todos los nodos. De manera similar, usted puede usar NIS para distribuír la información de nombres de nodo contenida en `/etc/hosts` a todas las máquinas de la red.

NIS está basado en RPC, y consta de un servidor, una biblioteca de la parte cliente, y varias herramientas de administración. Originalmente NIS se llamaba *Páginas Amarillas* (Yellow Pages), o YP, que todavía se utiliza para referirse a él. Desafortunadamente, ese nombre es una marca registrada de British Telecom, que exigió a Sun abandonar ese nombre. Al pasar el tiempo, algunos nombres se aferran en la mente de la gente, y así YP permanece como prefijo en los nombres de la mayoría de laso órdenes relacionadas con NIS, como **ypserv** y **ypbind**.

Hoy NIS está disponible prácticamente en todos los Unixes, e incluso existen implementaciones libres. BSD Net−2 publicó una que ha sido derivada de una implementación de referencia de dominio público donada por Sun. El código de la biblioteca de la parte cliente de esta versión existe en la `libc` de Linux desde hace mucho tiempo, y los programas de administración fueron portados a Linux por Swen Thümmler.[133] Sin embargo, falta un servidor NIS a partir de la implementación de referencia.

Peter Eriksson ha desarrollado una implementación nueva llamada NYS.[134] Soporta tanto NIS básico como la versión mejorada de Sun NIS+. NYS no sólo proporciona una serie de herramientas NIS y un servidor, sino que también añade un completo juego nuevo de funciones de biblioteca que necesita compilar en su `libc` si quiere utilizarlas. Esto incluye un esquema nuevo de configuración para la resolución de nombres de nodo que sustituye al esquema actual que usa el fichero `host.conf`.

La libc de GNU, conocida como `libc6` en la comunidad Linux, incluye una versión actualizada del soporte de NIS tradicional desarrollado por Thorsten Kukuk.[135] Soporta todas las funciones de biblioteca que proporcionaba NYS, y también utiliza el esquema avanzado de configuración de NYS. Todavía se necesitan las herramientas y el servidor, pero utilizando la `libc` de GNU se ahorra el trabajo de tener que parchear y recompilar la biblioteca.

Este capítulo se centra en el soporte de NIS incluído en la `libc` de GNU en vez de en los otros dos paquetes. Si usted quiere utilizar alguno de éstos, las instrucciones de este capítulo pueden o no ser suficientes. Si quiere información adicional, remítase al NIS−Como o a un libro como el *Managing NFS and NIS* de Hal Stern (O'Reilly).

13.1. Familiarizándose con NIS

NIS guarda la información de la base de datos en ficheros llamados *mapas*, que contienen pares clave−valor. Un ejemplo de par clave−valor es el identificativo de un usuario (username) y la forma encriptada de su contraseña. Los mapas se almacenan en un nodo central que corre el servidor NIS, desde el que los clientes deben obtener la información mediante varias llamadas RPC. Con bastante frecuencia, los mapas se almacenan en ficheros DBM. [136]

Los mapas suelen generarse a partir de ficheros de texto maestros como el `/etc/hosts` o el `/etc/passwd`. Para algunos ficheros se crean varios mapas, uno para cada tipo de clave de búsqueda. Por ejemplo, usted puede buscar en el fichero `hosts` tanto nombres de nodo como direcciones IP. Así pues, de él se derivan dos mapas NIS, llamados `hosts.byname` y `hosts.baddr`. La Tabla 13−1 muestra una lista de mapas comunes y los ficheros a partir de los que se generan.

Tabla 13−1. Algunos Mapas NIS Estándar y sus Correspondientes Ficheros

Fichero Maestro	Mapa(s)	Descripción
`/etc/hosts`	`hosts.byname`, `hosts.byaddr`	Corresponde direcciones IP con nombres de nodo
`/etc/networks`	`networks.byname`, `networks.byaddr`	Corresponde direcciones IP de red con nombres de red
`/etc/passwd`	`passwd.byname`, `passwd.byuid`	Corresponde contraseñas encriptadas con identificativos de usuario
`/etc/group`	`group.byname`, `group.bygid`	Corresponde IDs de Grupo con nombres de grupo
`/etc/services`	`services.byname`, `services.bynumber`	Corresponde descripciones de servicio con nombres de servicio
`/etc/rpc`	`rpc.byname`, `rpc.bynumber`	Corresponde números de servicio Sun RPC con nombres de servicio RPC
`/etc/protocols`	`protocols.byname`, `protocols.bynumber`	Corresponde números de protocolo con nombres de protocolo
`/usr/lib/aliases`	`mail.aliases`	Corresponde alias de correo con nombres de alias de correo

Puede encontrar soporte para otros ficheros y mapas en otros paquetes NIS. Normalmente contienen información sobre aplicaciones que no se discuten en este libro, como el mapa `bootparams` utilizado por el servidor `bootparamd` de Sun.

Hay mapas para los que la gente usa normalmente *apodos*, que son más cortos y por tanto más fáciles de escribir. Tenga en cuenta que estos apodos sólo los entienden **ypcat** e **ypmatch**, dos herramientas para comprobar su configuración NIS. Para obtener una lista completa de los apodos que entienden estas herramientas, ejecute la siguiente orden:

```
$ ypcat −x
Use "passwd" for "passwd.byname"
Use "group" for "group.byname"
Use "networks" for "networks.byaddr"
Use "hosts" for "hosts.byaddr"
Use "protocols" for "protocols.bynumber"
Use "services" for "services.byname"
Use "aliases" for "mail.aliases"
Use "ethers" for "ethers.byname"
```

El servidor NIS se llama tradicionalmente **ypserv**. Para una red mediana, normalmente un solo servidor es suficiente; las redes grandes pueden elegir ejecutar varios de estos servidores en máquinas diferentes y en segmentos de red diferentes para reducir la carga en las máquinas servidor y en los enrutadores. Estos servidores se sincronizan haciendo a uno de ellos el *servidor maestro*, y a los otros *servidores esclavos*. Los mapas se crean sólo en el nodo del servidor maestro. Desde él se distribuyen a todos los esclavos.

Hemos hablado muy vagamente sobre redes. Hay un término distintivo en NIS que se refiere a una colección de todos los nodos que comparten parte de sus datos de configuración de sistema a través de NIS: el *dominio NIS*. Desafortunadamente, los dominios NIS no tienen absolutamente nada en común con los dominios que nos encontramos en DNS. Para evitar cualquier ambigüedad a lo largo de este capítulo, siempre especificaremos a qué tipo de dominio nos referimos.

Los dominios NIS tienen una función puramente administrativa. En general son transparentes a los usuarios, excepto al compartir contraseñas entre todas las máquinas del dominio. Por tanto, el nombre que se le da a un dominio NIS es relevante sólo para los administradores. Normalmente, cualquier nombre servirá, mientras sea distinto a cualquier otro dominio NIS de su red local. Por ejemplo, la administradora de la Cervecera Virtual puede querer crear dos dominios NIS, uno para la propia Cervecera, y otro para la Vinatera, a los que llamará cervecera y vinatera respectivamente. Otro proceder común es usar simplemente el dominio DNS como dominio NIS.

Para establecer y mostrar el dominio NIS de su nodo, puede usar la orden **domainname**. Cuando se invoca sin argumentos, imprime el dominio NIS actual; para establecer el dominio, hace falta ser superusuario:

```
# domainname cervecera
```

Los dominios NIS determinan a qué servidor NIS consultará una aplicación. Por ejemplo, el programa **login** de un nodo de la Vinatera debe, por supuesto, consultar sólo al servidor NIS de la Vinatera (o a uno de ellos, si hay varios) la contraseña de un usuario, mientras que una aplicación de un nodo de la Cervecera debe llamar al servidor de la Cervecera.

Queda un misterio por resolver: ¿cómo averigua un cliente a qué servidor conectarse? La solución más simple sería utilizar un fichero de configuración que diga el nombre del nodo que hace de servidor. Sin embargo, esta solución es algo inflexible porque no permite a los clientes utilizar diferentes servidores (del mismo dominio, claro) dependiendo de su disponibilidad. Por tanto, las implementaciones de NIS cuentan con un demonio especial llamado **ypbind** para detectar un servidor NIS adecuado dentro del dominio NIS. Antes de realizar una consulta NIS, una aplicación averigua primero qué servidor usar mediante **ypbind**.

ypbind busca servidores haciendo un *broadcast* a la red IP local; se asume que el primero en responder es el más rápido, y es el utilizado en todas las consultas NIS subsiguientes. Después de que ha transcurrido un cierto intervalo de tiempo, o si el servidor deja de estar disponible, **ypbind** busca de nuevo servidores activos.

La ligadura dinámica es útil sólo cuando su red proporciona más de un servidor NIS. Además, la ligadura dinámica introduce un problema de seguridad. **ypbind** cree ciegamente en cualquiera que responda, sea un humilde servidor NIS o un intruso malicioso. No es necesario decir que esto es especialmente problemático si usted maneja sus bases de datos de contraseñas a través de NIS. Para protegerse de esto, el programa **ypbind** de Linux le proporciona la opción de buscar un servidor NIS en la red local o configurar el nombre de nodo del servidor NIS en un fichero de configuración.

13.2. NIS Versus NIS+

NIS y NIS+ comparten poco más que el nombre y una meta común. NIS+ se estructura de manera completamente diferente a NIS. En lugar de un espacio de nombres horizontal con dominios NIS desligados, NIS+ utiliza un espacio de nombres jerárquico similar al de DNS. En lugar de mapas, se utilizan las conocidas *tablas* constituidas por filas y columnas, en las que cada fila representa un objeto en la base de datos de NIS+, y las columnas cubren propiedades de los objetos que NIS+ conoce y trata. Cada tabla para un dominio NIS+

dado incluye las de sus dominios padre. Además, una entrada en una tabla puede contener un enlace a otra tabla. Estas características hacen posible estructurar la información de muchas maneras.

Adicionalmente, NIS+ soporta llamadas RPC seguras y encriptadas, que ayuda mucho a resolver los problemas de seguridad de NIS.

El NIS tradicional tiene un número de versión de RPC de 2, mientras que NIS+ usa la versión 3. Al tiempo de escribir esto, todavía no hay una buena implementación de NIS+ para Linux, por lo que no está cubierto aquí.

13.3. La Parte Cliente en NIS

Si está familiarizado con escribir o portar aplicaciones de red, puede haberse dado cuenta de que la mayoría de los mapas NIS listados anteriormente corresponden a funciones de biblioteca de la biblioteca C. Por ejemplo, para obtener la información de `passwd`, generalmente se utilizan las funciones `getpwnam` y `getpwuid`, que devuelven la información de cuenta asociada con el nombre de usuario o el ID numérico de usuario, respectivamente. Bajo circunstancias normales, estas funciones realizan la búsqueda requerida en el fichero estándar, `/etc/passwd`.

Sin embargo, una implementación NIS de estas funciones modifica este comportamiento y realiza una llamada RPC al servidor NIS, que busca el nombre de usuario o el ID de usuario. Esto ocurre transparentemente para la aplicación. La función puede tratar a los datos NIS como si hubiesen sido añadidos al fichero original `/etc/passwd` por lo que ambos juegos de información están disponibles para la aplicación, o como si lo hubiese reemplazado completamente, por lo que la información del `passwd` local es ignorada y sólo se utilizan los datos de NIS.

En las implementaciones tradicionales de NIS había ciertas convenciones sobre qué mapas eran reemplazados y cuáles se añadían a la información original. Algunos, como los mapas `passwd`, requerían de modificaciones extrañas en el fichero `passwd` que, si se hacían incorrectamente, abrían agujeros de seguridad. Para evitar estos riesgos, NYS y la `libc` de GNU utilizan un esquema de configuración general que determina si un juego particular de funciones de cliente debe utilizar los ficheros originales, NIS, o NIS+, y en qué orden. Este esquema será descrito más adelante en este capítulo.

13.4. Ejecutando un Servidor NIS

Después de tanta palabrería técnica, es hora de poner las manos en la masa con el verdadero trabajo de configuración. En esta sección cubriremos la configuración de un servidor NIS. Si ya hay un servidor NIS ejecutándose en su red, no necesitará configurarlo por usted mismo; en ese caso, puede saltarse esta sección sin problema.

Tenga en cuenta que si sólo quiere experimentar con el servidor, asegúrese de que no le asigna un nombre de dominio NIS que ya esté en uso en su red. Esto puede desbaratar todo el servicio de red y provocar infelicidad y enfado a mucha gente.

Existen dos configuraciones posibles del servidor NIS: maestra y esclava. La configuración esclava es una máquina que proporciona una copia de seguridad, por si el servidor maestro falla. Aquí sólo cubriremos la configuración de un servidor maestro. La documentación del servidor explica las diferencias, por si quiere configurar un servidor esclavo.

Actualmente existen dos servidores NIS disponibles para Linux: uno contenido en el paquete `yps` de Tobias Reber, y otro en el paquete `ypserv` de Peter Eriksson. No importa cuál ejecute.

Después de instalar el programa (**ypserv**) en `/usr/sbin`, debe crear el directorio que contendrá los ficheros de mapas que va a distribuir su servidor. Al configurar un dominio NIS para el dominio cervecera, los mapas irían en `/var/yp/cervecera`. El servidor determina si está sirviendo un dominio NIS particular comprobando si existe el directorio de los mapas. Si quiere deshabilitar el servicio para algún dominio NIS, asegúrese de eliminar el directorio.

Normalmente los mapas se almacenan en ficheros DBM para agilizar las búsquedas. Se crean a partir de los ficheros maestro utilizando un programa llamado **makedbm** (del servidor de Tobias) o **dbmload** (del servidor de Peter).

Transformar un fichero maestro en una forma que **dbmload** pueda entender requiere normalmente de algo de magia **awk** o **sed**, que tiende a ser algo aburrido de escribir y de recordar. Es por esto que el paquete `ypserv` de Peter Eriksson contiene un Makefile (llamado `ypMakefile`) que se encarga por usted de la conversión de la mayoría de los ficheros maestros. Debe instalarlo como `Makefile` en su directorio de mapas y editarlo para reflejar los mapas que quiere que el servidor NIS comparta. Al principio del fichero encontrará el objetivo all que lista los servicios que ofrece **ypserv**. Por defecto la línea se parecerá a esto:

```
    all: ethers hosts networks protocols rpc services passwd group netid
```

Si no quiere producir, por ejemplo, los mapas `ethers.byname` y `ethers.byaddr`, simplemente borre el prerrequisito ethers de esta regla. Para comprobar su configuración, puede empezar con sólo uno o dos mapas, como los mapas `services.*`.

Después de editar el `Makefile`, estando en el directorio de mapas, teclee **`make`**. Esto generará automáticamente los mapas y los instalará. Debe asegurarse de actualizar los mapas cada vez que cambie los ficheros maestro, o de otra manera los cambios permanecerán invisibles a la red.

La sección Configurando un Cliente NIS con la libc de GNU explicará cómo configurar el código NIS del cliente. Si su configuración no funciona, trate de averiguar si las peticiones llegan a su servidor. Si especifica la opción de línea de órdenes `debug` al ejecutar **ypserv**, imprimirá mensajes de depuración a la consola acerca de todas las consultas NIS que lleguen y de los resultados devueltos. Esto debería darle una pista acerca del origen del problema. El servidor de Tobias no tiene esta opción.

13.5. Seguridad en el Servidor NIS

NIS solía tener un defecto grave de seguridad: dejaba su fichero de contraseñas legible por prácticamente cualquier persona en toda Internet, lo que suponía un gran número de posibles intrusos. Si un intruso sabía su (de usted) dominio NIS y la dirección de su servidor, simplemente tenía que enviar una consulta al mapa `passwd.byname` y recibir al instante todas las contraseñas encriptadas del sistema. Con un programa rápido para *crackear* contraseñas como el **crack**, y un buen diccionario, averiguar unas cuantas contraseñas de usuario no es problema.

De todo esto trata la opción *securenets*. Esta opción simplemente restringe el acceso a su servidor NIS a ciertos nodos, basándose en su dirección IP o números de red. La última versión de **ypserv** implementa esta característica de dos maneras. La primera consta de un fichero de configuración especial llamado `/etc/ypserv.securenets` y la segunda utiliza convenientemente los ficheros `/etc/hosts.allow` y `/etc/hosts.deny` que ya nos encontramos en Capítulo 12.[137] Así, para restringir el acceso a los nodos de dentro de la Cervecera, su administrador de red añadiría esta línea al `hosts.allow`:

```
    ypserv: 172.16.2.
```

Esto permitiría a todos los nodos de la red 172.16.2.0 acceder al servidor NIS. Para denegar el acceso al resto de nodos, la correspondiente línea en el `hosts.deny` sería:

```
    ypserv: ALL
```

Las direcciones IP no son la única manera de especificar nodos y redes en `hosts.allow` y `hosts.deny`. Por favor, consulte la página del manual `hosts_access(5)` de su sistema para más detalles. Sin embargo, advierta que *no puede* utilizar nombres de nodo o de dominio en la entrada ypserv. Si especifica un nombre de nodo, el servidor tratará de resolver este nombre de nodopero el resolvedor a su vez llamará a **ypserv**, y caerá en un bucle infinito.

Para configurar la seguridad securenets utilizando el método `/etc/ypserv.securenets`, necesita crear el fichero de configuración, `/etc/ypserv.securenets`. Este fichero de configuración es simple en su estructura. Cada línea describe un nodo o red de nodos que tendrán permiso de acceso al servidor. Cualquier dirección no descrita con una entrada en este fichero tendrá denegado el acceso. Una línea que comience por # será tratada como comentario. El ejemplo 13−1 muestra cómo sería un sencillo fichero `/etc/ypserv.securenets`:

Ejemplo 13−1. Fichero ypserv.securenets de Ejemplo

```
# permitir conexiones desde el nodo local -- necesario
host 127.0.0.1
# lo mismo para 255.255.255.255 127.0.0.1
#
# permitir conexiones desde cualquier nodo de la red de la Cervecera Virtual
255.255.255.0   172.16.1.0
#
```

La primera entrada de cada línea es la máscara de red a utilizar, siendo host una palabra clave especial que significa máscara de red 255.255.255.255. La segunda entrada de cada línea es la dirección IP a la que aplicar la máscara de red.

Una tercera opción es utilizar el mapeador de puertos (portmapper) seguro en lugar de la opción `securenets` de **ypserv**. El mapeador de puertos seguro (`portmap−5.0`) utiliza también el esquema de `hosts.allow`, pero ofrece esto a todos los servidores RPC, no sólo a **ypserv**. [138] Sin embargo, no se debe utilizar la opción `securenets` y el mapeador de puertos seguro al mismo tiempo, por la sobrecarga que esto supondría.

13.6. Configurando un Cliente NIS con la libc de GNU

Ahora describiremos y discutiremos la configuración de un cliente NIS utilizando el soporte de la biblioteca libc de GNU.

Su primer paso debe ser decirle al cliente NIS de la libc de GNU qué servidor usar para el servicio NIS. Anteriormente mencionamos que el **ypbind** de Linux permite configurar el servidor NIS a utilizar. El comportamiento predeterminado es consultar al servidor de la red local. Si es probable que el nodo que está configurando se vaya a mover de un dominio a otro, como un portátil, debería dejar el fichero `/etc/yp.conf` vacío, y el nodo consultará en la red local qué servidor NIS es el que procede.

Una configuración más segura para la mayoría de nodos es especificar el nombre del servidor en el fichero de configuración `/etc/yp.conf`. Un fichero muy sencillo para un nodo de la red de la Vinatera sería así:

```
# yp.conf - configuración de YP para la biblioteca GNU libc.
#
ypserver vbardolino
```

La sentencia ypserver le dice a su nodo que use el nodo especificado como servidor NIS para el dominio local. En este ejemplo hemos especificado vbardolino como servidor NIS. Por supuesto, la dirección IP correspondiente a vbardolino debe especificarse en el fichero `hosts`; alternativamente, puede usar la propia dirección IP con el argumento server.

En la forma que se muestra en el ejemplo, la orden **ypserver** le dice a **ypbind** que use el servidor nombrado sin tener en cuenta cuál es el dominio NIS actual. Sin embargo, si usted quiere mover su máquina frecuentemente por varios dominios NIS, querrá tener la información de varios dominios en el fichero yp.conf. Puede tener información de los servidores de varios dominios NIS en yp.conf especificando la información mediante la sentencia domain. Por ejemplo, puede cambiar el ejemplo anterior en un portátil por esto:

```
# yp.conf - configuración de YP para la biblioteca libc de GNU.
#
domain vinatera server vbardolino
domain cervecera server vstout
```

Esto le permite levantar el portátil en cualquiera de los dos dominios simplemente especificando el dominio NIS deseado en tiempo de ejecución utilizando la orden **domainname**. Luego el cliente NIS utilizará el servidor que proceda para el dominio actual.

Hay una tercera opción que puede querer usar. Cubre el caso en el que usted no sabe el nombre o la dirección IP del servidor a utilizar en un dominio particular, pero quiere usar servidores fijos para ciertos dominios. Imagine que queremos insistir en utilizar un servidor especificado cuando trabajamos dentro del dominio de la Vinatera, pero queremos buscar un servidor cuando estamos dentro del dominio de la Cervecera. Tendríamos que modificar nuestro fichero yp.conf de nuevo para que quedara así:

```
# yp.conf - configuración YP para la biblioteca libc de GNU.
#
domain vinatera server vbardolino
domain cervecera broadcast
```

La palabra clave broadcast le dice a **ypbind** que use el servidor NIS que encuentre en el dominio.

Tras crear este fichero básico de configuración y asegurarse de que es legible por todo el mundo, debe realizar la primera comprobación para conectar con su servidor. Asegúrese de elegir un mapa que su servidor distribuya, como el hosts.byname, e intente obtenerlo utilizando la utilidad **ypcat**:

```
# ypcat hosts.byname
172.16.2.2      vbeaujolais.vbrew.com    vbeaujolais
172.16.2.3      vbardolino.vbrew.com     vbardolino
172.16.1.1      vlager.vbrew.com         vlager
172.16.2.1      vlager.vbrew.com         vlager
172.16.1.2      vstout.vbrew.com         vstout
172.16.1.3      vale.vbrew.com           vale
172.16.2.4      vchianti.vbrew.com       vchianti
```

La salida que obtenga debe parecerse a la que se muestra arriba. Si obtiene un mensaje de error que diga: Can't bind to server which serves domain, entonces o el dominio NIS que ha especificado no tiene un servidor concordante definido en yp.conf, o el servidor es inaccesible por alguna razón. En el último caso, asegúrese de que un **ping** al nodo arroja un resultado positivo, y de que de hecho está corriendo un servidor NIS. Puede verificar esto último utilizando la orden **rpcinfo**, que tendría que producir la siguiente salida:

```
# rpcinfo -u serverhost ypserv
program 100004 version 1 ready and waiting
program 100004 version 2 ready and waiting
```

13.7. Escogiendo los Mapas Correctos

Habiéndose asegurado de que puede acceder al servidor NIS, debe decidir qué ficheros de configuración reemplazar o aumentar con los mapas NIS. Normalmente querrá usar mapas NIS para las funciones de búsqueda de nodo y de contraseña. La primera es especialmente útil si carece de servicio BIND. La búsqueda de contraseña permite a todos los usuarios ingresar en sus cuentas desde cualquier sistema del dominio NIS; normalmente esto implica compartir un directorio /home central entre todos los nodos vía NFS. El mapa de contraseñas se explica con detalle en la siguiente sección.

Otros mapas, como el `services.byname`, no proporcionan ganancias tan dramáticas, pero sí le ahorran algo de trabajo de edición. El mapa `services.byname` cobra valor si instala alguna aplicación de red que utilice un servicio que no esté en el fichero estándar `services`.

Generalmente querrá tener donde elegir cuando una función de búsqueda utilice los ficheros locales, cuando consulte a un servidor NIS y cuando utilice otros servidores como el DNS. La libc de GNU le permite configurar el orden en el que una función accede a estos servicios. Esto se controla a través del fichero `/etc/nsswitch.conf`, que quiere decir *Conmutador de Servicio de Nombres* (Name Service Switch), aunque por supuesto no está limitado al servicio de nombres. El fichero contiene una línea nombrando el servicio a usar para cada una de las funciones de búsqueda de datos soportadas por la libc de GNU.

El orden correcto de los servicios depende del tipo de datos que ofrece cada servicio. No es probable que el mapa `services.byname` contenga entradas diferentes a las del fichero local `services`; sólo contendrá entradas adicionales. Por tanto, parece razonable consultar primero a los ficheros locales y usar NIS sólo si el nombre del servicio no se encuentra. Por otra parte, la información del nombre de nodo puede cambiar frecuentemente, por lo que el servidor DNS o NIS siempre debe tener la información más precisa posible, mientras que el fichero local `hosts` sólo se conserva como copia de seguridad por si DNS o NIS fallan. Por tanto, para los nombres de nodo, normalmente querrá que el fichero local se lea en último lugar.

El siguiente ejemplo muestra cómo forzar a `gethostbyname` y `gethostbyaddr` que busquen en NIS y DNS antes de buscar en el fichero `hosts`, y cómo hacer que la función `getservbyname` busque en los ficheros locales antes de consultar a NIS. Estas funciones de resolución probarán con cada uno de los servicios listados en orden; si una búsqueda tiene éxito, se devuelve el resultado; si no, probarán con el siguiente servicio de la lista. La configuración para estas prioridades es:

```
# pequeño ejemplo de /etc/nsswitch.conf
#
hosts:     nis dns files
services:  files nis
```

Lo siguiente es una lista completa de los servicios y lugares que pueden utilizarse en una entrada del fichero `nsswitch.conf`. Los verdaderos mapas, ficheros, servidores y objetos consultados dependen del nombre de la entrada. Lo siguiente puede aparecer a la derecha de los dos puntos:

nis
: Utilizar el servidor NIS del dominio actual. La situación del servidor consultado se configura en el fichero `yp.conf`, como se muestra en la sección anterior. Para la entrada hosts, se consultan los mapas `hosts.byname` y `hosts.byaddr`.

nisplus o nis+
: Utilizar el servidor NIS+ de este dominio. La situación del servidor se obtiene a partir del fichero `/etc/nis.conf`.

dns
: Utilizar el servidor de nombres DNS. Este tipo de servicio sólo es útil con la entrada hosts. Los servidores de nombres consultados todavía están determinados por el fichero estándar `resolv.conf`.

files
: Utilizar el fichero local, como el fichero `/etc/hosts` para la entrada hosts.

compat
: Ser compatible con formatos de fichero antiguos. Esta opción puede utilizarse cuando se use NYS o la glibc 2.x para hacer búsquedas NIS o NIS+. Como estas versiones normalmente no saben interpretar las entradas antiguas de NIS en los ficheros `passwd` y `group`, la opción compat les permite funcionar con esos formatos.

db

Buscar la información en los ficheros DBM situados en el directorio `/var/db`. Para ese fichero se utiliza el nombre de mapa NIS correspondiente.

Actualmente, el soporte de NIS de la libc de GNU proporciona las siguientes bases de datos de `nsswitch.conf`: aliases, ethers.group, hosts, netgroup, network, passwd, protocols, publickey, rpc, services, y shadow. Es probable que se añandan más entradas.

El Ejemplo 13−2 muestra un ejemplo más completo que introduce otra característica del fichero `nsswitch.conf`. La palabra clave [NOTFOUND=return] especificada en la entrada hosts le dice al cliente NIS que devuelva si el elemento deseado no pudo encontrarse en la base de datos de NIS o DNS. Esto es, el cliente NIS continuará buscando en los ficheros locales *sólo* si las llamadas a los servidores NIS y DNS fallan por alguna razón. Por tanto, cuando el servidor NIS no responda se utilizarán los ficheros locales sólo en tiempo de ejecución y como copia de seguridad.

Ejemplo 13−2. Fichero nsswitch.conf de Ejemplo

```
# /etc/nsswitch.conf
#
hosts:      nis dns [NOTFOUND=return] files
networks:   nis [NOTFOUND=return] files
services:   files nis
protocols:  files nis
rpc:        files nis
```

La libc de GNU proporciona otras acciones descritas en la página del manual `nsswitch`.

13.8. Utilizando los Mapas passwd y group

Una de las aplicaciones más importantes de NIS es sincronizar la información del usuario y de su cuenta en todos los nodos de un dominio NIS. Por consiguiente, normalmente usted sólo mantendrá un fichero `/etc/passwd` pequeño, al cual se añade la información global de los mapas NIS. Sin embargo, no es suficiente con habilitar las búsquedas NIS para este servicio en el fichero `nsswitch.conf`.

Antes de fiarse de la información de contraseñas distribuida por NIS, debe asegurarse de que todos los números ID de usuario que haya en el fichero local `passwd` concuerdan con los del servidor NIS. La consistencia de los IDs también es importante para otros propósitos, como montar particiones NFS desde otros nodos de su red.

Si alguno de los IDs numéricos de `/etc/passwd` o `group` difiere de los de los mapas, debe ajustar el dueño de todos los ficheros que pertenezcan a ese usuario. Primero, debe cambiar todos los uids y gids de `passwd` y `group` a los nuevos valores, luego mirar que todos los ficheros que pertenecen a los usuarios han cambiado, y cambiar su dueño. Suponga que news tenía un ID de usuario de 9 y que okir tenía un ID de usuario de 103, y que fueron cambiados a otro valor; luego puede ejecutar las siguientes órdenes como root:

```
# find / −uid   9 −print >/tmp/uid.9
# find / −uid 103 −print >/tmp/uid.103
# cat /tmp/uid.9   | xargs chown news
# cat /tmp/uid.103 | xargs chown okir
```

Es importante que ejecute estas ordenes con el nuevo fichero `passwd` instalado, y que reúna todos los ficheros antes de cambiar el dueño de alguno de ellos. Para actualizar los grupos dueños de los ficheros, utilice un método similar con gid en vez de uid y chgrp en vez de chown.

Una vez que haya hecho esto, los uids y gids de su sistema concordarán con los de todos los nodos de su dominio NIS. El siguiente paso será añadir a `nsswitch.conf` las líneas de configuración que habilitan la búsqueda NIS de la información de usuario y grupo:

```
    # /etc/nsswitch.conf - tratamiento de passwd y group
    passwd: nis files
    group:  nis files
```

Esto afecta a qué lugar buscarán la información de usuario la orden **login** y todos sus amigos. Cuando un usuario intente registrarse en el sistema, **login** consultará primero los mapas NIS, y si esta búsqueda falla, recurrirá a los ficheros locales. Normalmente, usted eliminará a casi todos los usuarios de sus ficheros locales, y sólo dejará las entradas de root y otras cuentas genéricas como mail. Eso es porque algunas tareas vitales del sistema pueden requerir resolver uids a partir de nombres de usuario o viceversa. Por ejemplo, los trabajos administrativos de **cron** pueden utilizar la orden **su** para convertirse temporalmente en el usuario news, o el subsistema UUCP puede tener que enviar un informe de estado por correo. Si news y uucp no tienen entradas en el fichero local `passwd`, estos trabajos fallarán estrepitosamente durante una caída del servicio NIS.

Finalmente, si usted está usando la implementación de NIS antigua (soportada por el modo compat para los ficheros `passwd` y `group` en las implementaciones de NYS o glibc), debe insertar las pesadas entradas especiales dentro de ellos. Estas entradas determinan dónde se insertarán los registros derivados NIS dentro de la base de datos. Las entradas pueden añadirse en cualquier lugar, pero normalmente se añaden al final. Las entradas que hay que añadir en el fichero `/etc/passwd` son:

```
+::::::
```

y las del fichero `/etc/groups`:

```
+:::
```

Con la glibc 2.x y NYS se puede ignorar los parámetros del registro de un usuario recibido desde el servidor NIS creando entradas con un + antes del nombre de ingreso, y excluir usuarios específicos creando entradas con un – antes del nombre de ingreso. Por ejemplo, las entradas:

```
+stuart::::::/bin/jacl
    -jedd::::::
```

ignorarían el intérprete de órdenes proporcionado por el servidor NIS para el usuario stuart, y no permitirían al usuario jedd ingresar en esta máquina. En los campos que se dejan vacíos se utiliza la información proporcionada por el servidor NIS.

Existen dos inconvenientes aquí. Primero, la configuración descrita arriba sólo funciona en los sistemas de ingreso que no utilizan contraseñas ocultas (shadow passwords). Los misterios de usar contraseñas ocultas con NIS se discutirán en la siguiente sección. Segundo, las órdenes de ingreso no son las únicos que acceden al fichero `passwd`fíjese en la orden **ls**, que casi todo el mundo utiliza constantemente. Al hacer listados largos, el comando **ls** muestra los nombres simbólicos de los usuarios y los grupos dueños de un archivo; esto es, para cada uid y gid que se encuentra, tiene que consultar al servidor NIS. Una consulta NIS tarda algo más que la búsqueda equivalente en un fichero local. Puede encontrarse con que al compartir la información de `passwd` y `group` mediante NIS se produce una reducción significativa del rendimiento de algunos programas que utilizan esta información de manera frecuente.

Y esto no es toda la historia. Imagine qué ocurriría si una usuaria quiere cambiar su contraseña. Normalmente invocará la orden **passwd**, que lee la nueva contraseña y actualiza el fichero local `passwd`. Esto es imposible con NIS, ya que ese fichero ya no está disponible localmente, pero hacer que los usuarios tengan que ingresar en el servidor NIS cada vez que quieran cambiar su contraseña tampoco es una opción. Es por esto que NIS proporciona un sustituto para **passwd** llamado **ypasswd**, que maneja los cambios de contraseña bajo NIS. Para cambiar la contraseña en el nodo servidor, contacta con el demonio **yppasswdd** de ese nodo mediante RPC, y le proporciona la información de la contraseña actualizada. Normalmente se instala **yppasswd** sobre el programa normal haciendo algo así:

```
    # cd /bin
    # mv passwd passwd.old
    # ln yppasswd passwd
```

Al mismo tiempo, debe instalar **rpc.yppasswdd** en el servidor y lanzarlo desde un script de red. Esto ocultará de manera efectiva las viscisitudes de NIS a sus usuarios.

13.9. Usando NIS con Soporte de Contraseñas Ocultas

Usar NIS en conjunción con contraseñas ocultas es algo problemático. Antes que nada tenemos malas noticias: usar NIS frustra los objetivos de las contraseñas ocultas. El esquema de `shadow` fue diseñado para evitar que los usuarios que no fuesen root tuvieran acceso a la forma encriptada de las contraseñas de ingreso. Usar NIS para compartir los datos de `shadow` necesariamente hace disponibles las contraseñas encriptadas a todo usuario de la red que pueda escuchar las respuestas del servidor NIS. Una política que fuerce a los usuarios a elegir buenas contraseñas es razonablemente mejor que intentar usar contraseñas ocultas en un entorno NIS. Veamos cómo se haría, en el caso de que decida seguir adelante.

En la libc5 no existe una solución real para compartir los datos de `shadow` con NIS. La única manera de distribuir las contraseñas y la información de usuario con NIS es a través de los mapas `passwd.*` estándar. Si usted tiene instaladas las contraseñas ocultas, la manera más sencilla de compartirlas es generar un fichero `passwd` adecuado a partir de `/etc/shadow` utilizando herramientas como `pwuncov`, y crear los mapas NIS a partir de ese fichero.

Por supuesto, existen soluciones chapuceras para usar NIS y contraseñas ocultas al mismo tiempo, por ejemplo, instalando un fichero `/etc/shadow` en cada nodo de la red, mientras la información de usuario se distribuye mediante NIS. Sin embargo, esta solución es realmente bruta y se opone a los objetivos de NIS, que son los de facilitar la administración del sistema.

El soporte de NIS de la biblioteca libc de GNU (libc6) proporciona soporte para bases de datos de contraseñas ocultas. No proporciona una solución real al problema de hacer accesibles las contraseñas, pero simplifica el mantenimiento de las contraseñas en entornos en los que usted quiere usar NIS con contraseñas ocultas. Para usarlo, debe crear una base de datos `shadow.byname` y añadir la siguiente línea a su `/etc/nsswitch.conf`:

```
# Soporte para contraseñas ocultas
    shadow:          compat
```

Si utiliza contraseñas ocultas con NIS, debe tratar de mantener alguna seguridad restringiendo el acceso a su base de datos de NIS. Vea Sección 13.5 anteriormente en este capítulo.

Capítulo 14. El Sistema de Ficherosde Red

El Sistema de Ficheros de Red (NFS, por sus siglas en inglés) es probablemente el servicio de red más prominente que usa RPC. Permite acceder a ficheros en anfitriones remotos exactamente en la misma manera que se accedería si fueran locales. Una mezcla de soporte en el núcleo y demonios en espacio de usuario en el lado del cliente, junto con un servidor NFS en el lado del servidor, hace esto posible. Este acceso a los ficheros es completamente transparente al cliente y funciona con varias clases de servidores y arquitecturas anfitrionas.

NFS ofrece varias características útiles:

- Los datos accedidos por todos los usuarios pueden mantenerse en un anfitrión central, con los clientes montando este directorio en tiempo de arranque. Por ejemplo, se puede mantener todas la cuentas de usuario en un anfitrión y hacer que todos los anfitriones de la red monten el directorio `/home` desde ese anfitrión. Si se instala NFS junto a NIS, los usuarios pueden entrar en cualquier sistema y trabajar en un conjunto de ficheros.

- La información que consume grandes cantidades de disco puede mantenerse en un único anfitrión. Por ejemplo, todos los ficheros y programas relativos a LaTeX y METAFONT pueden almacenarse y mantenerse en un lugar...

- Los datos administrativos pueden almacenarse en un único anfitrión. No hay necesidad de usar **rcp** para instalar el mismo fichero estúpido en 20 máquinas diferentes.

No es demasiado difícil preparar el funcionamiento básico de NFS en el cliente y el servidor; este capítulo le dice cómo.

Linux NFS es principalmente obra de Rick Sladkey, quien escribió el código del núcleo de NFS y gran parte del servidor de NFS.[139] Lo último se deriva del servidor NFS de espacio de usuario **unfsd** , originalmente escrito por Mark Shand, y el servidor NFS Harris **hnfs**, escrito por Donald Becker.

Echemos un vistazo a cómo funciona NFS. Primero, un cliente intenta montar un directorio de un anfitrión remoto en un directorio local justo de la misma manera que si fuese un dispositivo físico. Sin embargo, la sintaxis usada para especificar el directorio remoto es diferente. Por ejemplo, para montar `/home` desde el anfitrión vlager en `/users` en vale, el administrador escribe la siguiente orden en vale:[140]

```
# mount -t nfs vlager:/home /users
```

mount tratará de conectar con el demonio remoto sobre **rpc.mountd** de vlager vía RPC. El servidor verificará si vale tiene permiso para montar el directorio en cuestión, en cuyo caso, devuelve un descriptor de fichero. Este descriptor será usado en todas las peticiones subsecuentes que se hagan sobre los ficheros bajo `/users`.

Cuando alguien accede a un fichero sobre NFS, el núcleo manda una llamada de RPC a **rpc.nfsd** (el demonio de NFS) en la máquina servidor. Esta llamada toma el descriptor de fichero, el nombre del fichero a acceder y los identificadores de usuario y grupo del usuario como parámetros. Éstos se usan en la determinación de los derechos de acceso al fichero especificado. Para prevenir que usuarios no autorizados lean o modifiquen ficheros, los identificadores de usuario y grupo deben ser iguales en ambos anfitriones...

En la mayoría de las implementaciones de Unix, la funcionalidad NFS de cliente y servidor se implementan como demonios a nivel de núcleo que arrancan desde el espacio de usuario al arrancar la máquina. Éstos son los *Demonios NFS* (**rpc.nfsd**) en el anfitrión servidor, y *Block I/O Daemon* (**biod**) en el anfitrión cliente. Para mejorar el rendimiento, **biod** realiza la E/S usando prelectura y postescritura asíncrona; también, varios demonios **rpc.nfsd** usualmente se ejecutan concurrentemente.

La implementación actual de NFS de Linux es un poco diferente del NFS clásico en la que el código de servidor se ejecuta enteramente en espacio de usuario, así que ejecutar múltiples copias simultáneamente es más complicado. La implementación actual de**rpc.nfsd** ofrece una característica experimental que permite soporte limitado para múltiples servidores. Olaf Kirch desarrolló el soporte para servidor NFS basado en el núcleo ofrecido en la versión 2.2 del núcleo de Linux. Su desempeño es significativamente mejor que la de la implementación en el entorno de usuario existente. Lo describiremos más adelante en este capítulo.

14.1. Preparando NFS

Antes que pueda usar NFS, sea como servidor o cliente, debe asegurarse de que su núcleo tenga incluido el soporte de NFS compilado. Los núcleos más nuevos tienen una interfaz simple en el sistema de ficheros `proc` para esto, el fichero `/proc/filesystems`, el cual puede visualizar usando la orden **cat**:

```
$ cat /proc/filesystems
    minix
    ext2
    msdos
nodev       proc
nodev       nfs
```

Si falta la palabra nfs en esta lista, tendrá que compilar su propio núcleo con NFS habilitado, o quizás necesitará cargar el módulo del núcleo si su soporte de NFS fue compilado como un módulo. Las opciones de configuración en red del núcleo se explican en la sección Configuración del Núcleo Capítulo 3.

14.2. Montando un Volumen NFS

El montaje de volúmenes NFS se parece mucho al de los sistemas de ficheros comunes. Invoque **mount** usando la siguiente sintaxis:[141]

```
# mount -t nfs nfs_volume local_dir options
```

`nfs_volume` se da como *`remote_host`*:*`remote_dir`*. Dado que esta notación es sólo para NFS, se puede omitir la opción `t nfs`.

Hay varias opciones adicionales que se puede especificar para la orden **mount** al montar un volumen de NFS. Éstas pueden ser dadas siguiendo al modificador `o` en la línea de órdenes o en el campo de opciones de entrada para el volumen en el fichero `/etc/fstab`. En ambos casos las múltiples opciones son separadas por comas y no pueden contener espacios en blanco. Las opciones especificadas en la línea de órdenes siempre tienen preferencia sobre las que estén contenidas en el fichero `fstab`.

Aquí hay un ejemplo de entrada del fichero `/etc/fstab`:

```
# volumen              punto de montaje   tipo  opciones
news:/var/spool/news  /var/spool/news     nfs   timeo=14,intr
```

Este volumen puede ser montado usando esta orden:

```
# mount news:/var/spool/news
```

En ausencia de una entrada en `fstab`, las llamadas a **mount** parecen muy feas. Por ejemplo, suponga que monta su directorio inicial de usuario desde una máquina llamada moonshot, la cual usa un tamaño de bloque de 4 K para las operaciones de lectura/escritura. tendría que incrementar el tamaño del bloque a 8 K para obtener un mejor rendimiento escribiendo la orden:

```
# mount moonshot:/home /home -o rsize=8192,wsize=8192
```

La lista de todas las opciones válidas se describe completamente en la página de ayuda `nfs(5)` del manual. La siguiente es una lista parcial de opciones que probalemente querría usar:

rsize=n y wsize=n
: Especifican el tamaño de datagrama usado por los clientes de NFS en las peticiones de lectura y escritura respectivamente. El tamaño predeterminado depende de la versión del núcleo, pero normalmente es de 1.024 bytes.

timeo=n
: Establece el tiempo (en décimas de segundo) que el cliente de NFS esperará por la respuesta a una petición. El valor predefinido es 7 (0.7 segundos). Lo que pase después depende de si usa la opción *hard* o *soft*.

hard
: Explícitamente marca este volumen como montado físicamente. Es el valor predeterminado. Esta opción hace que el servidor muestr por la consola un mensaje cuando expoira el tiempo de operación y continúa indefinidamente.

soft
: Monta lógicamente (como opuesto al montaje físico) el controlador. Esta opción causa un error de E/S a ser informado al proceso que intenta poner en funcionamiento un fichero cuando expira el tiempo de operación.

intr
: Permite una señal para interrumpir una llamada a NFS. Es útil parar abortar cuando el el servidor no responde.

Salvo para *rsize* y *wsize*, todas estas opciones se aplican a la conducta del cliente en caso de que el servidor se vuelva temporalmente inaccesible. Funcionan juntos de la manera siguiente: Cuando el cliente envía una petición al servidor NFS, espera que la operación finalice tras un intervalo dado de tiempo (especificado en la opción *timeout*). Si no se recibe confirmación dentro de este tiempo, ocurre una *expiración menor*[142], y la operación se reintenta con el tiempo de expiración duplicado. Después de un tiempo de expiración máximo de 60 segundos ocurre una *expiración mayor*[143].

Por omisión, una expiración mayor causa que el cliente envíe un mensaje a la consola y comience de nuevo, esta vez con un tiempo de expiración inicial doble que la cascada anterior. Potencialmente, esto podría continuar para siempre. Los volúmenes que obstinadamente reintentan el funcionamiento hasta que el servidor esté nuevamente disponible se llaman *de montaje fuerte*. La variedad opuesta, llamada *de montaje débil*, genera un error de E/S para el proceso que llama siempre que expire el tiempo de conexión. Debido a la post−escritura [144] introducida desde la caché del búfer, esta condición de error no se propaga hacia el proceso mismo antes de que éste llame a la función `write` la próxima vez, así que un programa nunca puede estar seguro de que una operación de escritura sobre un volumen montado débilmente se ha completado del todo.

El montar fuerte o débilmente un volumen depende parcialmente de su gusto, pero también del tipo de información al que quiere acceder desde un volumen. Por ejemplo, si monta sus programas X por NFS, ciertamente no querría que su sesión X enloquezca tan sólo porque alguien llevó la red al colapso lanzando siete copias de DOOM al mismo tiempo o desenchufando el conector Ethernet por un momento. Mediante el montaje físico del directorio que contiene estos programas, puede asegurarse que su computadora espera hasta que pueda restablecer el contacto con su servidor NFS. Por otro lado, los datos no−críticos como las particiones de las noticias montadas sobre NFS o ficheros de FTP también pueden ser montadas débilmente, así si la máquina remota está temporalmente inaccesible o apagada, no cuelga su sesión. Si su conexión de la red al servidor es dificultosa o pasa por un encaminador muy cargado, puede en cualquier caso aumentar el tiempo de expiración inicial usando la opción *timeo* o montar fuertemente los volúmenes. Los volúmenes NFS son montados fuertemente por omisión.

El montaje fuerte presenta un problema porque por omisión las operaciones con los ficheros no son interrumpibles. Así, si un proceso intenta, por ejemplo, una escritura en un servidor remoto y ese servidor es inalcanzable, la aplicación del usuario se cuelga y el usuario no puede hacer nada para abortar la operación. Si usa la opción *intr* junto con un montaje fuerte, cualquier señal recibida por el proceso interrumpe la llamada NFS para que así los usuarios puedan todavía abortar los accesos a los ficheros colgados y continuar el trabajo (aunque sin guardar el fichero).

Usualmente, el demonio **rpc.mountd** de una manera u otra rastrea qué directorios han sido montados por qué anfitrioness. Esta información puede mostrarse usando la orden **showmount** que también está incluida en el paquete del servidor NFS.

```
# showmount -e moonshot
Lista de exportación para el anfitrión local:
/home <anon clnt>

# showmount -d moonshot
Directorios en el anfitrión local:
/home

# showmount -a moonshot
Todos los puntos de montaje para el anfitrión local:
localhost:/home
```

14.3. Los Demonios NFS

Si quiere dar servicio de NFS a otros anfitriones, debe ejecutar los demonios **rpc.nfsd** y **rpc.mountd** en su máquina. Como los programas basados en RPC, no son gestionados por **inetd**, sino que son iniciados en tiempo de arranque y se registran a sí mismos con el mapeador de puertos; por consiguiente, tiene que asegurarse de arrancarlos sólo después que **rpc.portmap** se esté ejecutando. Normalmente, usaría algo como el ejemplo siguiente en uno de sus guiones de arranque de red:

```
if [ -x /usr/sbin/rpc.mountd ]; then
        /usr/sbin/rpc.mountd; echo -n " mountd"
fi
if [ -x /usr/sbin/rpc.nfsd ]; then
        /usr/sbin/rpc.nfsd; echo -n " nfsd"
fi
```

La información de propiedad de los ficheros que un demonio de NFS proporciona a sus clientes usualmente contiene sólo identificadores numéricos de usuario y de grupo. Si tanto cliente como servidor asocian los mismos nombres de usuario y grupo con esos identificadores numéricos, se dice que éstos comparten el espacio uid/gid . Por ejemplo, éste es el caso cuando usa NIS para distribuir la información `passwd` a todos los anfitriones de su red de área local.

Sin embargo, en algunas ocasiones, los IDs en los diferentes anfitriones no coinciden. En lugar actualizar los uids y gids del cliente para que coincidan con los que hay en el servidor, puede usar el demonio mapeador **rpc.ugidd** para solucionar las disparidades. Usando la opción *map_daemon* explicada poco más adelante, puede mandar a **rpc.nfsd** que mapee el espacio uid/gid del servidor al espacio uid/gid del cliente con la ayuda de **rpc.ugidd** en el cliente. Desafortunadamente el demonio **rpc.ugidd** no es suministrado con todas las distribuciones modernas de GNU/Linux, así si lo necesita y la suya no lo tiene, necesitará compilarlo a partir de las fuentes.

rpc.ugidd Es un servidor basado en RPC que se inicia desde sus guiones de arranque de la red, como **rpc.nfsd** y **rpc.mountd**:

```
if [ -x /usr/sbin/rpc.ugidd ]; then
        /usr/sbin/rpc.ugidd; echo -n " ugidd"
fi
```

14.4. El Fichero exports

Ahora veremos cómo configurar el servidor NFS. Específicamente, veremos cómo decirle al servidor NFS qué sistemas de ficheros deben ponerse disponibles para el montaje y varios parámetros que controlan el acceso que tienen los clientes al sistema de ficheros. El servidor determina el tipo de acceso que se permite a los ficheros del servidor. El fichero `/etc/exports` lista los sistemas de ficheros que el servidor permitirá a los clientes montar y usar.

Por omisión, **rpc.mountd** desaprueba el montaje de todos los directorios, lo cual es una actitud bastante sensata. Si desea permitir a uno o más anfitriones montar un directorio de NFS, debe *exportarlo* es decir, especificarlo en el fichero `exports`. Un ejemplo del fichero puede parecer como éste:

```
# exports file for vlager
/home              vale(rw) vstout(rw) vlight(rw)
/usr/X11R6         vale(ro) vstout(ro) vlight(ro)
/usr/TeX           vale(ro) vstout(ro) vlight(ro)
/                  vale(rw,no_root_squash)
/home/ftp          (ro)
```

Cada línea define un directorio y los anfitriones a los que se les permite montarlo. Un nombre de anfitrión es usualmente un nombre de dominio completamente cualificado pero puede contener adicionalmente los comodines * y ?los cuales funcionan igual que en intérprete de órdenes Bourne. Por ejemplo, `lab*.foo.com` coincide con lab01.foo.com así como laboratory.foo.com. El anfitrión puede ser especificado también usando un rango de direcciones IP en la forma *`dirección`*/*`máscara de redk`*. Si no se da un nombre de anfitrión, como con el directorio `/home/ftp` en el ejemplo previo, cualquier anfitrión coincide y se le permite montar el directorio.

Cuando se verifica a un anfitrión cliente contra el fichero `exports`, **rpx.mountd** busca el nombre del anfitrión cliente usando la llamada `gethostbyaddr`. Con DNS, esta llamada devuelve el nombre canónico del anfitrión cliente, así debe asegurarse de no usar alias en `exports`. En un entorno NIS el nombre devuelto es la primera coincidencia de la base de datos de anfitriones, y sin DNS o NIS, el nombre devuelto es el primer nombre de anfitrión encontrado en el fichero `hosts` que coincida con la dirección del cliente.

El nombre del anfitrión es seguido por una lista opcional de señales separadas por comas, encerradas entre paréntesis. Algunos de los valores que éstas señales pueden tomar son:

secure
: Esta señal insiste en requerir que se haga desde un puerto origen reservado, por ejemplo, uno que sea menor que 1.024. Esta señal está puesta por omisión.

insecure
: Esta señal revierte el efecto de la señal *secure*.

ro
: Esta señal provoca que el montaje de NFS sea para sólo lectura. Esta señal está activada por omisión.

rw
: Esta opción monta la jerarquía de ficheros en lectura−escritura.

root_squash
: Este característica de seguridad deniega a los superusuarios en los hosts especificados cualquier derecho de acceso especial mapeando las peticiones desde el uid 0 en el cliente al uid 65534 (es decir, −2) en el servidor. Este uid debe ser asociado con el usuario nobody.

no_root_squash

No mapea las peticiones desde uid 0. Esta opción está habilitada por omisión, así los superusuarios tienen acceso de supervisor a los directorios exportados de su sistema.

link_relative
: Esta opción convierte los enlaces simbólicos absolutos (donde el contenido del enlace comienza con un slash) en enlaces relativos. Esta opción sólo tiene sentido cuando está montado el sistema de ficheros entero de un anfitrión; por otra parte, algunos de los enlaces podrían apuntar a ninguna parte, o peor aún, a ficheros que nunca debieran apuntar. Esta opción está habilitada de forma predeterminada.

link_absolute
: Esta opción deja todos los enlaces simbólicos como son (la conducta normal para los servidores de NFS suministrados por Sun).

map_identity
: Esta opción le indica al servidor asumir que el cliente usa el mismo uid y gid que el servidor. Esta opción está habilitada por omisión.

map_daemon
: Esta opción indica al servidor de NFS asumir que el cliente y el servidor no comparten el mismo espacio uid/gid. **rpc.nfsd** entonces construye una lista que mapea los IDs entre cliente y servidor preguntando al demonio **rpc.ugidd** del cliente.

map_static
: Esta opción le permite especificar el nombre de un fichero que contiene un mapa estático de uids. Por ejemplo, `map_static=/etc/nfs/vlight.map` especificaría el fichero `/etc/nfs/vlight.map` como un mapa de uid/gid. La sintaxis del mapa del fichero se describe en la página 5 del manual `exports(5)`.

map_nis
: Esta opción causa que el servidor de NIS haga un mapeado de uid y gid.

anonuid y anongid
: Estas opciones le permiten especificar el uid y el gid de la cuenta anónima. Esto es útil si tiene un volumen exportado para montajes públicos.

Cualquier error que ocurra al procesar el fichero `exports` se informa al demonio **syslogd** con el nivel notice en cualquier momento en que **rpc.nfsd** o **rpc.mountd** se inicien.

Se ha de tener en cuenta que los nombres del anfitrión se obtienen a partir de la dirección IP del cliente a través de resolución inversa, por lo cual la resolución de nombres tendrá que estar configurada adecuadamente. Si usa BIND y le preocupa la seguridad, deberá activar las comprobaciones de spoofing en su fichero `host.conf`. Se hablará más profusamente sobre este tema en Capítulo 6.

14.5. Soporte para NFSv2 Basado en Núcleo

Los servidores NFS tradicionales son ejecutados en el espacio del usuario y funcionan de forma fiable, pero tienen problemas de rendimiento cuando su carga es alta. Esto es causa de la sobrecarga que añade a su funcionamiento el interfaz que ejecuta las llamadas al sistema, y porque tiene que competir con otros procesos que se encuentran en el espacio del usuario (y que son potencialmente menos importantes) para ganar tiempo de CPU.

El núcleo 2.2.0 soporta un servidor NFS experimental programado inicialmente por Olaf Kirch y después por H.J. Lu, G. Allan Morris, and Trond Myklebust. El soporte de NFS basado en núcleo proporciona un incremento significativo en el rendimiento del servidor.

En las distribuciones actuales, puede encontrar todas las herramientas del servidor NFS en un paquete. Si no, puede localizarlas en http://csua.berkeley.edu/~gam3/knfsd/. Necesita compilar un núcleo 2.2.0 con el demonio de NFS basado en el núcleo incluido si se quiere hacer uso de las herramientas. Puede comprobar si su núcleo tiene el demonio de NFS incluido comprobando si existe el fichero `/proc/sys/sunrpc/nfsd_debug`. Si inicialmente no se encuentra, puede que necesite cargar el módulo **rpc.nfsd** a través de la utilidad **modprobe**.

El demonio de NFS basado en el núcleo utiliza un fichero de configuración `/etc/exports` estándar. El paquete incluye programas que sustituyen a los demonios **rpc.mountd** y **rpc.nfsd** que además se ejecutan de una forma prácticamente igual que sus equivalentes ejecutados en el espacio de usuario.

14.6. Soporte para NFSv2 Basado en Núcleo

La versión de NFS más comúnmente utilizada ha sido la 2. La tecnología, no obstante, ha seguido avanzando comenzando a mostrar algunos puntos débiles que sólo una nueva revisión del protocolo podría solucionar. La versión 3 de NFS añade soporte para ficheros y sistemas de ficheros de tamaños superiores, mejora de forma significativa la seguridad, y ofrece diversas mejoras de rendimiento que resultarán útiles para la mayoría de los usuarios.

Olaf Kirch y Trond Myklebust están desarrollando un servidor experimental de NFSv3. Aparece en los núcleos de desarrollo 2.3 aunque también existe un parche que permite incluirlo en el código fuente del núcleo 2.2. Se compila sobre la versión 2 del demonio de NFS basado en el núcleo.

Los parches necesarios se encuentran disponibles en la página principal del servidor NFS basado en el núcleo, que se encuentra en http://csua.berkeley.edu/~gam3/knfsd/.

Capítulo 15. IPX y el Sistema de Ficheros NCP

Mucho antes de que Microsoft aprendiera sobre redes, e incluso antes de que Internet fuera conocida fuera de los círculos académicos, los entornos corporativos compartían ficheros e impresoras utilizando servidores de ficheros y de impresión basados en el sistema operativo Novell NetWare y sus protocolos asociados. [145] Muchos de estos usuarios corporativos todavía tienen redes coaxiales que utilizan estos protocolos y quieren integrar este soporte con su nuevo soporte de TCP/IP.

Linux no sólo soporta los protocolos TCP/IP, sino también el juego de protocolos utilizado por el sistema operativo NetWare de Novell Corporation. Estos protocolos son primos lejanos del TCP/IP, y aunque realizan funciones similares, difieren en muchos aspectos y desafortunadamente son incompatibles.

Linux cuenta con software tanto libre como comercial que proporciona soporte para la integración con los productos Novell.

En este capítulo se proporcionará una breve descripción de los propios protocolos, pero nos enfocaremos en cómo configurar y utilizar el software libre para permitir que Linux pueda interoperar con los productos Novell.

15.1. Xerox, Novell, e Historia

Primero veamos de dónde salieron los protocolos y cómo son. A finales de los 70, Xerox Corporation desarrolló y publicó un estándar abierto llamado Especificación de Red Xerox (Xerox Network Specification, XNS). La Especificación de Red Xerox definía una serie de protocolos designados para la interconexión de propósito general, con un gran énfasis en el uso de redes de área local. Había dos protocolos de red principales implicados: el Protocolo de Datagramas de Internet (Internet Datagram Protocol, IDP), que proporcionaba un transporte de datagramas sin conexión y no fiable de un nodo a otro, y el Protocolo de Paquetes Secuenciados (Sequenced Packet Protocol, SPP), que era una forma modificada del IDP basada en la conexión y fiable. Los datagramas de una red XNS eran direccionados individualmente. El esquema de direccionamiento utilizaba una combinación de una dirección de red IDP de 4 bytes (que era asignada unívocamente a cada segmento de la LAN Ethernet), y la dirección de nodo de 6 bytes (la dirección de la tarjeta NIC). Los enrutadores eran dispositivos que desviaban datagramas entre dos o más redes IDP separadas. En IDP no existe el concepto de subred; cualquier colección nueva de nodos requiere la asignación de otra dirección de red. Las direcciones de red se escogen de manera que sean únicas en la interred en cuestión. A veces, los administradores desarrollan convenciones haciendo que cada byte codifique algún tipo de información, como la situación geográfica, de manera que las direcciones de red se reservan de manera sistématica; sin embargo, no es un requisito del protocolo.

La Novell Corporation eligió basar su propio juego de red en el juego XNS. Novell realizó pequeñas mejoras al IDP y al SPP y los renombró como IPX (Internet Packet eXchange, Intercambio de Paquetes de Internet) y SPX (Sequenced Packet eXchange, Intercambio de Paquetes Secuenciados). Novell añadió dos protocolos nuevos: el Protocolo Central de NetWare (NetWare Core Protocol, NCP), que proporcionaba funciones para compartir ficheros e impresoras sobre IPX, y el Protocolo de Anuncio de Servicios (Service Advertisement Protocol, SAP), que permitía a los nodos de una red Novell saber qué nodos proporcionaban qué servicios.

La Tabla 15−1 relaciona los juegos de protocolos XNS, Novell y TCP/IP en términos de la función que realizan. Las relaciones son sólo una aproximación, pero pueden ayudarle a comprender qué sucede cuando nos refiramos a estos protocolos más adelante.

Tabla 15−1. Relaciones entre los Protocolos de XNS, Novell, y TCP/IP

XNS	Novell	TCP/IP	Características
IDP	IPX	UDP/IP	Transporte sin conexión ni fiabilidad
SPP	SPX	TCP	Transporte basado en la conexión y fiable
	NCP	NFS	Servicios de fichero
	RIP	RIP	Intercambio de información de encaminamiento
	SAP		Intercambio de información sobre la disponibilidad de servicios

15.2. IPX y Linux

Alan Cox fue el que desarrolló el primer soporte de IPX para el núcleo de Linux, en 1985. [146] Al principio sólo era útil para poco más que encaminar datagramas IPC. Desde entonces, otra gente, entre los que destacan Greg Page, ha proporcionado soporte adicional. [147] Greg desarrolló las utilidades de configuración que utilizaremos en este capítulo para configurar nuestras interfaces. Volker Lendecke desarrolló el soporte para el sistema de ficheros NCP, que permite a Linux montar volúmenes en servidores de ficheros NetWare conectados a una red. [148] También creó herramientas que permiten imprimir en y desde Linux. Ales Dryak y Martin Stover, cada uno de manera independiente, desarrollaron demonios de ficheros NCP para Linux, que permiten que clientes NetWare conectados a una red puedan montar directorios Linux exportados como volúmenes NCP, de la misma manera que el demonio NFS permite a Linux servir sistemas de ficheros a clientes que usen el protocolo NFS. [149] Caldera Systems, Inc. ofrece un cliente y un servidor NetWare comerciales y autorizados que soportan los últimos estándares de Novell, incluyendo el soporte para el Servicio de Directorio de NetWare (NetWare Directory Service, NDS). [150]

Por lo tanto, hoy en día Linux soporta un amplio surtido de servicios que permiten a los sistemas integrarse con las redes basadas en Novell existentes.

15.2.1. Soporte de Caldera

Aunque no detallaremos el soporte de NetWare que ofrece Caldera en este capítulo, es importante que hablemos de él. Caldera fue fundada por Ray Noorda, el antiguo presidente ejecutivo de Novell. El soporte de NetWare de Caldera es un producto comercial completamente soportado por Caldera. Caldera proporciona el soporte de NetWare como un componente de su propia distribución de Linux, llamada Caldera OpenLinux. La solución de Caldera es una manera ideal de introducir Linux en los entornos que demandan tanto un soporte comercial como la posibilidad de integrarse en redes Novell nuevas o ya existentes.

El soporte de NetWare de Caldera está completamente autorizado por Novell, y proporciona una gran certeza de que los productos de las dos compañías serán interoperativos. Las dos excepciones a esta certeza son el funcionamiento "IP puro" del cliente, y el servidor NDS, aunque ninguno de los dos estaba disponible a la hora de escribir esto. El cliente NetWare y el servidor NetWare están ambos disponibles. También se proporciona un juego de herramientas de administración que puede simplificar la administración no sólo de las máquinas NetWare basadas en Linux, sino también de las máquinas Novell NetWare, adquiriendo la potencia de los lenguajes de *scripting* de Unix. Se puede encontrar más información sobre Caldera en su página web.

15.2.2. Más sobre el soporte de NDS

Junto con la versión 4 de NetWare, Novell introdujo una funcionalidad llamada Servicio de Directorio de NetWare (NetWare Directory Service, NDS). Las especificaciones del NDS no están disponibles sin un acuerdo de no divulgación (NDA), una restricción que dificulta el desarrollo del soporte libre. Sólo la versión 2.2.0 o posteriores del paquete `ncpfs`, del que hablaremos más tarde, tiene algún soporte de NDS. Este soporte fue desarrollado aplicando ingeniería inversa al protocolo NDS. El soporte parece funcionar, pero oficialmente todavía es considerado experimental. Se pueden usar las herramientas no−NDS con los servidores NetWare 4, siempre que tengan activado el modo bindery emulation.

El software de Caldera tiene soporte completo de NDS, porque su implementación está autorizada por Novell. Sin embargo, esta implementación no es libre, por lo que no se puede acceder al código fuente ni copiar y distribuir el software libremente.

15.3. Configurando el núcleo para IPX y NCPFS

Configurar el núcleo para IPX y el sistema de ficheros NCP es simplemente cuestión de seleccionar las opciones del núcleo apropiadas a la hora de compilar el núcleo. Como muchas otras partes del núcleo, los componentes para IPX y NCPFS pueden compilarse dentro del núcleo o en forma de módulos, que luego se pueden cargar usando la orden **insmod** cuando se necesiten.

Deben seleccionarse las siguientes opciones si quiere tener soporte del protocolo IPX en Linux:

```
    General setup  --->
        [*] Networking support

    Networking options  --->
        <*> The IPX protocol

    Network device support  --->
        [*] Ethernet (10 or 100Mbit)
        ... y los controladores de dispositivo Ethernet apropiados
```

Si quiere que Linux soporte el sistema de ficheros NCP para que pueda montar volúmenes NetWare remotos, debe seleccionar adicionalmente estas opciones:

```
    Filesystems  --->
        [*] /proc filesystem support
        <*> NCP filesystem support (to mount NetWare volumes)
```

Cuando haya compilado e instalado su nuevo núcleo, estará preparado para ejecutar IPX.

15.4. Configurando las interfaces IPX

Igual que con TCP/IP, se deben configurar las interfaces IPX antes de usarlas. El protocolo IPX tiene algunos requisitos propios; consecuentemente, ha sido desarrollado un juego especial de herramientas de configuración. Utilizaremos estas herramientas para configurar nuestras interfaces IPX y las rutas.

15.4.1. Dispositivos de Red que Soportan IPX

El protocolo IPX asume que cualquier colección de nodos que puede transmitir datagramas al resto sin necesidad de rutado pertenece a la misma red IPX. De manera similar (pero menos intuitiva), dos nodos que soporten un enlace serie basado en PPP deben pertenecer a la red IPX que es el propio enlace serie. En un entorno Ethernet se puede usar un número de tipos de trama distintos para transportar datagramas IPX. Los tipos de trama más comunes que va a encontrarse son el `802.2` y `ethernet_II`. Hablaremos más sobre los tipos de trama en la siguiente sección.

Los dispositivos de red de Linux que actualmente soportan el protocolo IPX son los controladores Ethernet y PPP. La interfaz Ethernet o el PPP debe estar activo antes de que pueda ser configurado para el uso de IPX. Normalmente se configura un dispositivo Ethernet para IP e IPX, por lo que el dispositivo ya existe, pero si su red sólo es IPX, necesita usar la orden **ifconfig** para cambiar el estado del dispositivo Ethernet a lo que sigue:

```
    # ifconfig eth0 up
```

15.4.2. Herramientas de Configuración del Interfaz IPX

Greg Page ha desarrollado un juego de herramientas de configuración para las interfaces IPX, que viene como paquete precompilado en las distribuciones modernas y también puede obtenerse en forma de código fuente mediante FTP anónimo a *http://metalab.unc.edu/* en el fichero `/pub/Linux/system/filesystems/ncpfs/ipx.tgz`.

Normalmente, un script `rc` ejecuta las herramientas IPX en tiempo de arranque. Puede que su distribución haga esto por usted si ha instalado el paquete precompilado.

15.4.3. La Orden ipx_configure

Cada interfaz IPX debe saber a qué red IPX pertenece y qué tipo de trama utilizar para IPX. Todo nodo que soporte IPX tiene al menos una interfaz que el resto de la red utilizará para referirse a él, conocido como la interfaz *primaria*. El soporte de IPX del núcleo de Linux proporciona una manera de configurar automáticamente estos parámetros; la orden **ipx_configure** activa o desactiva esta capacidad de configuración automática.

Sin argumentos, la orden **ipx_configure** muestra las opciones de configuración actuales:

```
# ipx_configure
Auto Primary Select is OFF
Auto Interface Create is OFF
```

Las opciones Auto Primary y Auto Interface están apagadas por defecto. Para activarlas y permitir la configuración automática, simplemente hay que proporcionar argumentos como éstos:

```
# ipx_configure --auto_interface=on --auto_primary=on
```

Cuando el argumento `--auto_primary` se pone en `on`, el núcleo se asegurará de manera automática de que al menos una interfaz activa opera como la interfaz primaria para el nodo.

Cuando el argumento `--auto_interface` se pone en `on`, el controlador IPX del núcleo escuchará a todas las tramas recibidas en las interfaces de red activas, y tratará de determinar la dirección de la red IPX y el tipo de trama utilizado.

El mecanismo de autodetección funciona bien en las redes administradas correctamente. A veces los administradores de red toman atajos e incumplen reglas, y esto puede causarle problemas al código de autodetección de Linux. El ejemplo más común de esto es cuando una red IPX está configurada para que funcione con múltiples tipos de trama en una misma Ethernet. Esto es técnicamente una configuración inválida, ya que un nodo **802.2** no puede comunicarse directamente con un nodo Ethernet−II, y por lo tanto no pueden estar en la misma red IPX. El software de red IPX de Linux eschucha en el segmento de red datagramas IPX que se transmiten en él. A partir de éstos, trata de identificar qué direcciones de red están en uso y qué tipos de trama están asociados a cada una. Si la misma dirección de red está en uso con varios tipos de trama o en varias interfaces, el código de Linux detecta esto como una colisión de direcciones de red, y es incapaz de determinar cuál es el tipo de trama correcto. Sabrá que ocurre esto si ve mensajes en el registro de su sistema que se parezcan a esto:

```
IPX: Network number collision 0x3901ab00
eth0 etherII and eth0 802.3
```

Si observa este problema, desactive la capacidad de autodetección y configure las interfaces manualmente, utilizando la orden **ipx_interface** descrito en la siguiente sección.

15.4.4. La Orden ipx_interface

La orden **ipx_interface** se utiliza para añadir, modificar y borrar manualmente la capacidad IPX de un dispositivo de red existente. Debe utilizar **ipx_interface** cuando el método de configuración automática descrito hace un momento no le funcione, o cuando no quiera abandonar la configuración de interfaz a la suerte. **ipx_interface** le permite especificar la dirección de red IPX, el estado de la interfaz primaria, y el tipo de trama IPX que utilizará un dispositivo de red. Si está creando múltiples interfaces IPX, necesitará un **ipx_interface** para cada una.

La sintaxis de la orden para añadir IPX a un dispositivo existente es sencilla y se explica mejor con un ejemplo. Añadamos IPX a un dispositivo Ethernet existente:

```
# ipx_interface add -p eth0 etherII 0x32a10103
```

Los parámetros de turno significan:

−p
: Este parámetro especifica que esta interfaz tiene que ser una interfaz primaria. Este parámetro es opcional.

eth0
: Éste es el nombre del dispositivo de red al que estamos añadiendo soporte IPX.

etherII
: Este parámetro es el tipo de trama, en este caso Ethernet−II. Este valor también puede codificarse como `802.2`, `802.3`, o `SNAP`.

0x32a10103
: Esto es la dirección de red IPX a la que pertenece esta interfaz.

El siguiente orden elimina el soporte IPX de una interfaz:

```
# ipx_interface del eth0 etherII
```

Finalmente, para mostrar la configuración IPX actual de un dispositivo de red, utilice:

```
# ipx_interface check eth0 etherII
```

La orden **ipx_interface** está explicado con más detenimiento en su página de manual.

15.5. Configurando un Encaminador IPX

Recordará de nuestra breve descripción de los protocolos utilizados en un entorno IPX que IPX es un protocolo encaminable y que el Protocolo de Información de Encaminamiento (Routing Information Protocol, RIP) se utiliza para propagar la información de encaminamiento. La versión IPX de RIP es bastante parecida a la versión IP. Funcionan esencialmente de la misma manera; los encaminadores difunden periódicamente los contenidos de sus tablas de encaminamiento y otros encaminadores los recogen escuchando e integrando la información que reciben. Los nodos sólo necesitan saber cuál es su red local y asegurarse de enviar datagramas al resto de destinos a través de su encaminador local. El encaminador es responsable de recoger estos datagramas y redirigirlos al siguiente salto de la ruta.

En un entorno IPX, hace falta propagar por la red una segunda clase de información. El Protocolo de Anuncio de Servicio (Service Advertisement Protocol, SAP) transporta información sobre qué servicios están disponibles en qué nodos de la red. Por ejemplo, es el protocolo SAP el que permite a los usuarios obtener listas de servidores de ficheros o de impresión de la red. El protocolo SAP trabaja haciendo que los nodos que proporcionan servicios difundan periódicamente la lista de servicios que ofrecen. Los encaminadores de la red IPX recogen esta información y la propagan por toda la red junto con la información de encaminamiento de la red. Para ser un encaminador IPX compatible, hay que propagar tanto la información RIP como la SAP.

Al igual que IP, el soporte de IPX en Linux proporciona un demonio de encaminamiento llamado **ipxd** que realiza las tareas asociadas al tratamiento del encaminamiento. De nuevo, igual que en el IP, es en realidad el núcleo el que administra el redireccionamiento de los datagramas entre las interfaces de red IPX, pero lleva a cabo esto de acuerdo con un conjunto de reglas recogidas en la tabla de encaminamiento IPX. El demonio **ipxd** mantiene actualizado ese conjunto de reglas escuchando a todas las interfaces de red activas y analizando cuándo es necesario un cambio de encaminamiento. El demonio **ipxd** también responde a las peticiones de los nodos de una red conectada directamente que piden información de encaminamiento.

El programa **ipxd** está disponible preempaquetado en algunas distribuciones, y en forma de código fuente mediante FTP anónimo a *http://metalab.unc.edu/* en el fichero `/pub/Linux/system/filesystems/ncpfs/ipxripd-x.xx.tgz`.

No es necesario configurar el demonio **ipxd**. Cuando es lanzado, él automáticamente administra el encaminamiento de los dispositivos IPX que han sido configurados. La clave está en asegurarse de que todos los dispositivos IPX están configurados correctamente utilizando la orden **ipx_interfaces** antes de lanzar **ipxd**. Aunque la autodetección puede funcionar, cuando esté haciendo funciones de encaminamiento es mejor no correr riesgos, así que configure manualmente las interfaces y ahórrese problemas de encaminamiento molestos. Cada 30 segundos, **ipxd** reinspecciona todas las redes IPX enganchadas y las administra automáticamente. Esto proporciona una manera de administrar redes en interfaces que pueden no estar activas todo el tiempo, como las interfaces PPP.

Normalmente **ipxd** es lanzado en tiempo de inicio desde un script de inicio `rc` como éste:

```
# /usr/sbin/ipxd
```

No se necesita un carácter `&` porque **ipxd** se pone por defecto en segundo plano. Aunque el demonio **ipxd** es útil sobre todo en máquinas que actúan como encaminadores IPX, también es útil a los nodos en segmentos donde existen múltiples encaminadores. Cuando se especifica el argumanto `p`, **ipxd** actuará pasivamente, escuchando la información de encaminamiento del segmento y actualizando las tablas de encaminamiento, pero no transmitirá ninguna información de encaminamiento. De esta manera, un nodo puede mantener actualizadas sus tablas de encaminamiento sin tener que solicitar las rutas cada vez que quiera contactar con un nodo remoto.

15.5.1. Encaminamiento IPX Estático Utilizando la orden ipx_route

En ocasiones puede que queramos especificar a mano una ruta IPX. Igual que en IP, podemos hacer esto en IPX. la orden **ipx_route** escribe una ruta en la tabla de encaminamiento IPX sin necesidad de que tenga que haberla obtenido mediante el demonio de encaminamiento **ipxd**. La sintaxis de encaminamiento es muy sencilla (ya que IPX no soporta subredes) y se parece a esto:

```
# ipx_route add 203a41bc 31a10103 00002a02b102
```

Este orden añadiría una ruta a la red IPX remota **203a41bc** a través del encaminador de nuestra red local **31a10103** con una dirección de nodo **00002a02b102**.

Puede encontrar la dirección de nodo de un encaminador haciendo un uso juicioso dla orden **tcpdump** con el argumento `e`, para mostrar las cabeceras de la capa de enlace y observar el tráfico del encaminador. Si el encaminador es una máquina Linux, es más sencillo usar la orden **ifconfig** para mostrarla.

Puede eliminar una ruta usando la orden **ipx_route**:

```
# ipx_route del 203a41bc
```

Puede listar las rutas activas en el núcleo echándole un vistazo al fichero `/proc/net/ipx_route`. Nuestra actual tabla de encaminamiento sería así:

```
# cat ipx_route
Network    Router_Net   Router_Node
203A41BC   31A10103     00002a02b102
31A10103   Directly     Connected
```

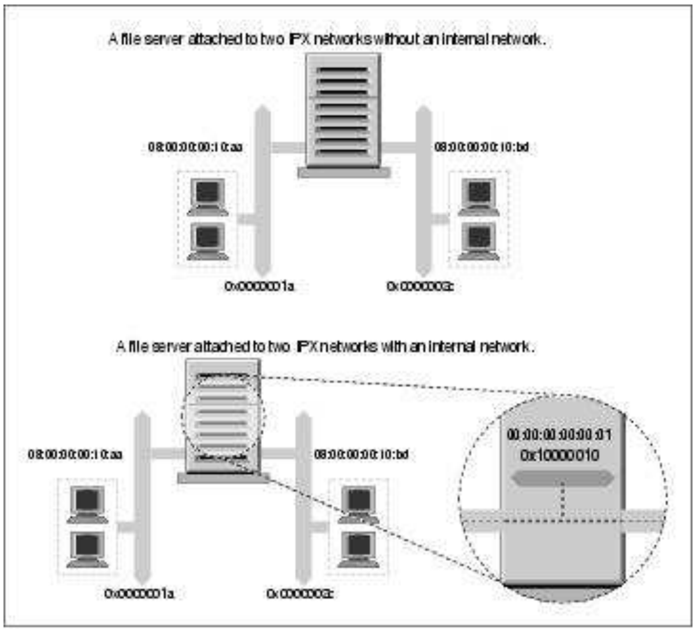
La ruta a la red **31A10103** fue creada automáticamente cuando configuramos la interfaz IPX. Cada una de nuestras redes locales estará representada por una entrada en `/proc/net/ipx_route` como ésta. Naturalmente, si nuestra máquina va a actuar como un encaminador, necesitará al menos una interfaz más.

15.5.2. Redes IPX Internas y Encaminamiento

Los nodos IPX con más de una interfaz IPX tienen una combinación de dirección de red/nodo única para cada una de sus interfaces. Para conectarse a un nodo así, se puede utilizar cualquiera de estas combinaciones de dirección de red/nodo. Cuando SAP anuncia servicios, proporciona la dirección de red/nodo asociada al servicio ofrecodo. En los nodos con múltiples interfaces, esto significa que se debe elegir una de las interfaces como la que va a propagar; ésta es la función de la bandera de interfaz primaria de la que hablamos anteriormente. Pero esto presenta un problema: la ruta a esta interfaz puede no ser siempre la más óptima, y si se da un fallo en la red que la aísle del resto de la red, el nodo quedará inaccesible aunque haya otras rutas *posibles* al resto de interfaces. Los otros nodos no conocen el resto de las rutas porque nunca son propagadas, y el núcleo no tiene manera de saber que tendría que escoger otra interfaz primaria. Para evitar este problema, ha sido desarrollado un dispositivo que permite que un nodo IPX sea conocido mediante una dirección de red/nodo individual independiente de la ruta, para los propósitos de la propagación de SAP. Esto resuelve nuestro problema, porque esta dirección de red/nodo nueva es accesible a través de todas las interfaces del nodo, y es la que SAP anuncia.

Para ilustrar el problema y su solución, Figura 15−1 muestra un servidor enganchado a dos redes IPX. La primera red no tiene red interna, pero la segunda sí. El nodo en el diagrama Figura 15−1 escogería una de sus interfaces como interfaz primaria, supongamos que la **0000001a:0800000010aa**, y es lo que sería anunciado como su punto de acceso al servicio. Esto funciona bien para los nodos de la red **0000001a**, pero significa que los usuarios de la red **0000002c** serían encaminados a través de la red para alcanzar ese puerto, a pesar de que el servidor tiene un puerto directamente en esa red, si han sabido de este servidor a partir de las difusiones de SAP.

Figura 15−1. Red IPX interna



Permitiendo a estos nodos que tengan una red virtual con direcciones de nodo virtuales, que son una construcción enteramente por software, se resuelve el problema. Esta red virtual puede imaginarse mejor como una red *dentro* del nodo IPX. Sólo necesita propagarse la información SAP para esta combinación de dirección de red/nodo virtual. A esta red virtual se la conoce como *red interna*. Pero ¿cómo saben los otros nodos cómo acceder a esta red interna? Los nodos remotos son encaminados a la red interna a través de las redes del nodo conectadas directamente. Esto significa que se verán entradas de encaminamiento que se refieren a la red interna de los nodos que soportan múltiples interfaces IPX. Esas rutas escogerán la ruta óptima disponible en el momento, y si una falla, el encaminamiento se actualiza automáticamente a la siguiente interfaz y ruta mejores. En Figura 15−1, hemos configurado una red IPX interna de dirección **0x10000010** y hemos usado una dirección de nodo **00:00:00:00:00:01**. Ésta será la dirección de nuestra interfaz primaria y la que erá anunciada via SAP. Nuestro encaminamiento reflejará que esta red es accesible a través de *cualquiera* de nuestro puertos de red reales, así que los nodos siempre usarán la mejor ruta de red para conectarse a nuestro servidor.

Para crear esta red interna, use la orden **ipx_internal_net** incluído en el paquete de herramientas IPX de Greg Page. De nuevo, un ejemplo sencillo demuestra su uso:

```
# ipx_internal_net add 10000010 000000000001
```

Este orden crearía una red IPX interna con dirección **10000010** y dirección de nodo **000000000001**. La dirección de red, como cualquier otra dirección de red IPX, debe ser única en su red. La dirección de nodo es completamente arbitraria, ya que normalmente sólo habrá un nodo en la red. Todo nodo debe tener sólo una red IPX interna, y siempre será la red primaria.

Para eliminar una red IPX interna, use:

```
# ipx_internal_net del
```

Una red IPX interna no le servirá absolutamente para nada a menos que su nodo proporcione un servicio y además tenga más de una interfaz IPX activa.

15.6. Montando un Volumen NetWare Remoto

IPX se usa comúnmente para montar volúmenes NetWare en el sistema de ficheros de Linux. Esto permite comparticiones de datos basadas en ficheros entre otros sistemas operativos y Linux. Volker Lendecke ha desarrollado el cliente NCP para Linux y un juego de herramientas asociadas que hacen posible la compartición de datos.

En un entorno NFS, hemos usado la orden **mount** de Linux para montar el sistema de ficheros remoto. Desafortunadamente, el sistema de ficheros NCP posee requisitos propios que hacen poco práctico integrarlo dentro del **mount** normal. Linux tiene un orden **ncpmount** que es el que usaremos en su lugar. El orden **ncpmount** es una de las herramientas del paquete `ncpfs` de Volker, que está disponible preempaquetado en la mayoría de las distribuciones modernas o en código fuente en el directorio `/pub/linux/misc/ncpfs/` de ftp.gwdg.de. La versión actual en el momento de escribir esto es la 2.2.0.

Antes de poder montar volúmenes NetWare, debe asegurarse de que su interfaz de red IPX está configurada correctamente (como se ha descrito anteriormente). Luego debe conocer sus detalles de ingreso (login) en el servidor NetWare que quiere montar; esto incluye la ID de usuario y la contraseña. Finalmente, necesita saber qué volumen desea montar y sobre qué directorio local quiere montarlo.

15.6.1. Un Sencillo Ejemplo de ncpmount

Un sencillo ejemplo del uso de **ncpmount**:

```
# ncpmount -S ALES_F1 -U rick -P d00-b-gud /mnt/cerveceria
```

Este orden monta todos los volúmenes del servidor de ficheros `ALES_F1` sobre el directorio `/mnt/cerveceria`, utilizando el nombre de ingreso `rick` con la contraseña `d00-b-gud`.

Normalmente, se pone el setuid dla orden **ncpmount** a root, y así puede ser utilizado por cu alquier usuario de Linux. Por defecto, ese usuario posee la conexión y sólo él o root podrá desmontarla.

NetWare incorpora la noción de *volumen*, que es análoga a un sistema de ficheros en Linux. Un volumen NetWare es la representación lógica de una sistema de ficheros NetWare, que puede ser una partición de disco individual o estar diseminada por muchas particiones. Por defecto, el soporte de NCPFS de Linux trata a los volúmenes como subdirectorios de una sistema de ficheros lógico mayor representado por todo el servidor de ficheros. la orden **ncpmount** hace que todos los volúmenes NetWare del servidor de ficheros montado aparezcan como un subdirectorio sobre el punto de montaje. Esto es conveniente si quiere acceso a todo el servidor, pero por razones técnicas complejas no podrá reexportar estos directorios usando NFS, en el caso de que desee hacerlo. En un momento discutiremos una alternativa más compleja que resuelve este problema.

15.6.2. La Orden ncpmount en Detalle

ncpmount tiene una gran número de opciones de línea de ordens que le ofrecen bastante flexibilidad a la hora de administrar sus montajes NCP. La más importante de todas se describe en la Tabla 15−2.

Tabla 15−2. Argumentos dla orden ncpmount

Argumento	Descripción
S *`server`*	El nombre del servidor de ficheros a montar.
U *`user_name`*	La ID del usuario NetWare a utilizar al ingresar en el servidor de ficheros.
P *`password`*	La contraseña a utilizar para el ingreso NetWare.
n	Se debe utilizar esta opción para los ingresos NetWare que no tienen asociados una contraseña.
C	Este argumento desactiva la conversión automática de contraseñas a mayúsculas.

c *client_name*	Esta opción le permite especificar quién posee la conexión al servidor de ficheros. Esto es útil para imprimir con NetWare, de lo que hablaremos luego con más detalle.
u *uid*	La ID de usuario de GNU/Linux que debe mostrarse como dueño de los ficheros en el directorio montado. Si no se especifica, se toma por defecto la ID de usuario que invoca a la orden **ncpmount**.
g *gid*	La ID de grupo que debe mostrarse como dueño de los ficheros del directorio montado. Si no se especifica, se toma por defecto el ID de grupo del usuario que invoca a la orden **ncpmount**.
f *file_mode*	Esta opción le permite especificar el modo de fichero (permisos) que deben tener los ficheros del directorio montado. El valor se debe especificar en octal, p. ej., `0664`. Los permisos que tendrá realmente son los permisos de fichero especificados con esta opción enmascarados con los permisos que tiene su ID de NetWare para los ficheros del servidor de ficheros. Debe poseer privilegios en el servidor y los privilegios especificados por esta opción para poder acceder a un fichero. El valor predeterminado se deriva del `umask`actual.
d *dir_mode*	Esta opción le permite especificar los permisos de directorio en el directorio montado. Se comporta de la misma manera que la opción *f*, excepto en que los permisos predeterminados se derivan del `umask` actual. Se concede el permiso de ejecución cuando se concede el acceso de lectura.
V *volume*	Esta opción le permite especificar el nombre de un volumen NetWare individual a montar bajo el punto de montaje, en lugar de montar todos los volúmenes del servidor de destino. Esta opción es necesaria si desea reexportar un volumen NetWare montado utilizando NFS.
t *time_out*	Esta opción le permite especificar el tiempo que esperará el cliente NCPFS a la respuesta de un servidor. El valor predeterminado es 60ms y el tiempo de espera se especifica en centésimas de segundo. Si experimenta algún problema de estabilidad al montar con NCP, pruebe a incrementar este valor.
r *retry_count*	El código de cliente de NCP intenta reenviar datagramas al servidor un número de veces antes de decidir que la conexión está muerta. Esta opción le permite cambiar el número de reintentos, que por omisión es 5.

15.6.3. Escondiendo Su Clave de Acceso NetWare

Es un problema de seguridad poner una clave en la línea de orden, como hicimos con la orden **ncpmount**. Otros usuarios activos y concurrentes podrían ver la clave si se les ocurre ejecutar un programa como **top** o **ps**. Para reducir el riesgo de que otros vean y roben claves de acceso NetWare, **ncpmount** es capaz de leer ciertos detalles de un fichero en el directorio raíz de un usuario. En este fichero, el usuario mantiene el nombre de acceso y la clave asociada a cada uno de los sistemas de ficheros que él o ella tiene intención de montar. El fichero se llama `~/.nwclient` y debe tener los permisos `0600` para asegurar que no puedan leerlo otros. Si los permisos no son correctos, la orden **ncpmount** rehusará utilizarlo.

El fichero tiene una sintaxis muy simple. Cualquier línea que empieze por un carácter # es considerada como un comentario y se ignora. El resto de las líneas tienen la sintaxis:

```
servidor/id clave
```

El *servidor* es el nombre del servidor de ficheros que contiene los volúmenes que se desean montar. La *id* es el nombre de acceso de su cuenta en ese servidor. El campo *clave* es opcional. Si no es proporcionado, la orden **ncpmount** le pide al usuario la clave cuando intenta montar. Si se especifica el campo *clave* con un carácter , no se utiliza ninguna clave; esto es equivalente al argumento de línea de orden `n`.

Puede proporcionar cualquier número de entradas, pero el servidor de ficheros debe ser único. La primera entrada tiene una significación especial. la orden **ncpmount** utiliza el argumento de línea de orden `S` para determinar qué entrada de `~/.nwclient` usar. Si no se especifica ningún servidor utilizando el argumento `S`, se considera la primera entrada de `~/.nwclient`, y es tratada como su servidor preferido. Debe situar el servidor de ficheros que monte más frecuentemente en la primera posición del fichero.

15.6.4. Un Ejemplo Más Complejo De ncpmount

Veamos un ejemplo más complejo de **ncpmount** que utilice unas cuantas de las características que hemos descrito. Primero, construyamos un fichero `~/.nwclient` simple:

```
# Detalles de acceso NetWare para la Cervecera y la Vinatera Virtuales
#
# Acceso a la Cervecera
ALES_F1/MATT staoic1
#
# Acceso a la Vinatera
REDS01/MATT staoic1
#
```

Asegúrese de que los permisos son correctos:

```
$ chmod 600 ~/.nwclient
```

Montemos un volumen del servidor de la Vinatera bajo un subdirectorio de un directorio compartido, especificando unos permisos de fichero y directorio tales que otros puedan compartir los datos situados en él:

```
$ ncpmount -S REDS01 -V RESEARCH -f 0664 -d 0775 /usr/share/vinatera/datos/
```

Esta orden, en combinación con el fichero `~/.nwclient` mostrado, montaría el volumen `RESEARCH` del servidor `REDS01` en el directorio `/usr/share/vinatera/datos/` utilizando la ID de acceso NetWare de `MATT` y la clave obtenida del fichero `~/.nwclient`. Los permisos de los ficheros montados son `0664` y los permisos de directorio son `0775`.

15.7. Explorando Algunas de las Otras Herramientas IPX

El paquete `ncpfs` contiene unas cuantas herramientas útiles que no hemos descrito todavía. Muchas de estas herramientas emulan a las herramientas que son proporcionadas con NetWare. En esta sección, veremos las más útiles.

15.7.1. Listado de Servidores

La orden **slist** lista todos los servidores de ficheros accesibles desde el nodo. La información es obtenida del encaminador IPX más cercano. Probablemente, esta orden estaba dirigida originalmente a permitir a los usuarios ver qué servidores de ficheros estaban disponibles para ser montados. Pero se ha hecho útil como herramienta de diagnóstico de red, permitiendo a los administradores de red ver dónde se está propagando la información SAP:

```
$ slist
NPPWR-31-CD01                              23A91330  000000000001
V242X-14-F02                               A3062DB0  000000000001
QITG_284ELI05_F4                           78A20430  000000000001
QRWMA-04-F16                               B2030D6A  000000000001
VWPDE-02-F08                               35540430  000000000001
NMCS_33PARK08_F2                           248B0530  000000000001
```

```
    NCCRD-00-CD01                              21790430  000000000001
    NWGNG-F07                                  53171D02  000000000001
    QCON_7TOMLI04_F7                           72760630  000000000001
    W639W-F04                                  D1014D0E  000000000001
    QCON_481GYM0G_F1                           77690130  000000000001
    VITG_SOE-MAIL_F4R                          33200C30  000000000001
```

slist no acepta argumentos. La salida muestra el nombre del servidor de ficheros, la dirección de red IPX, y la dirección del nodo.

15.7.2. Enviar Mensajes a Usuarios NetWare

NetWare soporta un mecanismo para enviar mensajes a usuarios que han ingresado en el sistema. La orden **nsend** implementa esta característica en GNU/Linux. Debe haber ingresado en el servidor para enviar mensajes, por lo que necesita proporcionar el nombre del servidor de ficheros y los detalles de acceso en la línea de orden junto con el usuario de destino y el mensaje a enviar:

```
    # nsend -S vbrew_f1 -U pepe -P j0yj0y supervisor
      ¡Tómate una birra conmigo antes de acabar con las colas de impresión!
```

Aquí, un usuario con nombre de acceso `pepe` le envía una tentadora invitación a la persona que usa la cuenta `supervisor` en el servidor de ficheros `ALES_F1`. Si no los proporcionamos, se utilizará nuestro servidor de ficheros y credenciales de acceso predeterminados.

15.7.3. Leyendo y Manipulando los Datos del *Bindery*

Todo servidor de ficheros NetWare mantiene una base de datos con la información sobre sus usuarios y su configuración. Esta base de datos se llama *bindery*. GNU/Linux proporciona un conjunto de herramientas que permiten leerla, y si tiene privilegios de supervisor en el servidor, cambiarla y borrarla. Hay un listado de estas herramientas en la Tabla 15−3.

Tabla 15−3. Herramientas de Manipulación de la *bindery* de GNU/Linux

Nombre de la orden	Descripción de la orden
nwfstime	Muestra o cambia la fecha y hora de un servidor NetWare
nwuserlist	Lista los usuarios conectados a un servidor NetWare
nwvolinfo	Muestra información sobre los volúmenes NetWare
nwbocreate	Crea un nuevo objeto bindery de NetWare
nwbols	Lista los objetos bindery de NetWare
nwboprops	Lista las propiedades de un objeto bindery de NetWare
nwborm	Elimina un objeto bindery de NetWare
nwbpcreate	Crea una propiedad bindery de NetWare
nwbpvalues	Imprime los contenidos de una propiedad bindery de NetWare
nwbpadd	Cambia el valor de una propiedad bindery de NetWare
nwbprm	Elimina una propiedad bindery de NetWare

15.8. Imprimiendo en una Cola de Impresión NetWare

El paquete `ncpfs` contiene una pequeña utilidad llamada **nprint** que envia trabajos de impresión a través de una conexión NCP a una cola de impresión NetWare. Esta orden crea la conexión si no existe ya, y utiliza el fichero `~/.nwclient` que describimos anteriormente para esconder el nombre de usuario y la clave de los ojos fisgones. Los argumentos de línea de orden utilizados para manejar el proceso de ingreso son los mismos que los utilizados en la orden **ncpmount**, así que no los veremos de nuevo aquí. Cubriremos las opciones de línea de orden más importantes en nuestros ejemplos; remítase a la página de manual `nprint(1)` para más detalles.

La única opción requerida en la orden **nprint** es el nombre del fichero a imprimir. Si se especifica un en el nombre de fichero o si no se especifica nada, **nprint** aceptará el trabajo de impresión desde `stdin`. Las opciones más importantes de **nprint** especifican el servidor de ficheros y la cola de impresión a los que desea enviar el trabajo. La Tabla 15−4 lista las opciones más importantes.

Tabla 15−4. Opciones de Línea de Orden de nprint

Opción	**Descripción**
−S *nombre_servidor*	El nombre del servidor de ficheros NetWare que mantiene la cola de impresión en la que se desea imprimir. Normalmente, es conveniente que el servidor tenga una entrada en el fichero `~/.nwclient`. Esta opción es obligatoria.
−q *nombre_cola*	La cola de impresión a la que enviar el trabajo de impresión. Esta opción es obligatoria.
−d *descripción_trabajo*	Texto que aparecerá en la consola de impresión al mostrar la lista de los trabajos en cola.
−l *líneas*	El número de líneas por página imprimida. Por defecto es 66.
−r *columnas*	El número de columnas por página imprimida. Por defecto es 80.
−c *copias*	El número de copias del trabajo que se imprimirán. Por defecto es 1.

Un ejemplo de **nprint** sería:

```
    $ nprint −S REDS01 −q PSLASER −c 2 /home/matt/ethylene.ps
```

Este orden imprimiría dos copias del fichero `/home/matt/ethylene.ps` a la impresora llamada `PLASER` en el servidor de ficheros `REDS01`, utilizando el nombre de usuario y la clave obtenidas del fichero `~/.nwclient`.

15.8.1. Utilizando nprint con el Demonio de Impresión en Línea

Recordará que ya hemos mencionado que la opción `c` de la orden **ncpmount** es útil para imprimir. Al final explicaremos porqué y cómo.

GNU/Linux utiliza el software de impresión en línea estilo BSD. El demonio de impresión en línea (**lpd**) es un demonio que mira en un directorio de cola (spool) local si hay trabajos en cola que tienen que imprimirse. **lpd** lee el nombre de la impresora y otros parámetros a partir del formato especial del fichero de la cola, y escribe los datos en la impresora, pasando opcionalmente los datos a través de un filtro para transformarlos o manipularlos de alguna manera.

El demonio **lpd** utiliza una simple base de datos llamada `/etc/printcap` para almacenar la configuración de la impresora, incluyendo qué filtros hay que ejecutar. Normalmente, **lpd** se ejecuta con los permisos de un usuario de sistema especial llamado lp.

Se puede configurar **nprint** como un filtro para **lpd**, lo que permite a los usuarios de su máquina GNU/Linux imprimir directamente en impresoras remotas alojadas en un servidor de ficheros NetWare. Para esto, el usuario lp debe poder escribir peticiones NCP en la conexión NCP al servidor.

Una manera fácil de conseguir esto sin que el usuario lp tenga que establecer su propia conexión e ingreso en el sistema es especificar lp como el dueño de una conexión establecida por otro usuario. Se lista un ejemplo completo de cómo configurar el sistema de impresión de GNU/Linux para que atienda trabajos de impresión de clientes a través de NetWare en tres pasos:

1. Escribir un *guión* de encapsulamiento (wrapper).

   El fichero `/etc/printcap` no permite que se le pasen opciones a los filtros. Por tanto, necesita escribir un pequeño *guión* que invoque la instrucción que desea junto con sus opciones. El *guión* de encapsulamiento puede ser tan simple como:

   ```
   #!/bin/sh
   # p2pslaser - sencillo script para redirigir stdin a la
   # cola PSLASER en el servidor REDS01
   #
   /usr/bin/nprint -S REDS01 -U stuart -q PSLASER
   #
   ```
   Guardar el *guión* en el fichero `/usr/local/bin/p2pslaser`.

2. Escribir la entrada en `/etc/printcap`.

   Necesitaremos configurar el *guión* `p2pslaser` que hemos creado como filtro de salida en `/etc/printcap`. Sería algo así:

   ```
   pslaser|Postguión Laser Printer hosted by NetWare server:\
   :lp=/dev/null:\
   :sd=/var/spool/lpd/pslaser:\
   :if=/usr/local/bin/p2pslaser:\
   :af=/var/log/lp-acct:\
   :lf=/var/log/lp-errs:\
   :pl#66:\
   :pw#80:\
   :pc#150:\
   :mx#0:\
   :sh:
   ```

3. Añadir la opción `c` a **ncpmount**.

   ```
   ncpmount -S REDS01 .... -c lp ....
   ```
   Nuestro usuario local stuart debe especificar al usuario lp como dueño de la conexión cuando monte el servidor NetWare remoto.

Ahora cualquier usuario de Linux puede elegir `pslaser` como el nombre de impresora cuando invoque a **lp**. El trabajo de impresión será enviado al servidor NetWare especificado y entrará en la cola de impresión.

15.8.2. Manejando Colas de Impresión

la orden **pqlist** lista todas las colas de impresión disponibles en el servidor especificado. Si no especifica un servidor de ficheros en la línea de orden con la opción `-S`, o un nombre de acceso y clave, éstos se tomarán de la entrada por defecto de su fichero `~/.nwclient`:

```
# pqlist -S vbrew_f1 -U guest -n
Server: ALES_F1
Print queue name                                         Queue ID
-----------------------------------------------------------------
TEST                                                     AA02009E
Q2                                                       EF0200D9
NPI223761_P1                                             DA03007C
```

```
    Q1                                                  F1060004
    I-DATA                                              0D0A003B
    NPI223761_P3                                        D80A0031
```

Nuestro ejemplo muestra una lista de las colas de impresión disponibles para el usuario `guest` en el servidor de ficheros `ALES_F1`. [151]

Para ver los trabajos de una cola de impresión, utilice la instrucción **pqstat**. Toma como argumento el nombre de la cola de impresión, y lista todos los trabajos de esa cola. Opcionalmente, puede proporcionarle otro argumento indicando cuántos trabajos de la lista quiere mostrar. La siguiente salida de ejemplo ha sido comprimida un poco para que quepa en el ancho de la página de este libro:

```
    $ pqstat −S ALES_F1 NPI223761_P1

    Server: ALES_F1     Queue: NPI223761_P1          Queue ID: 6A0E000C
       Seq  Name      Description                    Status   Form  Job ID
    −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
         1  TOTRAN    LyX document − propuesta.lyx    Active      0  02660001
```

Podemos ver que sólo hay un trabajo de impresión en la cola, que pertenece al usuario `TOTRAN`. El resto de las opciones incluyen una descripción del trabajo, su estado y su identificador de trabajo.

la orden **pqrm** se utiliza para eliminar trabajos de impresión de una cola de impresión especificada. Para eliminar el trabajo de la cola de la que acabamos de obtener el estado, sería:

```
    $ pqrm −S ALES_F1 NPI223761_P1 02660001
```

La orden es bastante simple, pero es pesado de utilizar cuando se tiene prisa. Sería un valioso proyecto escribir un *guión* básico para simplificar esta operación.

15.9. Emulación del Servidor NetWare

En GNU/Linux hay dos emuladores libres para servidores de ficheros NetWare. **lwared** ha sido desarrollado por Ales Dryak y **mars_nwe** por Martin Stover. Ambos paquetes proporcionan una emulación elemental del servidor de ficheros NetWare bajo GNU/Linux, permitiendo a clientes NetWare montar directorios de GNU/Linux exportados como volúmenes NetWare. Mientras que el servidor **lwared** es más sencillo de configurar, el servidor **mars_nwe** tiene más características. La instalación y configuración de estos paquetes está fuera de los objetivos de este capítulo, pero ambos están descritos en el IPX−HOWTO.

Capítulo 16. Administración deTaylor UUCP

UUCP fue diseñado a finales de los años setenta por Mike Lesk en los laboratorios Bell de AT&T con el objetivo de crear una simple red sobre líneas de teléfonos para conectarse mediante llamadas telefónicas. Dado que la mayoría de la gente que quiere tener correo electrónico y noticias de Usenet en sus ordenadores personales todavía se comunican por módem, UUCP ha seguido siendo muy popular. Aunque hay muchas implementaciones funcionando en una gran variedad de plataformas y sistemas operativos, todas son bastante compatibles.

Sin embargo, como con cualquier programa que se ha convertido en estándar con el tiempo, no hay un UUCP que se pueda denominar *el* UUCP. Ha sufrido un continuo proceso de evolución desde la primera versión que fue implementada en 1976. En la actualidad hay dos especies principales que se diferencian principalmente en su soporte del hardware y en su configuración. A su vez, hay varias implementaciones de estas dos clases, todas con ligeras diferencias respecto a sus familiares.

Una de las clases es la llamada UUCP Versión 2, que es una implementación de 1977 de Mike Lesk, David A. Novitz, y Greg Chesson. Aunque es bastante antigua, todavía se usa frecuentemente. Las implementaciones más recientes de la Versión 2 ofrecen muchas de las características de los tipos más nuevos de UUCP.

La segunda clase de UUCP se desarrolló en 1983, y se conoce comúnmente como BNU (Utilidades Básicas de Red, Basic Network Utilities) o HoneyDanBer UUCP. El último nombre deriva de los nombres de sus autores (P. Honeyman, D. A. Novitz, y B. E. Redman) y a menudo se abrevia en HDB, que es el término que usaremos en este capítulo. HDB se creó para eliminar algunas deficiencias de la Versión 2. Por ejemplo, se añadieron nuevos protocolos de transferencia, y se dividió el directorio de cola de manera que ahora sólo hay un directorio para cada ordenador con el que mantener tráfico UUCP.

La implementación de UUCP que se distribuye con GNU Linux es Taylor UUCP 1.06, versión en la que se está basado este capítulo. [152] La versión 1.06 de Taylor UUCP apareció en Agosto de 1995. Aparte de los ficheros de configuración tradicionales, Taylor UUCP también puede compilarse para entender los nuevos ficheros de configuración alias Taylor.

Taylor UUCP se suele compilar con compatibilidad HDB, el esquema de configuración Taylor o ambos. Al ser el esquema de Taylor mucho más flexible y probablemente más sencillo de entender que los ficheros de configuración HDB a menudo bastante oscuros, describiremos más abajo el esquema Taylor.

El propósito de este capítulo no es ofrecer una explicación exhaustiva de las opciones de la línea de órdenes para las instrucciones de UUCP y lo que hacen, sino darle una introducción sobre cómo poner en marcha un nodo de UUCP. La primera sección presenta una introducción de cómo UUCP implementa ejecución remota y transmisión de ficheros. Si usted tiene ya algunos conocimientos de UUCP, quizá desee saltarse esto y continuar con la sección Sección 16.2, que explica los distintos ficheros usados para configurar UUCP.

Sin embargo, asumiremos que usted está familiarizado con los programas de usuario del paquete UUCP. Éstos son **uucp** y **uux**. Si no los conoce suficientemente, consulte las correspondientes páginas de manual.

Aparte de los programas de usuario **uucp** y **uux**, el paquete UUCP contiene algunas órdenes más con fines únicamente administrativos. Se usan para monitorizar el tráfico UUCP en su nodo, eliminar ficheros antiguos de registro o crear estadísticas. No describiremos ninguna de estas utilidades porque son periféricas a las tareas principales de UUCP. Además, se encuentran bien documentadas y su comprensión resulta bastante sencilla; acuda a las páginas del manual para más información. De todos modos, hay una tercera categoría, que comprende los motores del UUCP. Se las conoce como **uucico** (donde *cico* significa copy−in copy−out), y **uuxqt**, que ejecuta tareas enviadas desde sistemas remotos. Nos concentraremos en estos dos importantes programas en este capítulo.

Si no le satisface la manera en que cubriremos estos temas, debería leer la documentación que viene con el paquete UUCP. Es un conjunto de ficheros Texinfo en los que se describe la instalación usando el esquema de configuración Taylor. Puede convertir los ficheros Texinfo a un fichero **dvi** con **texi2dvi** (que se encuentra en el paquete Texinfo de su distribución) y visualizar el fichero **dvi** por medio de la orden **xvi**.

El UUCP−HOWTO de Guylhem Aznar es otra buena fuente de información sobre UUCP en un entorno GNU/Linux. Se encuentra disponibles en cualquier servidor réplica del Linux Documentation Project y se envía con regularidad a comp.os.linux.answers.

También hay un grupo de noticias para la discusión de UUCP llamado comp.mail.uucp. Si tiene preguntas específicas sobre Taylor UUCP, será mejor que las haga allí en vez de en los grupos comp.os.linux.*.

16.1. Transferencias UUCP y ejecución remota

El concepto de tarea resulta vital para entender UUCP. Cada transferencia que inicia un usuario con **uucp** o **uux** se llama **tarea**. Consta de una orden a ejecutar en un sistema remoto, una recopilación de ficheros a transferir entre sitios o ambas cosas.

Por ejemplo, la siguiente orden hace que UUCP copie el fichero `netguide.ps` a un sistema remoto llamado pablo y ejecute la orden **lpr** en pablo para imprimir el fichero:

```
$ uux −r pablo!lpr !netguide.ps
```

Por lo general, UUCP no llama al sistema remoto de inmediato para llevar a cabo una tarea (o cualquier otra cosa que pueda hacer con **kermit**). En cambio, guarda la descripción de la tarea de manera temporal. Esto se conoce como *encolar*. El árbol de directorios bajo el que se guardan las tareas se llama por lo tanto *directorio de cola* y se encuentra generalmente en `/var/spool/uucp`. En nuestro ejemplo, la descripción de la tarea contendría información sobre la orden remota a ejecutar (**lpr**), el usuario que ordenó la ejecución y un par de elementos más. Además de la descripción de la tarea, UUCP tiene que guardar el fichero de entrada `netguide.ps`.

La localización y nomenclatura exactas de los ficheros de cola puede varias dependiendo de algunas opciones en tiempo de compilación. Los UUCPs compatibles con HDB guardan por lo general los ficheros de cola en el subdirectorio `/var/spool/uucp` con el nombre del sistema remoto. Cuando se compilan para la configuración Taylor, UUCP crea subdirectorios bajo el directorio de cola específico del sitio para diferentes tipos de ficheros de cola.

A intervalos regulares, UUCP llama al sistema remoto. Cuando se establece una conexión con el sistema remoto, UUCP transfiere los ficheros en los que se describe la tarea, además de los ficheros de entrada. Las tareas entrantes no se ejecutarán de inmediato, sino sólo tras haber terminado la conexión. La ejecución la gestion **uuxqt**, quien también se ocupa de redirigir cualquier tarea designada a otro sitio.

Para distinguir entre tareas más o menos importantes, UUCP asocia un *nivel* a cada tarea. Se trata de un único dígito de 0 a 9, de A a Z, y de a a z, en precedencia decreciente. El correo se suele colocar en la cola con nivel B o C, mientras que las noticias se colocan con un nivel N. Las tareas con niveles más altos se transfieren antes. Los niveles pueden asignarse por medio de la opción `g` al invocar a **uucp** o **uux**.

También se puede prohibir la transferencia de trabajos bajo un cierto nivel a horas determinadas. Esto también se llama *máximo nivel de cola* permitido durante una conversación y el valor predeterminado es z. Percátese de la ambigüedad de esta terminología: un fichero se transfiere sólo si es *igual* o *mayor* que el máximo nivel de cola.

16.1.1. El funcionamiento interno de uucico

Para comprender por qué **uucico** necesita saber ciertas cosas, una rápida descripción de cómo se conecta realmente a un sistema remoto resultará de ayuda.

Cuando usted ejecuta **uucico −s *`sistema`*** desde la línea de órdenes, primero tiene que conectarse físicamente. Las acciones a tomar dependen del tipo de conexión a usar. Por ejemplo, cuando se usa una línea telefónica, tiene que encontrar un módem, y marcar un número de teléfono. Sobre TCP, tiene que llamar `gethostbyname(3)` para convertir el nombre a una dirección de red, averiguar qué puerto abrir, y conectar la dirección al puerto correspondiente.

Una vez que se ha establecido la conexión, hay que pasar un proceso de autorización. Normalmente consiste en que el sistema remoto pide un nombre de usuario y posiblemente una clave. Esto se llama el *diálogo de entrada*. El proceso de autorización se lleva a cabo mediante el usual **getty/login**, o en conexiones TCP por el propio **uucico**. Si la autorización es permitida, la parte remota de la conexión ejecuta **uucico**. La copia local de **uucico** que inició la conexión se denomina *maestro*, y la copia remota se denomina *esclavo*.

Ahora viene la *fase de handshake*: El maestro envía su nombre, además de varias opciones. El esclavo comprueba el nombre para ver si tiene permiso para conectarse, para enviar y recibir ficheros, etc. Las opciones describen (entre otras cosas) el nivel máximo de ficheros de cola que hay que transferir. Si esta opción está activada, tiene lugar una cuenta de conversación, o *comprobación de la secuencia de llamada*. Con esta característica, ambos ordenadores mantienen una cuenta de conexiones exitosas, que se comparan. Si las cuentas no son iguales, la negociación de protocolos no tendrá lugar. Esto es útil para protegerse de impostores.

Finalmente los dos **uucico** tratan de ponerse de acuerdo en un *protocolo de transferencia* común. Este protocolo gobierna la manera en que los datos se transfieren, la manera en que se comprueba la consistencia de los datos, y la manera en que se retransmiten en caso de error. Hacen falta protocolos diferentes debido a los diferentes tipos de conexiones que se soportan. Por ejemplo, las líneas de teléfono precisan un protocolo seguro que es pesimista respecto a errores, mientras que una transmisión de TCP es fiable y puede usar un protocolo más eficiente que carece de la mayoría de las comprobaciones de errores.

Una vez que las negociaciones se han completado, comienza la fase de la verdadera transmisión. Ambos extremos ponen en funcionamiento el controlador del protocolo elegido. Los controladores posiblemente lleven a cabo alguna secuencia específica del protocolo para la inicialización.

Primero el maestro envía todos los ficheros en la cola de este sistema remoto cuyo nivel de cola es suficientemente alto. Cuando ha finalizado, informa al esclavo que ha terminado, y que el esclavo puede ahora colgar. El esclavo puede entonces colgar, o tomar el control de la conversación. Esto es un cambio de papeles: ahora el sistema remoto se convierte en maestro y el local en esclavo. El nuevo maestro envía ahora sus ficheros. Cuando ha terminado, ambos **uucico**s intercambian mensajes de terminación, y cierran la comunicación.

Si necesita información adicional sobre UUCP acuda al código fuente. También hay un artículo bastante antiguo escrito por David A. Novitz pululando por la red en el que se proporciona una descripción detallada del protocolo UUCP. [153] En las PUF sobre Taylor UUCP también se discuten algunos detalles de la implementación de UUCP. Se envía a comp.mail.uucp con relativa frecuencia.

16.1.2. Opciones en la línea de órdenes para uucico

En esta sección describimos las opciones de la línea de órdenes más importantes para **uucico**:

`system, s sistema`
: Llama al *`sistema`* si no está prohibido por restricciones en la hora de llamada.

S sistema
: Llama al *sistema* sin condiciones.

master, r1
: Inicia **uucico** en modo maestro. Éste es el modo predeterminado cuando se indica `s` o `S`. Por sí misma, la opción `r1` provoca que **uucico** intente llamar a todos los sistemas en el fichero `sys` que se describe en la siguiente sección de este capítulo, a no ser que sea prohibido por las restricciones de la hora de llamada o las veces que se puede reintentar la misma.

slave, r0
: Inicia **uucico** en modo esclavo. Éste es el modo predeterminado cuando no se indica `s` ni `S`. En modo esclavo, tanto la entrada como la salida estándar se asume que están conectadas a un puerto serie, o al puerto TCP especificado por la opción `p` si se usa.

ifwork, C
: La opción suplementa `s` o `S` y comunica a **uucico** que llame al sistema mencionado sólo si hay tareas en la cola para él.

debug tipo, x tipo, X tipo
: Activa la depuración del tipo especificado. Pueden proporcionarse muchos tipos en forma de lista separada por comas. Los siguientes tipos son válidos. abnormal, chat, handshake, uucp−proto, proto, port, config, spooldir, execute, incoming, y outgoing. Si usa all activará todas las opciones. Por compatibilidad con otras implementaciones de UUCP, también puede especificar un número, que activará la depuración para los primeros *n* elementos de la lista anterior.

  Los mensajes de depuración se registrarán en el fichero `Debug` bajo `/var/spool/uucp`.

16.2. Ficheros de configuración de UUCP

Al contrario que programas de transferencia de ficheros más simples, UUCP fue diseñado para ser capaz de llevar a cabo todas las tranferencias automáticamente. Una vez que está correctamente configurado, no es necesaria una constante participación del administrador. La información necesaria para esto se guarda en un par de ficheros de configuración que residen en el directorio `/usr/lib/uucp`. La mayoría de estos ficheros se usan sólo para conectarse a otro ordenador.

16.2.1. Una ligera introducción a Taylor UUCP

Decir que la configuración de UUCP es difícil sería una descripción insuficiente. Es cierto que es un asunto peliagudo, y el formato a veces demasiado conciso de los ficheros de configuración no hace las cosas más fáciles (aunque el formato de Taylor es casi fácil de leer comparado con los formatos más antiguos en HDB o Versión 2).

Para darle una idea de cómo se interactúa con estos ficheros, le introduciremos los más importantes, y echaremos un vistazo a algunos ejemplos. No explicaremos ahora todo en detalle; una explicación más precisa se describe en secciones posteriores. Si quiere configurar su ordenador para UUCP, puede comenzar con los ficheros de ejemplo, y adaptarlos gradualmente. Puede elegir los que se muestran a continuación, o los que se incluyen en su distribución de Linux preferida.

Todos los ficheros descritos en esta sección se guardan en `/etc/uucp` o en un subdirectorio de éste. Algunas distribuciones de Linux contienen binarios de UUCP con soporte tanto para la configuración HDB como Taylor activado, y emplean diferentes subdirectorios para cada grupo de ficheros de configuración. Seguramente habrá un fichero `README` en `/usr/lib/uucp`.

Para que UUCP funcione correctamente, estos ficheros tienen que pertenecer al usuario uucp. Algunos de ellos tienen claves y números de teléfono, y por lo tanto deberían tener permisos de 600. Aunque la mayoría de los comandos de UUCP tienen que tener el setuid a uucp, tiene que asegurarse de que el programa **uuchk** *no* lo es. Si no, los usuarios serían capaces de ver las claves aunque tengan modo 600.

El principal fichero de configuración de UUCP es `/etc/uucp/config`, que se usa para establecer las variables generales. La más importante (y por ahora la única) es el nombre UUCP de su máquina. En la Cervera Virtual, usan vstout como su pasarela UUCP:

```
# /etc/uucp/config − principal fichero de configuración de UUCP
nodename         vstout
```

El siguiente fichero de configuración en importancia es el fichero `sys`. Éste contiene toda la información específica al sistema de los ordenadores con los que usted se conecta. Esto incluye el nombre del ordenador, e información sobre la propia conexión, tal como el número de teléfono cuando se usa una conexión por módem. Un ejemplo típico para un ordenador llamado pablo que se conecta por módem sería:

```
# /usr/lib/uucp/sys − vecinos UUCP
# system: pablo
system          pablo
time            Any
phone           123−−456
port            serial1
speed           38400
chat            ogin: vstout ssword: lorca
```

time especifica las horas a las que puede llamarse al sistema remoto. chat describe la macro del diálogo de entrada la secuencia de caracteres que deben intercambiarse para permitir que **uucico** entre en pablo. Volveremos a las macros más tarde. El elemento port simplemente nombra una entrada en el fichero `port`. (Acuda a Figura 16−1.) Puede asignar cualquier nombre siempre que haga referencia a una entrada válida en `port`.

El fichero `port` contiene información específica a la propia conexión. Para conexiones por módem, describe el fichero de dispositivo a usar, el conjunto de velocidades soportadas, y el tipo de equipo de marcación conectado al puerto. La entrada a continuación describe `/dev/ttyS1` (o sea, el puerto COM 2), en el cual hay un módem NakWell conectado que es capaz de funcionar a velocidades de hasta 38400 bps. El nombre de la entrada se puede elegir para que coincida con el nombre usado en el fichero `sys`.

```
# /etc/uucp/port − puertos de UUCP
# /dev/ttyS1 (COM2)
port            serial1
type            modem
device          /dev/ttyS1
speed           38400
dialer          nakwell
```

La información que afecta al propio marcador se mantiene en otro fichero, llamado lo adivinaste`dial`. Para cada tipo de marcador, contiene básicamente la secuencia de instrucciones necesarias para llamar a otro ordenador, dado el número de teléfono. Una vez más, esto se especifica como una macro de diálogo. Por ejemplo, la entrada para el anterior NakWell puede parecerse a esta:

```
# /etc/uucp/dial − información por marcador
# módems NakWell
dialer          nakwell
chat            "" AT&F OK ATDT\T CONNECT
```

La línea que empieza con chat especifica el diálogo del módem, que no es sino la secuencia de órdenes enviadas y recibidas del módem para inicializarlo, y para hacerle marcar el número deseado. La secuencia \T será reemplazada con el número de teléfono por el programa **uucico**.

Para darle una idea a grandes rasgos de cómo utiliza **uucico** estos ficheros de configuración, suponga que utiliza la orden:

```
$ uucico -s pablo
```

Lo primero que hace **uucico** es buscar pablo en el fichero `sys`. A partir de la entrada en el fichero `sys` para pablo, el programa averigua que debería usar el puerto serial1 para establecer la conexión. El fichero `port` le dice a **uucico** que se trata de un puerto de módem al que hay conectado un módem NakWell.

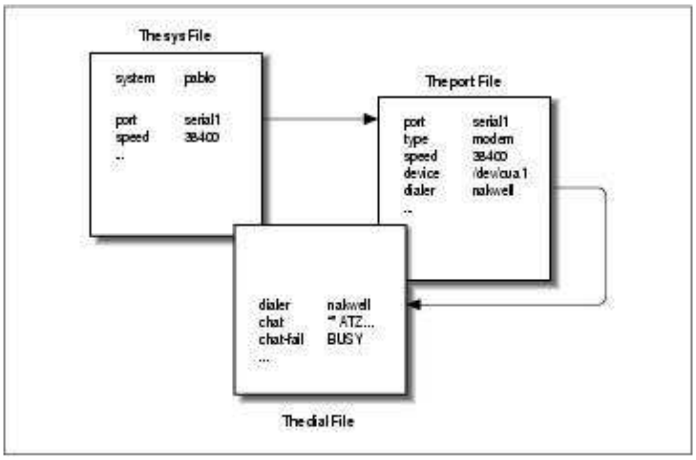
uucico busca ahora en `dial` la entrada en la que se describe el módem NakWell, y al encontrarla, abre el puerto serie `/dev/ttyS1` y ejecuta el diálogo de marcación. Es decir, envía **AT&F**, espera la respuesta **OK**, etc. Cuando encuentra la cadena \T, la sustituye por el número (123−−456) extraído del fichero `sys`.

Cuando el módem devuelve **CONNECT**, la conexión se ha establecido y el diálogo de marcación se ha completado. **uucico** vuelve ahora al fichero `sys` y ejecuta el diálogo de entrada. En nuestro ejemplo, esperaría al **login:**, enviaría entonces su nombre de usuario (**vstout**), esperaría a que se le solicitase el **password:** y enviaría la contraseña (**lorca**).

Tras completar la autorización, se supone que el sistema remoto ejecuta su propio **uucico**. Entran los dos entonces en la fase de negociación descrita en la sección previa.

Figura 16−1 ilustra las dependencias entre los ficheros de configuración.

Figura 16−1. Interacción entre los ficheros de configuración de Taylor UUCP



16.2.2. Lo que UUCP necesita saber

Antes de empezar a escribir los ficheros de configuración, debe conseguir cierta información que UUCP necesita.

Primero, tiene que averiguar a qué puerto serie está conectado su módem. Normalmente, los puertos (DOS) COM1: a COM4: se corresponden con los ficheros especiales de los dispositivos `/dev/ttyS0` a `/dev/ttyS3`. Algunas distribuciones, como Slackware, crean un enlace llamado `/dev/modem` apuntando al fichero del dispositivo `ttyS*` apropiado, y configuran **kermit**, **seyon** y cualquier otro programa de comunicaciones para usar este fichero genérico. En ese caso, debería usar `/dev/modem` en la configuración de UUCP también.

La razón para esto es que todos los programas, para llamar por teléfono, usan unos *ficheros de bloqueo* para indicar cuándo un puerto serie está en uso. Los nombres de estos ficheros cerrojo son una concatenación del texto `LCK..` y el nombre del fichero de dispositivo, por ejemplo, `LCK..ttyS1`. Si los programas usasen nombres diferentes para un mismo dispositivo, no podrían reconocer los ficheros cerrojo de los otros programas. En consecuencia, perturbarían la sesión de conexión de cada uno si se ejecutan a la vez. Esto no es raro que ocurra cuando organiza sus llamadas de UUCP usando una entrada en el fichero `crontab`. Para más detalles sobre la configuración de puertos serie, acuda a Capítulo 4.

A continuación tiene que averiguar a qué velocidad se comunicarán su módem y GNU/Linux. Tendrá que ajustar este valor a la velocidad de transferencia efectiva máxima que espere obtener. La velocidad efectiva puede ser mucho mayor que la velocidad física de tranferencia de su módem. Por ejemplo, muchos modems envían y reciben datos a 56 kbps. Usando protocolos de compresión como V.42bis, la velocidad real de transferencia puede alcanzar los 100 kbps.

Por supuesto, si quiere que UUCP sirva de algo, necesitará el número de teléfono al que llamar. También necesitará un nombre de usuario válido y probablemente una clave en el sistema remoto.[154]

También necesitará saber *exactamente* cómo entrar en el sistema. Por ejemplo, ¿tiene que pulsar la tecla Enter antes de que aparezca la pregunta de nombre de usuario?. ¿Muestra el sistema remoto un `login:` o `user`?. Esto es necesario para escribir la *macro de diálogo*, que es un *guión* que le dice a **uucico** cómo entrar. Si no lo sabe, o si la macro de diálogo normal no funciona, intente llamar al sistema con un programa como **kermit** o **minicom**, y apunte exactamente lo que tiene que hacer.

16.2.3. Nomenclatura de nodos

Al igual que en redes basadas en TCP/IP, todas las máquinas necesitan tener un nombre para la red de UUCP. Mientras sólo quiera usar UUCP para transferencia de ficheros desde y hacia ordenadores que usted llama directamente, o en una red local, el nombre no tiene que ajustarse a ninguna regla.[155]

De todas formas, si usa UUCP para una conexión de correo o noticias, debería pensar en registrar el nombre en el Proyecto de Mapeado UUCP. [156] El Proyecto de Mapeado UUCP se describe en Capítulo 17. Incluso aunque forme parte de un dominio, debería considerar tener un nombre UUCP oficial para su sitio.

Con frecuencia la gente elige su nombre UUCP de forma que coincida con el primer elemento de su nombre de dominio completamente cualificado. Suponga que la dirección de su dominio es swim.twobirds.com; entonces el nombre de su nodo UUCP sería swim. Piense en los nodos UUCP como si sólo se conociesen entre ellos por sus respectivos nombres propios. Por supuesto, también puede usar un nombre UUCP que no tenga nada que ver con su nombre de dominio completamente cualificado.

No obstante, asegúrese de no emplear un nombre de sitio no cualificado en direcciones de correo a menos que lo haya registrado como su nombre UUCP oficial. En el mejor de los casos, el correo a una máquina UUCP no registrada se perderá en algún enorme agujero negro digital. Si emplea un nombre que alguien ya esté usando, el correo se dirigirá a ese lugar causando al administrador del correo de ese lugar un sinfín de dolores de cabeza.

De manera predeterminada, UUCP usa el nombre especificado como **hostname** como el nombre UUCP del sitio. Este nombre suele adjudicarlo una orden en los guiones `rc` durante el arranque del sistema, y se suele guardar en `/etc/hostname`. Si su nombre UUCP difiere del de su máquina, tendrá que usar la opción **hostname** en el fichero `config` para comunicarle a **uucico** su nombre UUCP. Esto se describe más tarde.

16.2.4. Ficheros de configuración de Taylor

Volvemos ahora a los ficheros de configuración. Taylor UUCP obtiene su información de los siguientes ficheros:

`config`
: Éste es el principal fichero de configuración. Aquí puede definir el nombre de su sitio UUCP.

`sys`

En este fichero se describen todos los sitios conocidos. Para cada sitio se especifica su nombre, a qué horas llamarlo, qué número marcar (si es el caso), qué tipo de dispositivo usar, y cómo entrar en él.

port
: Este fichero contiene entradas en las que se describe cada puerto disponible, junto a la velocidad de la línea soportada y las instrucciones de marcación.

dial
: En este fichero se describen los marcadores a usar para establecer una conexión telefónica.

dialcode
: Este fichero contiene expansiones para códigos de marcación simbólicos.

call
: Este fichero contiene el nombre y la contraseña a utilizar cuando se llama a un sistema. Raramente se usa.

passwd
: Este fichero contiene los nombres y las contraseñas que pueden usar los sistemas al conectarse. Sólo se usa cuando **uucico** lleva a cabo su propia validación de contraseñas.

Los ficheros de configuración de Taylor se componen generalmente de líneas que contienen pares clave−valor. Una almohadilla inicia un comentario que se extiende hasta el final de la línea. Para emplear el signo de la almohadilla como tal, escápelo con una barra invertida de esta manera: `\#`.

Hay unas cuantas opciones que puede ajustar con estos ficheros de configuración. No podemos repasar todos los parámetros, sino que cubriremos sólo los más importantes. Con éstos usted podrá configurar una conexión de UUCP por módem. Otras secciones describirán las modificaciones necesarias si quiere usar UUCP en TCP/IP o sobre una línea serie. Junto con el código fuente de Taylor UUCP se incluye una referencia de órdenes completa en los documentos Texinfo.

Cuando crea haber configurado su sistema UUCP completamente, puede compobar su configuración con la herramienta **uuchk** (que puede encontrar en `/usr/lib/uucp`). **uuchk** lee sus ficheros de configuración y le muestra un informe detallado de los valores de configuración usados para cada sistema.

16.2.5. Opciones generales de configuración usando el fichero config

Normalmente no usará este fichero para otra cosa que especificar el nombre de su nodo UUCP. De manera predeterminada, UUCP usará el nombre que haya establecido con la orden **hostname**, pero por lo general resulta una buena idea especificar el nombre UUCP explícitamente. He aquí un fichero `config` de ejemplo:

```
# /usr/lib/uucp/config − principal fichero de configuración UUCP
hostname        vstout
```

Por supuesto, también existen otros parámetros configurables aquí, como los referentes al nombre del directorio de colas, o los nombres de acceso para el UUCP anónimo. Esto último se describirá posteriormente en este capítulo, en la sección UUCP anónimo.

16.2.6. Cómo informar a UUCP sobre otros sistemas mediante el fichero sys

En el fichero `sys` se describen los sistemas que conoce su máquina. La clave system nos presenta una nueva entrada; las líneas siguientes hasta la próxima directiva system detallan las variables específicas de cada sitio. Comúnmente, una entrada de sistema define variables como el número de teléfono y el diálogo de entrada.

Las variables anteriores a la primera línea system especifican valores predeterminados a usar en todos los sistemas. Normalmente, colocará en esta sección variables del protocolo y similares.

Los campos más importantes se tratan en detalle en las siguientes secciones.

16.2.6.1. Nombre del sistema

La orden *system* nombra el sistema remoto. Debe especificar el nombre correcto del sistema remoto, no un alias que se invente, porque **uucico** lo comparará con la identificación que reciba del sistema remoto una vez se conecte a él. [157]

Cada nombre de sistema puede aparecer una sola vez. Si quiere usar varias configuraciones para un mismo sistema (por ejemplo, números de teléfono diferentes que **uucico** puede usar alternativamente), puede especificar *alternativas*, que se describen más adelante.

16.2.6.2. Número de teléfono

Si va a conectarse con el sistema remoto por vía telefónica, en el campo phone se especifica el número que debería marcar el módem. Puede contener varios separadores que interpretará el procedimiento de marcado de **uucico**. Un signo de igual (=) significa esperar un tono de marcado secundario y un guión (−) genera una pausa de un segundo. Algunas instalaciones telefónicas pueden atrancarse si no se realizan pausas entre códigos de acceso especiales y los números de teléfono.[158]

A menudo resulta conveniente usar nombres en vez de números para describir los códigos de marcado según la zona. El fichero `dialcode` le permite asociar un nombre con un código que use al especificar números de teléfono para las máquinas remotas. Suponga que tiene el siguiente fichero `dialcode`:

```
# /usr/lib/uucp/dialcode − traducción de los códigos de marcación
Bogoham          024881
Coxton           035119
```

Con estas traducciones, puede usar un número de teléfono tal que Bogoham7732 en el fichero `sys`, que lo hará probablemente algo más legible y mucho será mucho más fácil actualizar el código de marcación para Bogoham cada vez que cambie.

16.2.6.3. puerto y velocidad

Las opciones de puerto y velocidad se usan para elegir el dispositivo a usar para llamar al sistema remoto y la velocidad máxima a la que debería ajustarse el dispositivo.[159] En una entrada de system se puede usar una opción o varias de manera conjunta. Cuando se busca un dispositivo adecuado en el fichero `port`, sólo se eligen los dispositivos con un nombre de puerto y/o rango de velocidad que coincidan con los especificados.

Por lo general debería ser suficiente utilizar únicamente la opción speed. Si sólo dispone de un dispositivo serie definido en `port`, **uucico** siempre toma el adecuado por lo que sólo tiene que especificar la velocidad deseada. Si tiene varios módems conectados a sus sistemas, con frecuencia no querrá nombrar un puerto concreto, porque si **uucico** encuentra que muchos coinciden prueba con cada dispositivo hasta que encuentra uno que no se esté usando.

16.2.6.4. El diálogo de entrada

Antes ya nos encontramos con la macro del diálogo de entrada, que le dice a **uucico** cómo entrar en el sistema remoto. Consiste de una lista de palabras clave, que especifican el texto que se espera y el que se envía por el proceso local de **uucico**. El objetivo es hacer que **uucico** espere hasta que la máquina remota envíe una línea pidiendo el nombre de usuario, y entonces enviar el nombre de usuario, luego esperar a que pida la palabre clave, y enviar dicha clave. Los textos de espera y de envío se dan alternativamente. **uucico** automáticamente añade un avance de línea (\r) a cualquier texto enviado. Por lo tanto, una macro de diálogo sencilla sería parecida a esta:

```
    ogin: vstout ssword: catch22
```

Dése cuenta de que los campos de texto de espera probablemente no contendrán el texto completo. Esto es así para asegurarse de que el proceso de entrada se lleve a cabo aunque el sistema remoto nos envíe **Login:** en vez de **login:**. Si la cadena que está esperando o enviando contiene espacios u otros caracteres de espacios en blanco, debe usar comillas para delimitar el texto.

uucico también permite usar estructuras condicionales, por ejemplo en el caso de que el programa **getty** de la máquina remota necesite ser reinicializado antes de enviar una pregunta. Por esta razón, usted puede añadir un sub−diálogo a un texto de espera, separado con un guión. El sub−diálogo se ejecuta sólo si el primer texto de espera falla, ej. si expira un temporizador. Una manera de usar esta característica es enviar un BREAK si el sistema remoto no envía una pregunta de nombre de usuario. El siguiente ejemplo muestra un ejemplo de una macro de diálogo que debería funcionar también en el caso de que usted tenga que pulsar Enter antes de que aparezca la pregunta de entrada. El primer parámetro vacío, `""`, comunica a UUCP que no espere nada sino que continúe con la siguiente cadena de envío:

```
    "" \n\r\d\r\n\c ogin:-BREAK-ogin: vstout ssword: catch22
```

Hay un par de cadenas de caracteres especiales y caracteres de escape que pueden aparecer en la macro de diálogo. Esta es una lista incompleta de caracteres legales en la pregunta de espera:

""
: La cadena vacía comunica a **uucico** que no espere nada, sino que siga de inmediato con la siguiente cadena enviada.

t
: Carácter de tabulador.

r
: Carácter de retorno de línea.

s
: Carácter de espacio. Se necesita para incluir espacios en un diálogo.

n
: Carácter de línea nueva.

\\\\
: Carácter de barra invertida.

En cadenas de caracteres de envío se pueden incluir, además de los mencionados anteriormente, los siguientes caracteres:

EOT
: Carácter de fin de transmisión (**^D**).

BREAK
: Carácter Break.

c
: Suprime el envío del carácter del línea nueva al final de cada cadena de caracteres.

d
: Retrasa el envío 1 segundo.

E

Activa la comprobación de eco. De esta forma, **uucico** esperará a leer el eco de todo lo que escribe en el dispositivo antes de que continúe con el diálogo. Se usa principalmente en diálogos de modems (que veremos más adelante). La comprobación de eco está desactivada por omisión.

e

Desactiva la comprobación del eco.

K

Lo mismo que BREAK.

p

Pausa de una fracción de segundo.

16.2.6.5. Alternativas

A veces es deseable tener múltiples entradas para un mismo sistema, por ejemplo si se puede acceder al sistema en diferentes líneas de módem. Con Taylor UUCP se puede hacer esto definiendo una *alternativa*

Una entrada alternativa mantiene todas las características de la entrada principal, y especifica solamente aquellos valores que tienen que ser cambiados, o añadidos. Una alternativa está separada de la entrada principal por una línea que contiene la palabre clave alternate

Para usar dos números de teléfono para pablo, habría que modificar su entrada `sys` de la siguiente manera:

```
system       pablo
phone        123−456
.. lo mismo de antes ...
alternate
phone        123−455
```

Ahora, cuando llame a pablo, el programa **uucico** marcará primero el 123−456, y si no funciona, probará la alternativa. La entrada alternativa retiene toda la otra información de la entrada de sistema principal, y altera sólo el número de teléfono.

16.2.6.6. Restringir horas de llamada

Taylor UUCP proporciona varios métodos para restringir las horas a las que se pueden efectuar llamadas a un sistema remoto. Una razón para hacer esto sería por las limitaciones que el sistema remoto impone en sus servicios durante horas de oficina, o simplemente para evitar las horas más caras. Siempre se pueden desactivar las restricciones con la opción `S` o `f` en el programa **uucico**.

Por defecto, Taylor UUCP no permite conexiones a ninguna hora, así que usted *tiene que* especificar algún horario en el fichero `sys`. Si no le importan las restricciones, puede especificar la opción **time** con un valor de Any en su fichero `sys`.

La manera más sencilla de restringir los horarios de las llamadas es incluir una entrada time seguida de una cadena formada por los subcampos día y hora. Día puede ser una combinación de Mo, Tu, We, Th, Fr, Sa, y Su. También puede especificar Any, Never, o Wk para los días laborables. La hora está formada por dos valores de reloj de 24 horas separados por un guión. Especifican las horas durante las que pueden efectuarse llamadas. La combinación de estos elementos se escribe sin espacios en blanco entre ellos. Se pueden especificar varios pares día−hora separados por comas, tal y como se muestra en esta línea:

```
time            MoWe0300−0730,Fr1805−2200
```

En este ejemplo se permiten llamadas en Lunes y Miércoles, de 3 de la mañana a 7:30, y los Viernes entre las 6:05 y las 8:00 de la tarde. Cuando un campo de hora incluye la medianoche, como Mo1830−0600, en realidad quiere decir el Lunes, entre medianoche y las 6 de la mañana, y entre las 6:30 de la tarde y medianoche.

Las palabras especiales Any y Never significan que se pueden hacer llamadas siempre o nunca, respectivamente.

Taylor UUCP también tiene algunos elementos especiales que puede usar en cadenas de tiempo como NonPeak y Night. Estos elementos especiales son abreviaturas de Any2300−0800,SaSu0800−1700 y Any1800−0700,SaSu respectivamente.

La orden *time* tiene una segunda variable opcional que describe el tiempo a esperar para reintentar en minutos. Cuando un intento de conexión falla, **uucico** no permitirá otro intento de llamar al ordenador remoto hasta que transcurra un cierto tiempo. De manera predeterminada, **uucico** usa un algoritmo de espera exponencial, según el cual el intervalo de espera se incrementa con cada intento fallido. Por ejemplo, si especifica un tiempo de reintento de 5 minutos, **uucico** no aceptará llamar otra vez en los 5 minutos después del ultimo intento fallido.

La orden **timegrade** le permite adjuntar un rango máximo de cola a un calendario. Por ejemplo, asuma que tiene las siguientes órdenes **timegrade** en una entrada system:

```
timegrade           N Wk1900-0700,SaSu
timegrade           C Any
```

Esto permite que los trabajos con rango de cola de C o mayor (normalmente el correo se pone en la cola con rango B o C) sean transferidos siempre que se establece una comunicación, mientras que las noticias (normalmente con rango N) serán transferidas sólo durante la noche y los fines de semana.

Al igual que **time**, la orden **timegrade** toma un intervalo entre reintentos de minutos como una tercera variable opcional.

De todas formas, hay que hacer una observación sobre los rangos de la cola. Primero, la opción **timegrade** sólo se afecta a lo que *sus* sistemas envían; el sistema remoto puede transferir lo que quiera. Puede usar la opción **call−timegrade** para solicitarle de manera explícita que envíe solamente tareas por encima de un determinado rango de cola; pero no hay hay ninguna garantía de que vaya a obedecer a su petición.[160]

De manera similar, el campo *`timegrade`* no se comprueba cuando llama un sistema remoto, por lo que se le enviará cualquier tarea de la cola que sea para él. De todos modos, el sistema remoto puede solicitar explícitamente a su **uucico** que se ocupe únicamente de cierto rango de la cola.

16.2.7. Identificar dispositivos disponibles mediante el fichero port

El fichero `port` hace saber a **uucico** los puertos disponibles. Se trata normalmente de puertos de módem, pero también se soportan otros tipos como las líneas serie y los sockets de TCP.

Al igual que el fichero `sys`, `port` está formado por entradas separadas que comienzan con la palabra clave port seguida del nombre del puerto. Este nombre también puede usarse en la sentencia port del fichero `sys`. No es necesario que el nombre sea único; si hay muchos puertos con el mismo nombre, **uucico** probará con cada uno hasta que encuentre alguno que pueda usar.

La orden **port** debería estar seguida inmediatamente por la sentencia type, que indica qué tipo de puerto se describe. Tipos válidos son modem, direct para conexiones directas y tcp para sockets de TCP. De no existir la orden **port** se usará de manera predeterminada módem como tipo de puerto.

En esta sección sólo hablaremos de puertos de módem; los puertos TCP y las líneas directas se tratarán en una sección posterior.

Tanto para el módem como para los puertos directos, debe especificar el dispositivo para llamar por medio de la directiva device. Normalmente, se trata del nombre del fichero especial de dispositivo del directorio `/dev`, como `/dev/ttyS1`.

En el caso de un módem, la entrada del puerto también determina qué tipo de módem hay conectado al puerto. Los diferentes tipos de módem tienen que configurarse de manera diferente. Incluso los módems que dicen ser compatibles con Hayes no son siempre realmente compatibles unos con otros. Por lo tanto, tiene que comunicarle a **uucico** cómo inicializar el módem y hacerle marcar el número deseado. Taylor UUCP mantiene las descripciones de todos los marcadores en un fichero llamado `dial`. Para usar cualquiera de éstos, tiene que especificar el nombre del marcador mediante la orden **dialer**.

A veces querrá usar un módem de diferentes maneras dependiendo de a qué sistema llame. Por ejemplo, algunos módems antiguos no entienden cuando un módem rápido trata de conectar a 56 kbps; simplementen dejan caer la línea en vez de negociar una conexión a 9.600 bps, por ejemplo. Cuando sabe que el sitio pesado usa un módem tan tonto, tiene que configurar su módem de una manera diferente cuando le llame. Para esto, necesita una entrada de puerto adicional en el fichero `port` en la que especificar un marcador diferente. Ahora puede darle al nuevo puerto un nombre diferente, como serie1−lento y usar la directiva port en la entrada del sistema pesado en `sys`.

Otra manera de distinguir los puertos es por la velocidad que usan. Por ejemplo, las dos entradas de puerto de la situación anterior pueden ser así:

```
# módem Nakwell; conectar a alta velocidad
port             serie1         # port name
type             modem           # modem port
device           /dev/ttyS1      # this is COM2
speed            115200          # supported speed
dialer           nakwell         # normal dialer
# módem Nakwell; conectar a baja velocidad
port             serie1         # port name
type             modem           # modem port
device           /dev/ttyS1      # this is COM2
speed            9600            # supported speed
dialer           nakwell-slow    # don't attempt fast connect
```

La entrada de sistema para el sitio pesado daría ahora serie1 como el nombre del puerto, pero solicitaría usarlo sólo a 9.600 bps. **uucico** usa entonces automáticamente la segunda entrada de puerto. Al resto de sitios que tengan una entrada de 115.200 bps en la entrada del sistema se les llamará usando la primera entrada de puerto. De manera predeterminada, se usará la primera entrada con una velocidad que coincida.

16.2.8. Cómo marcar un número usando el fichero dial

En el fichero `dial` se describen las maneras de utilizar diferentes marcadores. Tradicionalmente, UUCP habla de marcadores y no tanto de módems, porque antaño era habitual disponer de un (caro) dispositivo de marcado automático que servía a un completo banco de módems. Hoy en día la mayoría de los módems llevan ya el soporte de marcación integrado, por lo que esta distinción ha tendido a desvanecerse.

No obstante, marcadores o módems diferentes pueden requerir una configuración diferente. Puede describir cada uno de ellos en el fichero `dial`. Las entradas de `dial` comienzan con la orden **dialer** que proporciona el nombre del marcador.

La entrada más importante aparte de **dialer** es el diálogo del módem, especificado por la orden **chat**. Similar al diálogo de entrada en el sistema, consta de una secuencia de cadenas que **uucico** envía al marcador y las respuestas que espera como respuesta. Suele usarse para reiniciar el módem a algún estado conocido y marcar el número. En la siguiente entrada de dialer de ejemplo se muestra un típico diálogo de módem para un módem compatible con Hayes:

```
# módem NakWell; conectar a alta velocidad
dialer          nakwell         # nombre del marcador
chat            "" AT&F OK\r ATH1E0Q0 OK\r ATDT\T CONNECT
chat-fail       BUSY
chat-fail       ERROR
chat-fail       NO\sCARRIER
dtr-toggle      true
```

El diálogo comienza con `""`, la cadena vacía esperada. **uucico** envía entonces la primera orden **AT&F**. **AT&F** es la orden Hayes para reiniciar el módem a la configuración predeterminada de fábrica. **uucico** espera entonces hasta que el módem haya enviado `OK` y envía la siguiente orden, que desactiva el eco local y cosas así. Tras devolver el módem `OK` nuevamente, **uucico** envía la orden de marcado **ATDT**. La secuencia de escape `\T` de esta cadena se sustituye por el número de teléfono tomado de la entrada de sistema del fichero `sys`. **uucico** espera entonces a que el módem le devuelva la cadena `CONNECT`, que indica que se ha establecido con éxito la conexión con el módem remoto.

A veces el módem falla al conectar con el sistema remoto; por ejemplo, si el otro sistema está comunicándose con alguien más y la línea está ocupada. En este caso, el módem devuelve un mensaje de error indicando la razón. Los diálogos de módem son incapaces de detectar este tipo de mensajes; **uucico** sigue esperando la cadena esperada hasta que se agota el temporizador. El fichero de registro de UUCP sólo muestra entonces un tiempo agotado en el guión de diálogo en vez de la razón específica.

No obstante, Taylor UUCP le permite informar a **uucico** sobre estos mensajes de error usando la orden **chat−fail** como se ve en el ejemplo. Cuando **uucico** detecta una cadena de caracteres de error en el diálogo mientras lo ejecuta, interrumpe la llamada y anota el error en el fichero de registro de UUCP.

En la última orden del ejemplo anterior se comunica a UUCP que cambie la línea de control DTR (Terminal de Datos Preparado) antes de iniciar el diálogo del módem. Normalmente, el controlador serie levanta DTR cuando un proceso abre el dispositivo para decirle al módem conectado que alguien quiere hablar con él. La prestación dtr−toggle deja caer DTR, espera un momento, y lo levanta de nuevo. Muchos módems pueden configurarse para reaccionar ante una caída de DTR entrando en "off−hook", entrando en estado de órdenes o reiniciándose ellos mismos. [161]

16.2.9. UUCP sobre TCP

Por muy absurdo que suene en principio, el uso de UUCP para transferir datos sobre TCP no es una idea tan mala, especialmente cuando se transfieren grandes cantidades de datos como los grupos de noticias Usenet. En conexiones basadas en TCP, los grupos de noticias se transmiten generalmente usando el protocolo NNTP, según el cual los artículos se piden y se transmiten individualmente, sin compresión ni ninguna otra optimización. Aunque es una técnica adecuada para ordenadores grandes con varias fuentes de grupos de noticias simultáneas, esta técnica no es favorable para pequeños sistemas que reciben los grupos a través de una conexión lenta, como RDSI. Estos ordenadores normalmente desean combinar las cualidades de TCP con las ventajas de enviar artículos en grandes lotes, que se pueden comprimir y por lo tanto transferir con muy poco gasto. Un método estándar de enviar estos lotes es usando UUCP sobre TCP.

En `sys`, especificaríamos que se llamase a un sistema por TCP de esta manera:

```
system          gmu
address         news.groucho.edu
time            Any
port            tcp-conn
chat            ogin: vstout word: clouseau
```

La orden **address** da la dirección IP de la máquina o su nombre de dominio completamente cualificado. La entrada `port` correspondiente sería tal que así:

```
port            tcp-conn
type            tcp
```

```
    service         540
```

En la entrada se afirma que debería usarse una conexión TCP cuando una entrada `sys` hiciese referencia a tcp−conn, y que **uucico** debería intentar conectarse al puerto 540 de la red TCP en la máquina remota. Éste es el número de puerto predeterminado del servicio UUCP. En vez del número de puerto, también puede proporcionar un nombre de puerto simbólico a la orden **service**. El número de puerto correspondiente a este nombre se buscará en `/etc/services`. El nombre común para el servicio UUCP es uucpd.

16.2.10. Usar una conexión directa

Supongamos que usa una línea directa para conectar su sistema vstout con tiny. Al igual que en el caso del módem tiene que escribir una entrada de sistema en el fichero `sys`. La orden **port** identifica el puerto serie en el que está conectado tiny:

```
    system          tiny
    time            Any
    port            direct1
    speed           38400
    chat            ogin: cathcart word: catch22
```

En el fichero `port`, tiene que describir el puerto serie para la conexión directa. Una entrada dialer no es necesaria porque no hay necesidad de marcar:

```
    port            direct1
    type            direct
    speed           38400
    device                  /dev/ttyS1
```

16.3. Controlar el acceso a las prestaciones de UUCP

UUCP es un sistema bastante flexible. Con esa flexibilidad viene la necesidad de controlar cuidadosamente el acceso a sus prestaciones para prevenir abusos, tanto intencionados como accidentales. Las principales propiedades a tener en cuenta por el administrador de UUCP son la ejecución de órdenes remotas, la transferencia de ficheros y el reenvío. Taylor UUCP proporciona medios para limitar la libertad de máquinas UUCP remotas al aprovechar cada una de estas prestaciones. Con una cuidadosa selección de los permisos, el administrador de UUCP puede asegurarse de preservar la seguridad de la máquina.

16.3.1. Ejecución de órdenes

Es tarea de UUCP copiar ficheros de un sistema a otro y solicitar la ejecución de ciertas órdenes en sistemas remotos. Evidentemente, usted como administrador querrá controlar qué derechos garantiza a otros sistemas permitirles ejecutar cualquier orden que elijan en su sistema definitivamente no es una buena idea.

De manera predeterminada, las únicas órdenes que permite ejecutar Taylor UUCP a otros sistemas en su máquina son **rmail** y **rnews**, que se usan habitualmente para intercambiar correo−e y noticias de Usenet sobre UUCP. Para cambiar el conjunto de órdenes para un sistema en particular, puede usar la palabra clave commands en el fichero `sys`. De manera similar, puede querer limitar la ruta de búsqueda a los directorios que contengan las órdenes permitidas. Puede cambiar la ruta de búsqueda permitida para una máquina remota con la sentencia command−path . Por ejemplo, puede querer permitir al sistema pablo ejecutar la orden **bsmtp** además de **rmail** y **rnews**:[162]

```
    system          pablo
    ...
    commands        rmail rnews bsmtp
```

16.3.2. Transferencias de ficheros

Taylor UUCP también le permite ajustar las transferencias de ficheros con un gran detalle. De manera extrema, puede desactivar las transferencias hacia y desde un sistema en particular. Simplemente configure request como no, y el sistema remoto no podrá ni descargar ficheros de su sistema ni enviarle fichero alguno. De manera similar, puede prohibir a sus usuarios que transfieran ficheros hacia o desde un sistema configurando transfer como no. De manera predeterminada, se permite a los usuarios de los sistemas local y remoto tanto cargar como descargar ficheros.

Además, puede configurar hacia y desde qué directorios pueden copiarse ficheros. Normalmente querrá restringir el acceso desde los sistemas remotos a una jerarquía de un único directorio, pero permitiendo a sus usuarios el envío de ficheros desde sus directorios de usuario. Comúnmente, se permite a los usuarios remotos recibir ficheros sólo desde el directorio UUCP público `/var/spool/uucppublic`. Éste es el lugar tradicional donde poner los ficheros disposición pública, de manera similar a los servidores FTP en Internet. [163]

Taylor UUCP ofrece cuatro órdenes diferentes para configurar los directorios de envío y recepción de ficheros. Se trata de: **local−send**, que especifica la lista de directorios desde los que un usuario puede solicitar a UUCP que envíe ficheros; **local−receive**, que proporciona una lista de directorios desde los que un usuario puede solicitar recibir ficheros; y **remote−send** y **remote−receive**, que se comportan de manera análoga desde un sistema externo. Observe el siguiente ejemplo:

```
system          pablo
...
local-send      /home ~
local-receive   /home ~/receive
remote-send     ~ !~/incoming !~/receive
remote-receive  ~/incoming
```

La orden **local−send** permite a los usuarios de su máquina enviar cualquier fichero bajo `/home` y desde el directorio público de UUCP hacia pablo. La orden **local−receive** les permite entonces recibir tanto en el directorio `receive` de `uucppublic` en el que cualquiera puede escribir, o en cualquier fichero con permisos de escritura universal bajo `/home`. La directiva **remote−send** permite a pablo solicitar ficheros desde `/var/spool/uucppublic`, excepto los de los directorios `incoming` y `receive`. Esto se le señala a **uucico** precediendo los nombres de los directorios con signos de exclamación. Finalmente, la última línea permite a pablo subir ficheros a incoming.

Uno de los mayores problemas con la transferencia de ficheros usando UUCP es que sólo recibe ficheros en los directorios con permiso de escritura universal. Esto puede tentar a algunos usuarios a poner trampas para otros usuarios, etc. Sin embargo, no hay salida a este problema excepto la desactivación total de la transferencia de ficheros por UUCP.

16.3.3. Reenviar

UUCP ofrece un mecanismo para que otros sistemas lleven a cabo transferencias de ficheros por usted. Por ejemplo, suponga que su sistema tiene acceso por **uucp** a un sistema llamado seci, pero no a otro sistema llamado uchile. Esto le permite hacer que seci descague un fichero desde uchile por usted y se lo envíe a su sistema. La siguiente orden lograría esto:

```
$ uucp −r seci!uchile!~/find−ls.gz ~/uchile.files.gz
```

A esta técnica de pasar una tarea a través de muchos sistemas se la conoce como *reenvío*. En su propio sistema UUCP, querrá limitar el servicio de reenvío a unas pocas máquinas en las que confíe para no acabar con una monstruosa factura telefónica tras descargar por ellas las fuentes de la última versión de X11R6.

De manera predeterminada, Taylor UUCP prohibe el reenvío. Para habilitar el reenvío en un sistema en particular, puede usar la orden **forward**. Esta orden especifica una lista de sitios hacia o desde los que el sistema puede solicitarle reenviar tareas. Por ejemplo, el administrador UUCP de seci tendría que añadir las siguientes líneas al fichero `sys` para permitir a pablo solicitar ficheros desde uchile:

```
####################
# pablo
system          pablo
...
forward         uchile
####################
# uchile
system          uchile
...
forward-to      pablo
```

La entrada forward−to para uchile es necesaria para que cualquier fichero devuelto por él se pase realmente a pablo. De otra manera, UUCP se desharía de ellos. Esta entrada usa una variación de la orden **forward** que permite a uchile enviar ficheros sólo a pablo a través de seci y no a la inversa.

Para permitir el reenvío a cualquier sistema, use la palabra clave especial ANY (las mayúsculas son necesarias).

16.4. Configuración de su sistema para recibir llamadas

Si quiere configurar su sitio para recibir llamadas tendrá que permitir conexiones por su puerto serie y personalizar algunos ficheros del sistema para ofrecer cuentas UUCP, lo que cubriremos en esta sección.

16.4.1. Proporcionar cuentas UUCP

A continuación tiene que configurar las cuentas de usuarios que permiten a sistemas remotos entrar en su sistema y establecer una conexión de UUCP. Generalmente tendrá que suministrar un nombre de usuario para cada sistema que se conecte con usted. Cuando configura una cuenta para el sistema pablo, puede darle el nombre de usuario Upablo. No hay ninguna política sobre los nombres de entrada; pueden ser cualquier cosa, pero le convendrá que el nombre de entrada pueda relacionarse fácilmente con el nombre del sitio remoto.

Para los sistemas que se conectan con el suyo a través de puerto serie, normalmente tendrá que añadir estas cuentas al fichero de contraseñas del sistema `/etc/passwd`. Una buena práctica es poner todas las entradas de UUCP en un grupo especial como uuguest. El directorio principal de la cuenta debería configurarse como el directorio público de la cola `/var/spool/uucppublic`; su intérprete de órdenes de entrada debe ser **uucico**.

Para servir a sistemas UUCP que se conecten con su sitio sobre TPC, tendrá que configurar **inetd** de forma que gestione las conexiones entrantes por el puerto uucp añadiendo la siguiente línea a su `/etc/inetd.conf`:[164]

```
uucp   stream  tcp   nowait  root  /usr/sbin/tcpd  /usr/lib/uucp/uucico −l
```

La opción `l` hace que **uucico** lleve a cabo su propia autorización de entrada. Solicitará un nombre de entrada y una contraseña igual que el programa **login** estándar, pero basándose en su propia base de datos privada con las claves en vez de en `/etc/passwd`. Este fichero privado de contraseñas se conoce como `/etc/uucp/passwd` y contiene pares de nombres de entrada y contraseñas:

```
Upablo  IslaNegra
Ulorca  co'rdoba
```

Este fichero tiene que ser propiedad de uucp y tener 600 como permiso.

¿Le parece esta base de datos una idea tan buena que le gustaría en las entradas al sistema por serie también? Bien, en algunos casos puede hacerlo. Lo que necesita es un programa **getty** al que pueda pedirle que invoque a **uucico** en vez de a **/bin/login** para sus usuarios UUCP. [165] La invocación de **uucico** sería de esta forma:

```
    /usr/lib/uucp/uucico −l −u usuario
```

La opción `u` le dice que use el nombre de usuario especificado en vez de preguntarlo.[166]

Para proteger a sus usuarios de UUCP de otros que den un nombre de sistema falso y les lean todo el correo, tiene que añadir órdenes **called−login** a cada entrada de sistema en el fichero `sys`. Esto se describe en la sección siguiente.

16.4.2. Protegerse uno mismo de los estafadores

Uno de los mayores problemas con UUCP es que el sistema que nos llama puede mentir acerca de su nombre; comunica su nombre al sistema que llama después de entrar, pero el servidor no tiene manera de comprobarlo. Por consiguiente, un atacante podría entrar con su propia cuenta de UUCP, pretender ser otra persona, y coger el correo de esa otra persona. Esto representa un grave problema, especialmente si usted ofrece entrada mediante UUCP anónimo, que tiene una clave pública.

Usted *debe* protegerse de estos impostores. La cura de esta enfermedad va a requerir que cada sistema un nombre de entrada en particular especificando un called−login en `sys`. Una entrada de sistema de ejemplo sería algo así:

```
    system          pablo
    ... usual options ...
    called−login    Upablo
```

La ventaja es que cuando un sistema entre y finja ser pablo, **uucico** comprobará si ha entrado como Upablo. De no ser así, se desconectará al sistema que haya llamado. Debería acostumbrarse a añadir la orden **called−login** a cada entrada de sistema que añada a su fichero `sys`. Es importante que haga esto en *todos* los sistemas de su fichero **sys**, independientemente de si van a llamar a su sistema o no. Para los sitios que nunca le llamen, probablemente debería asignar called−login a algún nombre de usuario totalmente ficticio, como nuncaentras.

16.4.3. Sea un paranoico: comprobación de la secuencia de llamadas

Otra manera de detectar y rechazar a los impostores es usar **comprobaciones de la secuencia de llamada**. Éstas le ayudan a protegerse de intrusos que hayan logrado averiguar de alguna manera una contraseña con la que entrar a su sistema UUCP.

Cuando usa comprobación de secuencia de llamadas, ambas máquinas mantienen una cuenta del número de conexiones establecidas hasta el momento. Se incrementa con cada conexión. Después de entrar, el llamador envía su número de secuencia de llamadas y el sistema llamado lo comprueba con su propio número. Si no son iguales, el intento de conexión es rechazado. Si el número inicial se elige aleatoriamente, los atacantes lo tendrán más dificil para adivinar el número de secuencia de llamadas correcto.

Pero la comprobación de la secuencia de llamada sirve para más que esto: aunque una persona muy inteligente descubriese su número de secuencia de llamada así como su clave, usted sabrá que esto ha ocurrido. Cuando el atacante llama al sistema de UUCP que le provee el correo a usted y roba su correo, esto incrementa el número de secuencia de llamada en uno. La siguiente vez que *usted* se conecte con su proveedor de correo e intenta entrar, el **uucico** remoto le rechazará, porque los números de secuencia ya no son iguales.

Si ha activado la comprobación de las secuencias de llamadas, debería mirar sus ficheros de registro de manera regular en busca de mensajes de error que apunten a posibles ataques. Si su sistema rechaza el número de secuencias de llamadas que le ofrece el sistema que llama, **uucico** pondrá un mensaje en el fichero de registro diciendo algo como Out of sequence call rejected. (Llamada fuera de secuencia rechazada). Si su sistema es rechazado por quien le suministra porque los números de la secuencia no están sincronizados, pondrá un mensaje diciendo Handshake failed (RBADSEQ) (Negociación fallida (RBADSEQ)).

Para activar la comprobación de secuencias de llamada, añada la siguiente orden a la entrada del sistema:

```
# activar comprobación de secuencias de llamada
sequence        true
```

Además, tiene que crear el fichero que contiene el número de secuencias en sí. Taylor UUCP guarda el número de secuenciasen un fichero llamado `.Sequence` en el directorio de cola del sitio remoto. *Debe* ser propiedad de uucp y debe estar en modo 600 (esto es, sólo uucp puede leerlo y modificarlo). Lo mejor es inicializar este fichero con un valor que ambas partes hayan acordado previamente. Una manera sencilla de crear este fichero es:

```
# cd /var/spool/uucp/pablo
# echo 94316 > .Sequence
# chmod 600 .Sequence
# chown uucp.uucp .Sequence
```

Evidentemente, el sitio remoto tiene que habilitar igualmente la comprobación de secuencias de llamadas y comenzar usando exactamente el mismo número de secuencia.

16.4.4. UUCP anónimo

Si quiere ofrecer acceso a su sistema por UUCP anónimo primero tendrá que configurar una cuenta especial para ello como se describe anteriormente. Una práctica común es proporcionar a la cuenta anónima uucp como nombre de entrada y contraseña.

Además, tiene que configurar unas pocas opciones de seguridad para sistemas desconocidos. Por ejemplo, puede querer prohibir que ejecuten cualquier orden en su sistema. De todas maneras, no puede ajustar estas variables en una entrada del fichero `sys` porque la orden **system** requiere el nombre del sistema, que usted no tiene. Taylor UUCP resuelve este dilema mediante la orden **unknown**. **unknown** puede usarse en el fichero `config` para especificar cualquier orden que pueda aparecer de manera habitual en una entrada de sistema:

```
unknown         remote-receive ~/incoming
unknown         remote-send ~/pub
unknown         max-remote-debug none
unknown         command-path /usr/lib/uucp/anon-bin
unknown         commands rmail
```

Esto restringirá la descarga de ficheros desde sistemas desconocidos bajo el directorio `pub` y la carga de ficheros en el directorio `incoming` bajo `/var/spool/uucppublic`. La próxima línea hará que **uucico** ignore cualquier petición desde el sistema remoto para activar la depuración localmente. Las dos últimas líneas permiten a sistemas descnocidos ejecutar **rmail**; pero la ruta especificada hace que **uucico** busque la orden **rmail** únicamente en un directoro privado llamado `anon-bin`. Esta restricción le permite ofrecer un **rmail** que, por ejemplo, reenvíe todo el correo al superusuario para que lo examine. Esto permite a los usuarios anónimos ponerse en contacto con el administrador del sistema previniéndoles al mismo tiempo de inyectar correo en otros sitios.

Para habilitar el UUCP anónimo debe especificar al menos una sentencia unknown en `config`. De otra manera **uucico** rechazará todos los sistemas desconocidos.

16.5. Protocolos UUCP de bajo nivel

Para negociar el control de la sesión y las transferencias de ficheros con el sistema remoto, **uucico** usa un grupo de mensajes estándar. Esto es lo que se llama normalmente **protocolo de alto nivel**. Durante la fase de inicialización y la fase de desconexión éstos se envían simplemente como cadenas de caracteres. Sin embargo, durante la fase de transferencia, se usa también un protocolo de bajo nivel, que resulta transparente para los niveles superiores. De esta manera es posible comprobar errores cuando se usan líneas poco fiables, por ejemplo.

16.5.1. Descripción del protocolo

Dado que UUCP se usa sobre diferentes tipos de conexiones, como líneas serie, TCP, o incluso X.25, es preciso usar protocolos de bajo nivel específicos. Además, varias implementaciones de UUCP han introducido diferentes protocolos para hacer lo mismo.

Los protocolos se pueden dividir en dos categorías: de flujo **streaming** y por **paquetes**. La primera clase de protocolos transfiere un fichero entero, posiblemente calculando una suma de comprobación. Esto apenas supone un gasto extra de tiempo, pero precisa una conexión fiable, porque cualquier error causaría que todo el fichero tenga que volver a ser enviado. Estos protocolos se suelen usar sobre conexiones de TCP, pero no sobre líneas telefónicas. Aunque los modems modernos hacen un buen trabajo corrigiendo errores, no son perfectos, y tampoco lo es la detección de errores entre el ordenador y el módem.

Por otra parte, los protocolos por paquetes parten el fichero en varias partes de igual tamaño. Cada paquete se envía y recibe por separado, se realiza una suma de comprobación, y se devuelve al origen un paquete de confirmación. Para que sea más eficiente, se inventaron protocolos de ventanas deslizantes, que permiten un número limitado (una ventana) de paquetes sin esperar confirmación en un momento dado. Esto reduce considerablemente la cantidad de tiempo que **uucico** tiene que esperar durante una transmisión. Aún así, todos los cálculos extra necesarios en comparación a un protocolo de flujo hace que los protocolos de paquetes sean ineficientes sobre TCP pero ideales para las líneas telefónicas.

El caudal del flujo de datos también supone una diferencia. A veces enviar caracteres de 8 bits sobre una conexión serie puedes resultar imposible; por ejemplo, si la conexión atraviesa un estúpido servidor de terminales que se deshace del octavo bit. Cuando transmite caracteres de 8 bits sobre una conexión de 7 bits tienen que codificarse. En el peor caso posible, la codificación duplica la cantidad de datos a transmitir aunque la compresión por hardware pueda compensarlo. Las líneas por las que se pueden transmitir caracteres de 8 bits arbitrarios suelen llamarse *preparadas para 8 bits*. Éste es el caso de todas las conexiones por TCP, así como de la mayoría de las conexiones por módem.

Taylor UUCP 1.06 soporta una amplia variedad de protocolos UUCP. Los más comunes son éstos:

g
: Éste es el protocolo más común y deberían entenderlo prácticamente todos los **uucico**s. Al estar dotado de una potente comprobación de errores resulta especialmente apropiado para conexiones telefónicas con interferencias. *g* requiere una conexión preparada para 8 bits. Es un protocolo orientado a paquetes que usa una técnica de ventana deslizante.

i
: Éste es un protocolo de paquete bidireccionales por el que pueden enviar y recibirse ficheros al mismo tiempo. Requiere una conexión full−duplex y un flujo de datos preparado para 8 bits. Actualmente sólo lo entiende Taylor UUCP.

t
: Este protocolo está pensado para usarse sobre una conexión TCP u otras redes realmente libres de errores. Usa paquetes de 1.024 bytes y requiere una conexión preparada para 8 bits.

e
: Éste debería hacer básicamente lo mismo que *t*. La principal diferencia reside en que *e* es un protocolo de flujo por lo que está orientado únicamente a conexiones de red eficientes.

f

Este protocolo está orientado a conexiones X.25 eficientes. Es un protocolo de flujo y espera un flujo de datos de 7 bits. Los caracters de 8 bits tienen que codificarse, lo que puede hacerlo muy poco eficiente.

G

Ésta es la versión 4 System V del protocolo *g*. También lo entienden otras versiones de UUCP.

a

Este protocolo es similar al ZMODEM. Requiere una conexión de 8 bits pero codifica ciertos caracterse de control como XON y XOFF.

16.5.2. Afinar el protocolo de transmisión

Todos los protocolos permiten alguna variación en el tamaño de los paquetes, el cronómetro y similares. Usualmente, los valores por omisión funcionan bien, pero puede no ser óptimo para su configuración. El protocolo *g*, por ejemplo, usa tamaños de ventanas de 1 a 7, y tamaños de paquetes en potencias de 2 desde 64 a 4096. Si su línea telefónica es tan ruidosa que ignora el 5 por ciento de los paquetes, probablemente debería disminuir el tamaño de los paquetes y de la ventana. Sin embargo, en líneas de teléfono muy buenas el hecho de enviar acuses de recibo por cada 128 bytes puede resultar un desperdicio, así que podría incrementar el tamaño de los paquetes a 512 o incluso 1024. La mayoría de los binarios que se incluyen en las distribuciones de Linux usan de manera predeterminada un tamaño de ventana 7 y paquetes de 128 bytes.

Taylor UUCP le permite ajustar los parámetros con la orden **protocol−parameter** en el fichero `sys`. Por ejemplo, para ajustar el tamaño de paquete a 512 en el protocolo *g* cuando se hable con pablo, tendrá que añadir:

```
system          pablo
...
protocol-parameter g  packet-size  512
```

Los parámetros configurables y sus nombres varian de un protocolo a otro. Para una lista completa de ellos acuda a la documentación que acompaña a las fuentes de Taylor UUCP.

16.5.3. Elegir protocolos específicos

No todas las implementaciones de **uucico** son capaces de comunicarse por medio de todos los protocolos, por lo que durante la fase de negociación inicial ambos procesos tienen que ponerse de acuerdo en la elección de un protocolo común. El **uucico** maestro proporciona al esclavo una lista de protocolos soportados enviándole `P`*`protlist`*, de la cual el esclavo elegirá uno.

Basándose en el tipo de puerto usado (módem, TCP o conexión directa) **uucico** compondrá una lista de protocolos predeterminados. Para la conexión directa o por módem esta lista suele constar de *i*, *a*, *g*, *G* y *j*. Para las conexiones por TCP la lista suele ser *t*, *e*, *i*, *a*, *g*, *G*, *j* y *f*. Puede sobreescribir esta lista predeterminada con la orden **protocols**, que puede especificarse en una entrada de sistema así como en una entrada de puerto. Por ejemplo, puede editar la entrada de su módem en el fichero `port` de esta manera:

```
port            serial1
...
protocols       igG
```

Esto requerirá que cualquier conexión entrante o saliente por este puerto use *i*, *g* o *G*. Si el sistema remoto no soporta ninguno de éstos la negociación fallará.

16.6. Resolución de problemas

En esta sección se describe lo que puede ir mal con su conexión UUCP y se sugieren lugares donde corregir el error. Aunque estos problemas suelen aparecer con frecuencia hay muchas más cosas que pueden fallar de las que hemos listado.

Si tiene algún problema active la depuración con `xall`, y mire la salida de `Debug` en el directorio de cola. Este fichero debería ayudarle a reconocer rápidamente el problema. A menudo resulta de ayuda activar el altavoz del módem cuando no se conecta. Con módems compatibles con Hayes puede activar el altavoz añadiendo `ATL1M1 OK` al diálogo de módem en el fichero `dial`.

La primera comprobación debería ser siempre si todos los permisos de ficheros son los correctos. **uucico** debería ser setuid uucp y todos los ficheros de `/usr/lib/uucp`, `/var/spool/uucp` y `/var/spool/uucppublic` debería tener a uucp como propietario. Hay también algunos ficheros ocultos en el directorio de cola de los que uucp debe ser propietario igualmente.[167]

Cuando esté seguro de que los permisos de todos los ficheros son los correctos y siga teniendo problemas podrá empezar entonces a interpretar los mensajes de error de una manera más literal. Echaremos ahora un vistazo a los problemas y errores más comunes.

16.6.1. uucico sigue diciendo Wrong Time to Call

Esto probablemente signfica que en la entrada de sistema en `sys` no especificó una orden **time** que determina cuándo se puede llamar al sistema remoto o especificó unas horas que en realidad prohíben llamar en ese momento. Si no se especifica cuándo se puede llamar **uucico** asume que nunca se puede llamar al sistema.

16.6.2. uucico se queja de que el sistema ya está en uso

Esto significa que **uucico** detecta un fichero de bloqueo para el sistema remoto en `/var/spool/uucp`. El fichero de bloqueo puede provenir de una llamada anterior al sistema que hubiese fallado o se hubiera interrumpido. Otra posible explicación es que hubiera otro proceso **uucico** intentando llamar al sistema remoto y se hubiese atascado en una macro de diálogo o se hubiese detenido por cualquier otra razón.

Para corregir este error mate todos los procesos **uucico** abiertos para el sitio con una señal hangup y elimine todos los ficheros de bloqueo que hayan podido dejar.

16.6.3. Puede conectar con el sistema remoto pero falla la macro de diálogo

Mire el texto que recibe del sistema remoto. Si está salteado, esto puede ser un problema relacionado con la velocidad. Si no, confirme que realmente envía lo que su macro de diálogo espera recibir. Recuerde, la macro de diálogo empieza con una cadena de caracteres esperada. Si usted recibe la invitación de entrada al sistema (login), después envía su nombre pero luego no se le pregunta por la clave de acceso, inserte un retraso antes de enviarlo, o incluido entre las letras. Puede ser que usted sea demasiado rápido para su módem.

16.6.4. Su módem no marca

Si su módem no indica que la línea DTR se ha levantado al hacer **uucico** una llamada, posiblemente no le ha especificado el dispositivo correcto a **uucico**. Si su módem reconoce DTR, compruebe un programa de terminal que puede enviar órdenes al módem. Si esto funciona, active el eco con la orden \E al comienzo del diálogo del módem. Si el módem no genera el eco de las órdenes durante el diálogo compruebe que la velocidad de su línea no sea demasiado alta o baja. Si ve el eco, compruebe que no haya desacticado la respuesta del módem o la haya configurado como un código numérico. Verifique que la macro de diálogo en sí misma sea válida. Recuerde que tiene que escribir dos barras invertidas para enviar una al módem.

16.6.5. Su módem intenta marcar pero no lo consigue

Inserte una pausa en el número de teléfono, especialmente si tiene que marcar una secuencia especial para obtener el acceso a una línea exterior desde una red telefónica corporativa. Asegúrese de estar usando el tipo de marcado correcto, ya que algunas redes telefónicas sólo soportan un tipo de marcado. De manera adicional, compruebe un par de veces el número de teléfono para asegurarse de que es el correcto.

16.6.6. Se entra con éxito pero falla la negociación

Esta situación puede deberse a diversos problemas. Debería poder obtener bastante información de la salida del fichero de registro. Mire qué protocolos ofrece el sitio remoto (envía una cadena `P` *`protlist`* durante la negociación). Para que la negociación se lleve a cabo con éxito ambas máquinas deben soportar al menos un protocolo común, así que compruebe que efectivamente esto sea así.

Si el sistema remoto envía **RLCK** significa que hay un fichero de bloqueo suyo en el sistema remoto. Si no está conectado a él por otra línea solicite al administrador del sistema remoto que lo elimine.

Si el sistema remoto envía **RBADSEQ**, significa que la comprobación de secuencias de llamada está activada para usted pero los números no coinciden. Si le envía **RLOGIN** es que no le permite entrar bajo esa identidad.

16.7. Ficheros de registro y depuración

Cuando compile UUCP para registrar los eventos a la Taylor, dispondrá de tres ficheros globales únicamente, todos ellos bajo el directorio de cola. El fichero de registro principal es `Log` y contiene toda la información sobre las conexiones establecidas y los ficheros transferidos. Un extracto típico podría ser algo como esto (tras formatearlo un poco para que quede bien en la página):

```
uucico pablo - (1994-05-28 17:15:01.66 539) Calling system pablo (port cua3)
uucico pablo - (1994-05-28 17:15:39.25 539) Login successful
uucico pablo - (1994-05-28 17:15:39.90 539) Handshake successful
               (protocol 'g' packet size 1024 window 7)
uucico pablo postmaster (1994-05-28 17:15:43.65 539) Receiving D.pabloB04aj
uucico pablo postmaster (1994-05-28 17:15:46.51 539) Receiving X.pabloX04ai
uucico pablo postmaster (1994-05-28 17:15:48.91 539) Receiving D.pabloB04at
uucico pablo postmaster (1994-05-28 17:15:51.52 539) Receiving X.pabloX04as
uucico pablo postmaster (1994-05-28 17:15:54.01 539) Receiving D.pabloB04c2
uucico pablo postmaster (1994-05-28 17:15:57.17 539) Receiving X.pabloX04c1
uucico pablo - (1994-05-28 17:15:59.05 539) Protocol 'g' packets: sent 15,
                resent 0, received 32
uucico pablo - (1994-05-28 17:16:02.50 539) Call complete (26 seconds)
uuxqt pablo postmaster (1994-05-28 17:16:11.41 546) Executing X.pabloX04ai
               (rmail okir)
uuxqt pablo postmaster (1994-05-28 17:16:13.30 546) Executing X.pabloX04as
               (rmail okir)
uuxqt pablo postmaster (1994-05-28 17:16:13.51 546) Executing X.pabloX04c1
               (rmail okir)
```

El siguiente fichero de registro importante es `Stats`, que lista estadísticas de transferencia de ficheros. La sección de `Stats` correspondiente a la transferencia anterior sería algo similar a esto (de nuevo, las líneas se han partido para que cuadren en la página):

```
postmaster pablo (1994-05-28 17:15:44.78)
                  received 1714 bytes in 1.802 seconds (951 bytes/sec)
postmaster pablo (1994-05-28 17:15:46.66)
                  received 57 bytes in 0.634 seconds (89 bytes/sec)
postmaster pablo (1994-05-28 17:15:49.91)
                  received 1898 bytes in 1.599 seconds (1186 bytes/sec)
```

```
    postmaster pablo (1994-05-28 17:15:51.67)
                      received 65 bytes in 0.555 seconds (117 bytes/sec)
    postmaster pablo (1994-05-28 17:15:55.71)
                      received 3217 bytes in 2.254 seconds (1427 bytes/sec)
    postmaster pablo (1994-05-28 17:15:57.31)
                      received 65 bytes in 0.590 seconds (110 bytes/sec)
```

El tercer fichero es `Debug`. La información de depuración se escribe aquí. Si usa la depuración asegúrese de que este fichero tenga el modo de protección 600. Dependiendo del modo de depuración que elija puede contener el nombre de usuario y la contraseña que use para conectarse al sistema remoto.

Si dispone de herramientas que esperan que sus ficheros de registro estén en el formato tradicional que usan las implementaciones de UUCP compatibles con HDB, también puede compilar Taylor UUCP de forma que genere registros al estilo HDB. Es simplemente cuestión de activar una opción en tiempo compilación en el fichero `config.h`.

Capítulo 17. Correo Electrónico

El transporte de correo electrónico ha sido uno de los usos más prominentes del trabajo en red desde que las primeras redes de trabajo se diseñaron. Empezó siendo un simple servicio que copiaba un fichero de una máquina a otra, y lo añadía al fichero *mailbox* (buzón de correo) del destinatario. Básicamente, en esto sigue consistiendo el correo electrónico (e−mail), aunque el crecimiento continuo de la red y, consiguientemente, el aumento de la complejidad de encaminado, ha hecho necesario un esquema más elaborado.

Se han diseñado varios estándares de intercambio de correo. Los anfitriones conectados a Internet cumplen uno recogido en el RFC 822, complementado en algunos RFCs que describen un método independiente de la máquina para transferir casi *cualquier cosa*, incluso gráficos, ficheros de sonido y conjuntos de caracteres especiales.[168] El CCITT definió otro estándar, el X.400. Todavía se usa en entornos de grandes corporaciones y gobiernos, pero está siendo retirado progresivamente.

Hay ya una gran cantidad de programas de transporte de correo para sistemas Unix. Uno de los más conocidos es **sendmail**, desarrollado por Eric Allman en la Universidad de California, en Berkeley. Eric Allman ofrece ahora **sendmail** como un producto comercial, pero el programa sigue siendo software libre. **sendmail** se ofrece como el agente de correo estándar en algunas distribuciones de GNU/Linux. Describimos la configuración de **sendmail** en Capítulo 18.

GNU/Linux también usa **Exim**, escrito por Philip Hazel de la Universidad de Cambridge. Describimos la configuración de **Exim**en Capítulo 19.

Comparado con **sendmail**, **Exim** es bastante joven. Para la gran mayoría de los sitios con requerimientos de correo electrónico, sus capacidades son muy parecidas.

Ambos admiten un conjunto de ficheros de configuración que deben ser adaptados a cada caso particular. Aparte de la información que se necesita para hacer funcionar el subsistema de correo (como puede ser el nombre de la computadora local), hay muchos más parámetros que deben ajustarse. El fichero principal de configuración de **sendmail** es muy difícil de entender al principio. Parece como si un gato se hubiese echado una siesta sobre el teclado con la tecla de mayúsculas pulsada. Los ficheros de configuración de **Exim** son más estructurados y son más fáciles de entender que los de **sendmail**, pero no ofrecen soporte directo para UUCP y manejan sólo direcciones de dominio. Hoy esto no es una gran limitación como lo era anteriormente; en cualquier caso, para la mayoría de los sitios, el trabajo requerido en configurar ambos es aproximadamente el mismo.

En este capítulo trataremos sobre qué es el correo electrónico y qué temas tendrá que abordar usted como administrador del sistema. Capítulo 18 y Capítulo 19 darán instrucciones para poner a punto **sendmail** y **Exim** por primera vez. La información que se suministra debe bastar para poner en marcha pequeños anfitriones, pero hay muchas más opciones y usted podrá pasar muchas horas felices frente a su ordenador configurando las características más superficiales.

Hacia el final de este capítulo nos ocuparemos brevemente de cómo poner a punto **elm**, un programa para usuario de correo muy común en muchos sistemas Unix, incluyendo GNU/Linux.

Para más información sobre temas específicos de correo electrónico sobre GNU/Linux, por favor, consulte el 'Electronic Mail HOWTO' de Guylhem Aznar,[169], que aparece en comp.os.linux.answers con regularidad. Las distribuciones fuente de **elm**, **Exim**, y **sendmail** contienen también una documentación muy extensa que debe solucionar la mayoría de sus dudas sobre instalación y puesta a punto. Si busca información sobre correo electrónico en general, hay varios RFCs que tratan específicamente este tema. Una lista de ellos se encuentra en la bibliografía al final del libro.

17.1. ¿Qué es un mensaje de correo?

Un mensaje de correo consta de un cuerpo, que es el texto que ha escrito el remitente, y datos especiales que especifican el destinatario o destinatarios, el medio de transporte, etc., de manera similar a lo que aparece en el sobre de una carta ordinaria.

Estos datos administrativos se clasifican en dos categorías; en la primera categoría están los datos que son específicos del medio de transporte, como son las direcciones del remitente y del destinatario. A esto se le llama el *sobre* (envelope). Puede ser modificado por el software de transporte a medida que el mensaje se transmite.

La segunda categoría es cualquier dato necesario para la manipulación del mensaje, que no es propio de ningún mecanismo de transporte, como es la línea del encabezado en la que indicamos el tema del mensaje (Subject), la lista de todos los destinatarios, y la fecha en la que se envió el mensaje. En muchas redes, se ha convertido en un estándar incluir estos datos al comienzo del mensaje, formando lo que se denomina *cabecera del mensaje* (mail header). Se separa del *cuerpo del mensaje* (mail body) por una línea en blanco.[170]

La mayoría del software para transporte de correo que se emplea en el mundo Unix usa un formato de cabecera definido en el RFC 822. Su propósito original era especificar un estándar para usar en la ARPANET, pero dado que fue diseñado para ser independiente del entorno de uso, ha sido adaptado fácilmente a otras redes, incluyendo muchas basadas en UUCP.

Pero RFC 822 es sólo el mínimo común denominador. Otros estándares más recientes han sido concebidos para dar respuesta a las crecientes necesidades como pueden ser, por ejemplo, encriptación de datos, soporte de conjuntos de caracteres internacionales, y MIME (Multipurpose Internet Mail Extensions, Extensiones de Correo Multipropósito, descritas en el RFC−1341 y otros RFCs).

En todos esos estándares, la cabecera consiste en varias líneas, separadas por caracteres de retorno de carro. Cada línea consiste en un nombre de campo, que comienza en la columna uno, y el campo en sí, separados por dos puntos (:) y un espacio. El formato y la semántica de cada campo varía dependiendo del nombre del mismo. Un campo de al cabecera se puede continuar más allá de una línea, si la línea siguiente comienza con un carácter de espacio, como puede ser un tabulador. Los campos pueden aparecer en cualquier orden.

Una cabecera de correo típico puede ser algo así:

```
Return−Path: <ph10@cus.cam.ac.uk>
Received: ursa.cus.cam.ac.uk (cusexim@ursa.cus.cam.ac.uk [131.111.8.6])
    by al.animats.net (8.9.3/8.9.3/Debian 8.9.3−6) with ESMTP id
WAA04654
    for <terry@animats.net>; Sun, 30 Jan 2000 22:30:01 +1100
Received: from ph10 (helo=localhost) by ursa.cus.cam.ac.uk with
local−smtp
    (Exim 3.13 #1) id 12EsYC−0001eF−00; Sun, 30 Jan 2000 11:29:52 +0000
Date: Sun, 30 Jan 2000 11:29:52 +0000 (GMT)
From: Philip Hazel <ph10@cus.cam.ac.uk>
Reply−To: Philip Hazel <ph10@cus.cam.ac.uk>
To: Terry Dawson <terry@animats.net>, Andy Oram
<andyo@oreilly.com>
Subject: Electronic mail chapter
In−Reply−To: <38921283.A58948F2@animats.net>
Message−ID:
<Pine.SOL.3.96.1000130111515.5800A−200000@ursa.cus.cam.ac.uk>
```

Usualmente, todos los campos de la cabecera necesarios son generados por el interfaz de correo que usted use, como **elm**, **pine**, **mush**, o **mailx**. Algunos, sin embargo, son opcionales y pueden ser añadidos por el usuario. **elm**, por ejemplo, permite editar parte del encabezado del mensaje. Otros campos son añadidos por el software de transporte de correo. Si usted mira el fichero donde se almacena el correo local, puede ver que cada mensaje está precedido por una línea From (nota: sin dos puntos). Esta *no* es una cabecera RFC−822; ha sido insertada por su software de correo para facilitar la lectura a los programas que usen ese fichero. Para prevenir potenciales problemas con las líneas del cuerpo del mensaje que también empiecen por From, se ha convertido en un procedimiento estándar evitar estas ocurrencias poniendo antes un carácter >.

Esta lista es una colección de cabeceras de campos comunes, y sus significados:

`From:`
: Contiene la dirección de correo electrónico del remitente, y posiblemente el nombre real. Aquí se usa un zoológico completo de formatos distintos.

`To:`
: Esta es la dirección de e−mail del destinatario. Si hay varias direcciones se separan por comas.

`Cc:`
: Esta es una lista de las direcciones de correo que recibirán una copia de carbón del mensaje. Si hay varias direcciones, se separan por comas.

`Bcc:`
: Esta es una lista de las direcciones de correo que recibirán una copia de carbón ciega del mensaje. La diferencia principal entre Cc: y Bcc: es que las direcciones listadas en el Bcc: no aparecerán en la cabecera del mensaje que se envía a cada destinatario. Es una forma de avisar a los destinatarios de que usted ha enviado copias del mensaje a otras personas, sin decir quiénes son. Si hay varias direcciones, se separan por comas.

`Subject:`
: Describe el contenido del mensaje en pocas palabras.

`Date:`
: Indica la fecha y hora en que se envió el mensaje.

`Reply−To:`
: Especifica la dirección a la que el remitente desea que el destinatario le conteste. Esto puede ser útil si se tienen varias direcciones, pero se desea recibir la mayor parte del correo sólo en aquella que se usa más a menudo. Este campo es opcional.

`Organization:`
: La organización que posee la máquina desde la que se ha enviado el mensaje. Si la máquina usada es la suya propia no incluya este campo, o bien indique privado o cualquier trivialidad sin sentido. Este campo no está descrito en ningún RFC y es completamente opcional. Algunos programas de correo lo soportan directamente, pero la mayoría no.

`Message−ID:`
: Una cadena generada por el transporte de correo en el sistema remitente. Es única para cada mensaje.

`Received:`
: Cada anfitrión que procesa su correo (incluyendo las máquinas del remitente y el destinatario) insertan este campo en el encabezado, dando el nombre del anfitrión, una identificación de mensaje, hora y fecha a la que lo recibieron, de qué anfitrión procede, y qué software de transporte ha sido usado. Esto se hace así para que usted pueda conocer la ruta que su mensaje ha seguido, y pueda protestar a la persona responsable si algo ha ido mal.

`X−cualquier−cosa:`
: Ningún programa relacionado con el correo debe protestar sobre cualquier encabezado que comience con `X−`. Esto se usa para implementar características adicionales que aún no han sido incluidas en un

RFC, o que no lo serán nunca. Por ejemplo, existió un gran servidor de listas de correo de GNU/Linux que permitía especificar a qué canal quería que fuera su mensaje incluyendo la cadena **`X-Mn-Key:`** seguido del nombre del canal.

17.2. ¿Cómo se reparte el correo?

Generalmente, usted escribirá su correo usando un interfaz de correo como **mail** o **mailx** u otros más sofisticados como **mutt**, **tkrat**, o **pine**. Estos programas se denominan *agentes de usuario de correo* (mail user agents), o MUAs para abreviar. Si usted envía un mensaje de correo, el programa interfaz en la mayoría de los casos se lo pasara a otro programa para que lo transmita. Este programa se denomina *agente de transporte de correo* (mail transport agent), o MTA. En la mayoría de los sistemas se usa el mismo MTA tanto para el reparto local como remoto, y normalmente se invoca como **/usr/sbin/sendmail** o, en algunos sistemas que no cumplen la norma FSSTND, **/usr/lib/sendmail**. En sistemas UUCP no es raro ver que el correo se reparte por dos programas distintos: **rmail** para el envío remoto de correo, y **lmail** para el reparto local.

Un envío local de correo es, por supuesto, algo más que añadir el mensaje al buzón del destinatario. Usualmente el MTA local entenderá como usar alias (definir direcciones locales de destinatarios que dirigen a otras direcciones) y como usar redirecciones (dirigir el correo de un usuario a otra dirección). Además, los mensajes que no pudieron ser enviados deben ser normalmente *devueltos*(bounced) al remitente junto con algún mensaje de error.

Para envíos lejanos, el software de transporte usado depende del tipo de enlace. Si el correo debe enviarse a través de una red que usa TCP/IP, se usará normalmente *Protocolo Simple de Trasferencia de Correo* (SMTP), que se define en el RFC 821. SMTP se diseñó para repartir correo directamente en la máquina de un destinatario, negociando la transferencia del mensaje con el demonio SMTP del lado remoto. Hoy es práctica común de las organizaciones establecer máquinas especiales que aceptan todo el correo para destinatarios de la organización, y estos anfitriones se encargan de controlar el reparto apropiado a los destinos adecuados.

En redes tipo UUCP, el correo no suele ser enviado directamente, sino que es redirigido hasta su destino a través de un conjunto de máquinas intermedias. Para enviar un mensaje a través de un enlace UUCP, el MTA remitente ejecutará usualmente **rmail** en la máquina intermedia usando **uux**, y suministrándole el mensaje en la entrada estándar.

Desde que se llama a **uux** separadamente para cada mensaje, puede producirse una carga considerable en un anfitrión procesador de correo, además de inundar las colas UUCP con cientos de pequeños mensajes que ocupan una cantidad de disco desproporcionada.[171] Por esto algunos MTAs permiten recopilar varios mensajes de un sistema remoto en un único lote. El fichero de lotes contiene los comandos SMTP que el anfitrión local ejecutaría normalmente si usara una conexión SMTP directa. A esto se le llama BSMTP, o *batched* SMTP (SMTP por lotes). El lote se suministra a los programas **rsmtp** o **bsmtp** en el sistema remoto, que procesará la entrada como si se hubiera dado una conexión SMTP normal.

17.3. Direcciones de correo electrónico

Las direcciones de correo electrónico constan de dos partes: la primera es el nombre de un *dominio de correo* encargada de traducir la información, o bien al anfitrión del receptor o a cualquiera que acepte correo de su parte. La segunda es la identificación exclusiva del usuario que puede ser tanto el nombre que le permite el acceso al sistema, como el nombre del usuario en formato nombre.apellido , o un alias que se transmitirá a un usuario o a la lista de usuarios. Otros formatos de dirección de correo como el X.400 utilizan otra serie de atributos que sirven para localizar al sistema anfitrión del receptor en el directorio del servidor X.400.

La interpretación de las direcciones de correo electrónico depende en gran medida del tipo de red de la que usted disponga. A hora nos centraremos en cómo los protocolos TCP/IP y UUCP interpretan las direcciones de correo electrónico.

17.3.1. RFC−822

Los sitios de Internet están ligados al estándar RFC−822, que requiere la conocida notación usuario@anfitrión.dominiopara el cual anfitrión.dominio es el nombre de dominio más adecuado para el anfitrión. El símbolo que separa ambas partes recibe el nombre de arroba en inglés at. Esta nomenclatura no especifica la ruta al sistema anfitrión. Otros mecanismos que trataremos en breve son los encargados del encaminamiento de los mensajes.

Al ejecutar un sitio en Internet encontrarás muchos RFC−822. EL estándar RFC−822 no es exclusivo del correo, ya que se ha extendido a otros servicios como los grupos de noticias. Ahora veremos el uso de RFC−822 en los grupos de noticias. Capítulo 20.

17.3.2. Formatos de dirección de correo obsoletos

En el entorno original UUCP la forma corriente era ruta!anfitrión!usuario, para el cual la ruta indicaba una secuencia de anfitriones por los que el mensaje tenía que pasar antes de llegar a su anfitrión de destino. Este modelo recibe el nombre de notación *bang path*, porque en inglés coloquial, la exclamación se conoce con el nombre bang. Actualmente, muchas de las redes basadas en UUCP han adoptado el formato RFC−822 y aceptan las direcciones de correo basadas en el dominio.

Hay redes que usan otros sistemas de dirección. Por ejemplo, las basadas en DECnet, usan los dos puntos (:) como elemento separador de sus partes, resultando la dirección anfitrión::usuario.[172] El estándar X.400 utiliza un estilo totalmente diferente, describiendo al receptor por medio de pares de atributos como país y organización a la que éste pertenece.

En último lugar, está FidoNet, en donde cada usuario se identifica con un código como 2:320/204.9, que consiste en cuatro números que indican la zona donde se encuentra (el 2 es para Europa), la red (el 320 se refiere a París y los alrededores), el anfitrión (distribuidor local), y el punto de conexión (el ordenador del usuario). Se puede trabajar con direcciones Fidonet en RFC−822; la anterior, por ejemplo, se escribiría de la siguiente manera Thomas.Quinot@p9.f204.n320.z2.fidonet.org. No dijimos que los nombres de dominio eran fáciles de recordar?

17.3.3. Cómo combinar distintos formatos de correo electrónico

Cuando a un conjunto de sistemas le sumamos gente inteligente, lo normal es que se intente buscar maneras para poder conectarse entre sí y trabajar en red. Por consiguiente, existen distintas pasarelas de correo que vinculan dos sistemas diferentes, de forma que el correo pueda ser transmitido de uno a otro. El problema más crítico a la hora de interconectar dos sistemas es el referente a las direcciones de correo. No vamos a centrarnos en las pasarelas de correo en sí mismas, sino que repasaremos algunas de las complicaciones referidas al correo que pueden surgir al usarlas.

Imagine que queremos trabajar con la notación bang−path de UUCP y RFC−822. No son dos formatos fáciles de combinar. Supongamos que tenemos la siguiente dirección dominioA!usuario@dominioB . No está claro si el símbolo @ tiene prioridad sobre la ruta, o viceversa: entonces, ¿tendríamos que mandar el mensaje a dominioB, que lo enviaría a dominioA!usuario, o por el contrario deberíamos hacerlo a dominioA, que lo dirigiría a usuario@dominioB?

Aquellas direcciones formadas por distintos proveedores de correo se llaman *direcciones híbridas*. El tipo más común, que es el que acabamos de ilustrar, normalmente se resuelve dando prioridad al símbolo @ sobre la ruta. Esto significaría enviar primero el mensaje a dominioB en dominioA!usuario@dominioB

Sin embargo, hay una manera de especificar las ruta en RFC−822: <@dominioA,@dominioB:usuario@dominioC> indica la dirección delusuario en el dominioC, donde llegamos al dominioCpasando por dominioA y dominioB (en ese orden). Este tipo de dirección se llama con frecuencia dirección*encaminada desde la fuente* Tampoco es bueno basarse exclusivamente en este sistema, ya que un posterior repaso al estándar RFC en su apartado de la descripción del encaminamiento del correo, recomienda que se intente enviar el correo directamente a su destino remoto en lugar de hacerlo por medio de la fuente.

Existe además el % proveedor de correousuario%dominioB@dominioA que lo envía primero a dominioA, y transforma el símbolo de porcentaje más indicado (que en este caso es el único) a un símbolo arroba@ sign. La direccion en este caso es usuario@dominioB, y el mensajero dirige su mensaje a dominioB, el cual lo envía al usuario. A este tipo de dirección la solemos llamar Ye Olde ARPAnet Kludge, y su uso no es alentador.

El uso de estos tipos distintos de direccionamiento puede tener sus repercusiones, las cuales veremos en las secciones siguientes. En un entorno RFC−822 no se deberá usar otra dirección que no sea una absoluta como usuario@anfitrión.dominio.

17.4. ¿Cómo funciona el encaminamiento del correo?

Conocemos como *encaminamiento*el proceso de dirigir un mensaje al sistema anfitrión del receptor. Aparte de localizar una ruta desde el sitio del emisor hasta el de destino, la elección de rutas implica la detección de errores e incluso la optimización de la velocidad y el coste.

Hay una gran diferencia entre la elección de rutas por parte de un sitio UUCP y un sitio de Internet. En Internet, la función principal a la hora de dirigir los datos al anfitrión del receptor (cuando el protocolo IP lo conoce), la realiza la capa de red IP, mientras que en el entorno UUCP, la ruta tiene que ser provista por el usuario o generado por el agente de transmisión de correo.

17.4.1. Encaminamiento en Internet

La configuración del sistema anfitrión del destinatario determina si se está trabajando con un determinado sistema de localización de la ruta de correo en Internet. La opción predeterminada es transmitir el mensaje a su destino determinando primero el anfitrión al que debe ser enviado, y mandándolo allí directamente. La mayoría de los sitios de Internet buscan dirigir todo el correo entrante a un servidor de correo con alta disponibilidad capaz de manejar todo el tráfico y distribuirlo localmente. Para dar a conocer este servicio, el sitio publica el llamado registro MX para su dominio local en su base de datos DNS. MX (Mail Exchanger) significa *Intercambiador de Correo* y básicamente indica que el anfitrión del servidor es capaz de convertirse en emisor de correo para todas las direcciones del dominio. Los registros MX también pueden manejar el tráfico de anfitriones que no están conectados a la red, como UCCP o FidoNet, que necesitan una pasarela de correo.

Los registros MX siempre tienen siempre asignada una *preferencia* Esto es positivo. Si son muchos los proveedores de correo existentes (MX) para un anfitrión, el agente de transporte de correo tratará de enviar el mensaje al proveedor cuya preferencia sea la menor. Sólo si esta operación falla, lo enviará a un anfitrión de mayor índice de preferencia. Si el anfitrión local es el proveedor de correo para la dirección de destino, puede enviar los mensajes a un anfitrión menos preferente que él mismo; ésta es una manera segura de evitar los bucles en el correo. Si no hay ningún registro MX para un determinado dominio, o no es disponible, el agente de transporte de correo puede comprobar si la dirección IP del dominio está asociada a él, y así intentar mandarlo directamente a ese anfitrión.

Supongamos que hay una organización dada, por ejemplo Foobar, Inc., que quiere que todo su correo lo controle el servidor de correode su ordenador. Por ello llevará registros MX como el que se muestra a continuación, en la base de datos DNS:

```
    green.foobar.com.         IN    MX       5     mailhub.foobar.com.
```

Esto da a conocer a mailhub.foobar.com como proveedor de correo para green.foobar.comcon un nivel de preferencia de 5. Un anfitrión que pretenda enviar un mensaje a joe@green.foobar.com revisa la base de datos DNS y busca el MX en el distribuidor de correo . Si no hay ningún MX con una preferencia menor a 5, el mensaje se envía al distribuidor de correo, que lo entrega a green.

Ésta es una descripción muy básica de cómo funcionan los registros MX. Para obtener más información sobre el encaminamiento de correo en Internet, consulte RFC−821, RFC−974 y RFC−1123.

17.4.2. Encaminamiento en el entorno UUCP

El encaminamiento del correo en las redes UUCP es mucho más complicado que en Internet porque sus programas no realizan el encaminamiento ellos mismos. Antes, todo el correo debía ser dirigido mediante las rutas bang path. Éstas especificaban una lista de sistemas anfitriones separados por signos de exclamación y seguidos por el nombre del usuario, por los que el correo debía pasar. Por ejemplo, para escribir a un usuario llamado Janet que se encuentra en un ordenador llamado moria, usuaríamos la ruta eek!swim!moria!janet.De esta manera el mensaje se enviaría desde su sistema anfitrión a eek,desde aquí a swim, y por último lugar a moria.

El inconveniente obvio de este sistema es que es necesario que el usuario recuerde muchos más datos sobre topología de red que la que Internet requiere. Y mucho peor son los cambios de la topología como los enlaces eliminados o anfitriones que desaparecen que produce fallos en los mensajes al no ser el usuario consciente de estos cambios. Y por último, si usted cambia de sitio o se traslada, probablemente deberá actualizar estas rutas.

Una razón por la que el encaminamiento desde la fuente se hizo necesario fue la presencia de nombres de anfitrión ambiguos. Por ejemplo, imaginemos que hay dos sitios llamados moria, uno en los Estados Unidos y otro en Francia. ¿A cuál de los dos se referiría moria!janet ahora? El problema quedaría solucionado especificando una ruta concreta para acceder a moria

El primer paso para evitar la ambigüedad con los nombres de anfitrión fue el proyecto de mapeado UUCP. Se encuentra en la Universidad de Rutgers y registra de manera oficial todos los nombres de anfitrión, junto con información sobre otros sistemas UUCP y su situación geográfica, procurando que no se repita ninguno. Esta información en manos del proyecto de mapeado UUCP, se publica bajo el nombre *Mapas Usenet* , y son distribuidos regularmente a través de Usenet. El formato típico de entrada a un mapa (eliminados ya los comentarios) es de la siguiente manera:[173]

```
    moria
            bert(DAILY/2),
            swim(WEEKLY)
```

Esta entrada indica que moria está vinculado a bert, al cual llama dos veces al día, y a swim, al cual llama semanalmente. Explicaremos con más detalle lo referente al formato de fichero de mapas.

Con la información sobre la conectividad que obtenemos de los mapas, podemos generar la totalidad de rutas existentes entre su sistema anfitrión y cualquier sitio. Esta información se encuentra en el fichero de `rutas`, también conocido como *base ruta−alias*. Supongamos que los mapas indican que usted puede ponerse en contacto con bert a través deernie; una entrada en forma de alias de ruta para moria generado del retazo del mapa anterior podría ser de la siguiente manera:

```
    moria          ernie!bert!moria!%s
```

Si usted propone la dirección janet@moria.uucp, el MTA seguirá la ruta anterior y enviará el mensaje a, ernie con la direcciónbert!moria!janet.

No obstante, crear un fichero de `rutas` a partir de los mapas Usenet no es buena idea. La información que contienen suele estar distorsionada, y también es posible que no esté actualizada. Es por ello que sólo un determinado número de anfitriones utilizan los mapas UUCP completos para crear sus ficheros de `rutas`. Muchos sitios mantienen la información de ruta sólo para sitios que se encuentran en su entorno, y envían cualquier mensaje a los sitios que no están presentes en su base de datos a anfitriones más inteligentes con información de ruta más completa. Este esquema se llama *encaminamiento por anfitrión inteligente*. Los anfitriones que tienen sólo un vínculo de correo UUCP (los llamados *leaf sites*), no pueden realizar el encaminamiento por su cuenta, deben dejar esa labor a un anfitrión inteligente.

17.4.3. Mezclar UUCP y RFC−822

La mejor manera de evitar los problemas referentes al encaminamiento del correo en las redes UUCP es adoptar el sistema de nombre de dominio de dichas redes. Por supuesto, usted no puede cuestionar un servidor de nombres de UUCP. Sin embargo, muchos sitios UUCP han creado pequeños dominios que coordinan su encaminamiento internamente. En los mapas, estos dominios anuncian uno o dos anfitriones en forma de pasarela de correo propio de tal forma que no tiene que existir un indicador de entrada al mapa para cada anfitrión en el dominio. Las pasarelas de correo controlan tanto el fluido de correo interno como externo al dominio. El plan de encaminamiento dentro del dominio es independiente e invisible para el mundo exterior.

Esto funciona muy bien en el esquema de encaminamiento por anfitrión inteligente. El encaminamiento global de la información sólo lo mantienen los portales; los anfitriones menores dentro de un dominio pueden trabajar únicamente con ficheros de `rutas` pequeños, escritos a mano que indiquen las rutas de ese dominio y el camino hacia el encaminador. Incluso las pasarelas de correo ya no necesitan la información de ruta para cada anfitrión UUCP del mundo. Aparte de la información de ruta, ahora tan sólo necesitan conocer rutas hacia dominios absolutos. Por ejemplo, esta entrada ruta−alias conducirá todo el correo hacia los sitios en el dominio sub.org hacia smurf:

```
    .sub.org         swim!smurf!%s
```

Todo el correo enviado a claire@jones.sub.org será enviado a swim con la dirección smurf!jones!claire.

La organización jerárquica del nombre de dominio permite a los servidores de correo mezclar rutas más y menos específicas. Por ejemplo, un sistema francés puede tener rutas específicas para los subdominios en ofr, y encaminar el correo hacia los anfitriones en el dominio, us en algún sistema de los Estados Unidos. De esta manera, gracias al encaminamiento basado en el dominio (nombre que recibe esta técnica) tanto el tamaño de las bases de datos de encaminamiento como las necesidades administrativas, se ven reducidos.

La ventaja principal al usar nombres de dominio en un entorno UUCP es que las normas de conformidad con RFC−822 permiten el contacto entre las redes UUCP en Internet. Actualmente, muchos dominios UUCP tienen vínculos con pasarelas de Internet que actúan como anfitrión. Es más rápida y más fiable la información de encaminamiento si mandamos los mensajes por Internet, ya que éstos anfitriones pueden funcionar con DNS en lugar de Mapas Usenet.

Con el fin de se ser localizados desde Internet, los dominios basados en UUCP muestran un registro MX (los registros MX se comentaron en la secciónSección 17.4.1). Por ejemplo, supongamos que moria pertenece al dominio orcnet.org gcc2.groucho.edu actúa como su pasarela a Internet. Entonces moria utilizaría gcc2 como anfitrión, para que toda la correspondencia dirigida a dominios extranjeros se distribuyese a través de Internet. Por otro lado, gcc2 mostraría un registro MX para *.orcnet.org y llevaría todo el correo entrante para los sitios orcnet a moria. El asterisco en *.orcnet.org es un comodín que empareja todos los anfitriones de ese dominio que no están relacionados con ningún registro. Esto ocurre con frecuencia sólo con los dominios UUCP.

El único problema que queda es que los programas de transmisión UUCP no pueden funcionar con nombres de dominio ilimitados. Muchos sitios UUCP fueron diseñados para trabajar con nombres de hasta ocho caracteres, o incluso menos, y sin utilizar caracteres alfanuméricos como el punto.

Por lo tanto, habría que hacer un mapeado entre los nombres RFC−822 y los nombres de anfitrión UUCP. El mapeado depende totalmente de su puesta en práctica. Una manera común de mapear los nombres FQDN y los UUCP, es usar el fichero del alias de ruta:

```
moria.orcnet.org  ernie!bert!moria!%s
```

Esto producirá un bang path al estilo UUCP desde una dirección que especifique un nombre de dominio completamente cualificado. algunos agentesd de transporte proporcionan un fichero especial para esto: **sendmail**, por ejemplo, usa el fichero `uucpxtable`.

La transformación inversa (conocida coloquialmente como *domainizing*) a veces es necesaria cuando se envía un mensaje desde una red UUCP a Internet. Mientras el emisor utilice el nombre de dominio completo en la dirección de destino, este problema se puede evitar si no eliminamos dicho nombre de dominio. Sin embargo, hay sitios UUCP que no pertenecen a ningún dominio. Normalmente llevan el pseudo−dominio uucp.

La base de datos ruta−alias proporciona la principal información de ruta en las redes basadas en UUCP. La entrada es de esta manera (el nombre del sitio y la ruta están separados mediante tabulaciones):

```
moria.orcnet.org  ernie!bert!moria!%s
moria             ernie!bert!moria!%s
```

Esto hace que cualquier mensaje enviado a moria sea entregado pasando por ernie y bert. Tanto el nombre moriacomo el nombre UUCP deben ser dados si el emisor no los incluye.

Si usted quiere dirigir todos los mensajes a los anfitriones dentro de un dominio a su repetidor de correo, puede especificar una ruta en la base de datos del alias de ruta, indicando el nombre de dominio precedido por un punto como el destino. Por ejemplo, si a todos los anfitriones en sub.org llegamos por medio de swim!smurf, la entrada de alias de ruta podrías ser de la siguiente manera:

```
.sub.org        swim!smurf!%s
```

Escribir el fichero de alias de ruta es aceptable sólo cuando accede a un sitio de Internet donde no son necesarias muchas operaciones de encaminamiento. Si tiene que realizar diversas operaciones de encaminamiento para un gran número de anfitriones, la mejor manera de hacerlo es usar la orden **alias de ruta** para crear el fichero a partir del fichero de mapas. Los mapas son más fáciles de mantener, porque se añade o elimina un sistema editando la entrada al mapa del sistema y volviendo a crear el fichero de mapa. Aunque los mapas publicados por el Proyecto de Mapeado Usenet ya no se usan tanto para el encaminamiento, las redes pequeñas UUCP nos pueden dar la información sobre el encaminamiento de sus propios mapas.

Un fichero de mapa consiste principalmente en una lista de sitios que cada sistema selecciona, o bien seleccionada por algún sistema. El nombre del sistema empieza en la primera columna y va seguido por una lista de enlaces separados por una coma. La lista puede continuar si la siguiente línea comienza por el tabulador. Cada vínculo consiste en el nombre del sitio seguido por un coste entre paréntesis. El coste es una expresión aritmética formada por números y expresiones simbólicas como DAILY o WEEKLY. Las líneas que empiezan por > se ignoran.

Por ejemplo, consideremos moria, que selecciona swim.twobirds.com dos veces al día y bert.sesame.com que lo hace una por semana. El vínculo a bert usa modem lento a 2.400 bps. moria publicaría la siguiente entrada:

```
moria.orcnet.org
        bert.sesame.com(DAILY/2),
        swim.twobirds.com(WEEKLY+LOW)
moria.orcnet.org = moria
```

La última línea también da a conocer a moria bajo su nombre UUCP. Tenga en cuenta que el coste se debe especificar como DAILY/2 porque conectando dos veces al día limita a la mitad el coste del vínculo

Al usar la información de los ficheros de mapas **pathalias** es capaz de calcular las rutas óptimas a cualquier destino indicado en el fichero de ruta y producir una base de datos ruta−alias con la que realizar el encaminamiento a estos sitios.

alias de ruta proporciona otras opciones como el ocultamiento del sitio (es decir, que sólo se pueda llegar a los sitios a través de una pasarela). Consulte la página sobre **alias de ruta** del manual para obtener detalles y una lista completa de vínculos cost.

Los comentarios sobre el fichero de mapas suelen contener información adicional sobre los sitios descritos en él. Existe un formato rígido en el que se puede especificar esta información de tal forma que se pueda recuperar a partir de los mapas. Por ejemplo, un programa llamado **uuwho** utiliza una base de datos creada a partir de los ficheros de mapa para mostrar tal información de manera cómoda. Por ello, si usted contrata un sitio con una organización que distribuye ficheros de mapas, deberá rellenar dicha entrada. A continuación se muestra un ejemplo de entrada de mapa (es la perteneciente al sitio web de Olaf):

```
#N      monad, monad.swb.de, monad.swb.sub.org
#S      AT 486DX50; Linux 0.99
#O      private
#C      Olaf Kirch
#E      okir@monad.swb.de
#P      Kattreinstr. 38, D-64295 Darmstadt, FRG
#L      49 52 03 N / 08 38 40 E
#U      brewhq
#W      okir@monad.swb.de (Olaf Kirch); Sun Jul 25 16:59:32 MET DST
1993
#
monad   brewhq(DAILY/2)
# Domains
monad = monad.swb.de
monad = monad.swb.sub.org
```

El espacio en blanco que sigue a los dos primeros caracteres equivale a una tabulación. El significado de la mayoría de los campos está bastante claro; de todas maneras, en caso de registrarse en cualquier dominio, recibiría dicha descripción detallada. El caso de la L es el más curioso: proporciona la posición geográfica (latitud/longitud) del usuario y se encarga de dibujar los mapas PostScript que controlan todos los sitios web de cada país e incluso de toda la red.[174]

17.5. como configurar elm

elm significa electronic mail (correo electrónico), y es una de las herramientas más importantes de Unix. Proporciona una interfaz a pantalla completa que incluye elementos de ayuda muy útiles. No nos detendremos más en cómo usar el mencionado **elm** sino que trataremos las opciones de configuración.

Teóricamente, se puede ejecutar**elm** que no esté configurado, y todo funcionará correctamentecon suerte. Sin embargo hay algunas opciones que hay que configurar, aunque serán requeridas tan solo en ocasiones.

Cuando se inicia**elm**, tenemos un conjunto de variables de configuración del fichero`elm.rc` en `/etc/elm`. El fichero `.elm/elmrc` aparece en su directorio local. Usted no tendrá que elaborar este fichero, ya que se crea al seleccionar Save new options en el menú opciones de **elm**

Las opciones correspondientes al fichero `elmrc` están también disponibles en el fichero global `elm.rc`. Muchos elementos de la configuración del fichero`elmrc` personal, invalidan los del global.

17.5.1. opciones globales de elm

En el fichero global `elm.rc`, siempre hay que establecer las opciones correspondientes al nombre del anfitrión. Por ejemplo, en la Cervecera Virtual, el fichero vlager contiene la siguiente información:

```
#
# Nombre local del anfitrión
hostname = vlager
```

```
#
# Nombre del dominio
hostdomain = .vbrew.com
#
# Nombre completamente cualificado del dominio
hostfullname = vlager.vbrew.com
```

Estas opciones de **elm** están ideadas por el nombre del anfitrión local. Cierto es que esta información tal vez no se llegue a usar nunca, pero de todas maneras hay que establecer las opciones. Tenga en cuenta que estas opciones privadas sólo serán válidas en el fichero de configuración; cuando se trate del `elmrc` privado, serán ignoradas.

17.5.2. Juegos de caracteres Nacionales

Se han desarrollado un conjunto de estándares y RFC que preparan al RFC−822 para poder recibir varios tipos de mensajes, tanto de texto, datos binarios, como ficheros PostScript y demás. Estos estándares son conocidos como MIME o Multipurpose Internet Mail Extensions (Extensiones Multipropósito del Correo Internet). Entre otras cosas, MIME permite que el receptor del mensaje sepa si se han utilizado caracteres distintos a los del estándar ASCII, por ejemplo, los acentos del francés o la diéresis del alemán. **elm** mantiene este juego de caracteres hasta cierto punto.

El juego de caracteres interno de GNU/Linux que se usa para representar los caracteres es conocido como ISO−8859−1, que es el nombre del estándar correspondiente. También se conoce como Latin−1. Cualquier mensaje que utilice cualquier carácter específico de este juego de caracteres, deberá llevar el siguiente encabezado:

```
Content-Type: text/plain; charset=iso-8859-1
```

El sistema receptor del mensaje deberá reconocer ese campo y adaptar las medidas adecuadas para que su visualización sea la correspondiente. En cuanto a los mensajes de texto lo correspondiente es el juego de caracteres us−ascii.

Para ser capaz de mostrar mensajes con conjuntos de caracteres distintos al código ASCII, **elm** debe saber cómo imprimir tales caracteres. Por omisión, cuando **elm** recibe un mensaje con un campo *`charset`* distinto a us−ascii (o con contenido distinto al tipo texto), intenta mostrar el mensaje usando una orden llamada **metamail**. Los mensajes que requieren de **metamail**, para poder ser visualizados, se muestran con una **M** en la primera columna en la pantalla de resumen.

Como el caracter patrón de Linux es ISO−8859−1, no es necesario el uso de **metamail** cuando sea éste el juego de caracteres presente. Si **elm** tiene la orden de que el sistema puede leer ISO−8859−1, no usará **metamail**, sino que en su lugar mostrará el mensaje directamente. Esto es posible si se configura la siguiente opción en `elm.rc`:

```
displaycharset = iso-8859-1
```

Tenga en cuenta que se debe configurar esta opción incluso cuando no tenga por seguro que va a mandar o recibir mensajes con caracteres distintos a los de ASCII. La razón es porque los usuarios que sí los utilizan, normalmente configuran el correo con el fin de poner el `tipo de contenido:` de campo apropiado en la cabecera, aunque no utilizen sólo caracteres ASCII.

Sin embargo, activar esta opción en `elm.rc` no es suficiente. Cuando se muestra un mensaje con su paginador incorporado, **elm** llama a una funcion de biblioteca para cada carácter con el fin de determinar si es imprimible. Por omisión, esta función sólo reconocerá los caracteres ASCII como imprimibles y visualizará el resto como `^?`. Puede pasar por alto esta función tan sólo con indicar la variable de entorno LC_CTYPE con ISO−8859−1, que le indica a la biblioteca que acepte los caracteres Latin−1 como imprimibles. el soporte para esta y otras características está disponible desde la versión 4.5.8 de la biblioteca estándar de Linux.

Cuando envía mensajes que contienen caracteres especiales de ISO−8859−1, debe asegurarse de poner dos variables más en el fichero `elm.rc` :

```
    charset = iso-8859-1
    textencoding = 8bit
```

Esto hace que **elm** reporte el conjunto de caracteres como ISO−8859−1 en la cabecera del correo, y lo envía como un valor de 8 bits (la opción predeterminada es cortar los caracteres a 7 bits).

También se pueden configurar todas estas opciones sobre los caracteres que hemos tratado en el fichero privado `elmrc`, de tal forma que cada usuario puede tener sus propia configuración predeterminada aunque el fichero global no se adapte.

Capítulo 18. Sendmail

18.1. Introducción a sendmail

Se dice que no se es un *verdadero* administrador de sistemas Unix hasta que se ha editado fichero `sendmail.cf`. Se dice asimismo que se está loco si se intenta hacerlo dos veces.

sendmail es un programa increíblemente potente. Y también, para la mayoría de la gente, increíblemente difícil de aprender y comprender. Un programa cuyo manual definitivo de referencia (**sendmail**, por Bryan Costales y Eric Allman, publicado por O'Reilly), ocupa 1,050 páginas, lo que es suficiente para espantar a cualquiera. Información sobre referencias a **sendmail** se pueden encontrar en la bibligrafía, al final de este libro.

Afortunadamente, las nuevas versiones de sendmail son diferentes. Ya no se necesitará más editar directamente el enigmático fichero `sendmail.cf`; las nuevas versiones proveen de una herramienta de configuración, la cual creará el fichero `sendmail.cf` por nosotros basándose en ficheros de macros mucho más simples. No se necesitará entender la sintaxis complicada del fichero `sendmail.cf`; los ficheros de macros no lo requieren. En lugar de eso, sólo se necesitará listar ítems, como por ejemplo el nombre de las características que se desee incluir en nuestra configuración, y especificar algunos de los parámetros que determinan cómo operará esa característica. Para esto se usará una utilidad Unix tradicional llamada **m4**, la cual toma nuestros ficheros de macros de configuración y los combina con los datos obtenidos de las plantillas que contienen la sintaxis actual de `sendmail.cf`, de forma tal que ganará nuestro propio fichero `sendmail.cf`.

En este capítulo se presentará **sendmail**, se le describirá, se enseñará cómo instalarlo, configurarlo y testearlo, usando como ejemplo la Cervecera Virtual. Si la información que se presenta aquí, hace que el proceso de configurar **sendmail** sea menos desalentador, esperamos que se gane la suficiente autoestima como para que el usuario intente abordar configuraciones más complejas por sí mismo.

18.2. Instalando Sendmail

El agente de transporte de correo, **sendmail**, se incluye preempaquetado en casi todas las distribuciones de GNU/Linux. La instalación en estos casos es relativamente sencilla. Sin embargo, existen algunas razones para instalar **sendmail** desde el código fuente, especialmente si nos importa la seguridad. El programa **sendmail** es muy complejo y se ha ganado la reputación con el paso de los años por contener fallos que permiten abrir brechas en la seguridad. Uno de los más conocidos ejemplos es el gusano de Internet RTM, el cuál sacó provecho de un problema de desbordamiento del búfer en versiones antiguas de **sendmail**. Este ejemplo ya se detalló brevemente en Capítulo 9. La mayoría de los abusos[175] que utilicen desbordamientos del búfer, dependen de que todas las copias de **sendmail** sean idénticas en las diferentes máquinas, ya que los "exploits" se basan en datos almacenados en ubicaciones específicas. Esto, desde luego, es precisamente lo que ocurre con **sendmail** instalado en las distribuciones GNU/Linux. Compilar **sendmail** desde el código fuente por usted puede ayudar a reducir este riesgo. Las versiones modernas de **sendmail** son menos vulnerables, ya que vienen con exámenes extremadamente cuidadosos en cuanto a la seguridad, la cual se ha vuelto una inquietud ampliamente generalizada en la comunidad de Internet.

El código fuente de **sendmail** está disponible por FTP anónimo en el servidor ftp.sendmail.org.

La compilación es bastante simple, ya que el paquete de los fuentes de **sendmail** soporta directamente a GNU/Linux. Los pasos a seguir para compilar **sendmail** se resumen en:

```
# cd /usr/local/src
# tar xvfz sendmail.8.9.3.tar.gz
# cd src
```

```
# ./Build
```

Para completar la instalación de los ficheros binarios resultantes, se necesitará los permisos de `root`, usando:

```
# cd obj.Linux.2.0.36.i586
# make install
```

Se han instalado ahora los binarios de **sendmail** en el directorio `/usr/sbin`. Muchos enlaces simbólicos al ejecutable **sendmail** también se instalarán en el directorio `/usr/bin/`. Se comentará sobre estos enlaces, cuando se discutan las tareas comunes al utilizar **sendmail**.

18.3. Un Vistazo a los Ficheros de Configuración

Tradicionalmente, **sendmail** era configurado a través de un fichero de configuración del sistema (llamado usualmente `/etc/mail/sendmail.cf`, o en otras distribuciones más antiguas, `/etc/sendmail.cf`, o incluso `/usr/lib/sendmail.cf`) que no se parece en nada a ningún lenguaje que haya visto usted antes. La edición del fichero `sendmail.cf` para proporcionar un comportamiento a medida puede ser una experiencia traumática.

A día de hoy, **sendmail** crea todas las opciones de configuracion a través de macros, con una sintaxis fácil de entender. El sistema de macros genera configuraciones que cubren muchas de las instalaciones, pero siempre se tiene la opción de afinar manualmente el fichero resultado, `sendmail.cf`, para trabajar en un entorno más complejo.

18.4. Los Ficheros sendmail.cf y sendmail.mc

El programa procesador de macros **m4** genera el fichero `sendmail.df` cuando procesa el fichero de configuración de macros proporcionado por el administrador del sistema local. A través de lo que resta de este capítulo, nos referiremos al fichero de configuración como el fichero `sendmail.mc`.

EL proceso de configuración es básicamente una forma de crear un fichero `sendmail.mc` apropiado que incluya macros que describan su configuración deseada. Las macros son expresiones que el procesador de macros **m4** entiende y expande en la sintaxis compleja de `sendmail.cf`. Las expresiones macro se componen del nombre de la macro (el texto en letras mayúsculas al principio), que se asemejan a una función en un lenguaje de programación, y algunos parámetros (el texto entre corchetes) que se utilizan en la expansión. Los parámetros pueden ser pasados literalmente a la salida de `sendmail.cf` o pueden ser utilizados para controlar la manera en que se realiza el procesamiento de la macro.

El fichero `sendmail.mc` para una configuración mínima (UUCP o SMTP con todos los mensajes no locales retransmitidos a un anfitrión inteligente conectado directamente a Internet) puede ser tan corto como 10 o 15 líneas, excluyendo comentarios.

18.4.1. Dos Ejemplos de los Ficheros sendmail.mc

Si usted es un administrador de varios hospedajes de correo distintos, quizá no quiera nombrar su fichero de configuración `sendmail.mc`. En vez de ésto, es practica usual nombrarlo después del nombre del anfitrión, `vstout.m4` en nuestro caso. El nombre no importa realmente en tanto que la salida sea llamada `sendmail.cf`. Proporcionar un nombre único para el fichero de configuración de cada anfitrión le permite conservar todos los ficheros de configuración en el mismo directorio y es tan sólo una conveniencia administrativa. Echemos un vistazo a dos ejemplos de ficheros de configuración de macros para que sepamos dónde estamos apuntando.

La mayoría de las configuraciones de **sendmail** de la actualidad usan SMTP únicamente. Es muy sencillo configurar **sendmail** para SMTP. Ejemplo 18−1 espera que esté disponible un servidor de nombres DNS para resolver anfitriones e intentará aceptar y enviar todo el correo para los anfitriones usando únicamente SMTP.

Ejemplo 18−1. Ejemplo de Fichero de Configuración vstout.smtp.m4

```
divert(-1)
#
# Ejemplo de fichero de configuración para vstout - smtp exclusivamente
#
divert(0)
VERSIONID(`@(#)sendmail.mc  8.7 (Linux) 3/5/96')
OSTYPE(`linux')
#
# Incluye soporte para los protocolos de transporte locales y smtp
MAILER(`local')
MAILER(`smtp')
#
FEATURE(rbl)
FEATURE(access_db)
# fin
```

Un fichero `sendmail.mc` para vstout en la Cervecera Virtual se muestra en Ejemplo 18−2. vstout usa SMTP para comunicarse con todos los anfitrións en la LAN de la cervecera, y verá la semejanza con la configuración genérica para SMTP exclusivamente que acaba de presentarse. En suma, la configuración de vstout envía todo el correo para otros destinatarios moria, su anfitrión de retransmisión a Internet por UUCP.

Ejemplo 18−2. Ejemplo de Fichero de Configuración para vstout.uucpsmtp.m4

```
divert(-1)
#
# Ejemplo de fichero de configuración para vstout
#
divert(0)
VERSIONID(`@(#)sendmail.mc  8.7 (Linux) 3/5/96')
OSTYPE(`linux')
dnl
# moria es nuestro anfitrión inteligente (smart anfitrión), usando el transporte "uucp-new".
define(`SMART_ANFITRIÓN', `uucp-new:moria')
dnl
# Soportar los protocolos de transporte de correo local, smtp y uucp.
MAILER(`local')
MAILER(`smtp')
MAILER(`uucp')
LOCAL_NET_CONFIG
# Esta regla asegura que todo correo local sea entregado usando
# transporte smtp, todo lo demás irá por el anfitrión inteligente.
R$* < @ $* .$m. > $*        $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
dnl
#
FEATURE(rbl)
FEATURE(access_db)
# fin
```

Si se comparan y contrastan las dos configuraciones, quizá sea capaz de inferir qué hace cada uno de los parámetros de configuración. Los explicaremos todos ellos en detalle.

18.4.2. Parámetros sendmail.mc Típicamente Empleados

Unos pocos de los items en el fichero `sendmail.mc` se requieren todo el tiempo; otros pueden ignorarse si las configuraciones por omisión le sirven. La secuencia general de las definiciones en el fichero `sendmail.mc` son como lo que sigue:

1. VERSIONID
2.

OSTYPE

3. DOMAIN

4. FEATURE

5. Local macro definitions

6. MAILER

7. LOCAL_* conjuntos de reglas (rulesets)

Hablaremos acerca de cada uno en las siguientes secciones y nos referiremos a nuestros ejemplos en Ejemplo 18−1 y Ejemplo 18−2, cuando sea apropiado, para explicarlos.

18.4.2.1. Comentarios

Las líneas en el fichero `sendmail.mc` que comiencen con el carácter literal # no son analizadas por **m4**, y por omisión serán sacadas directamente a la salida en el fichero `sendmail.cf`. Esto es útil si se desea comentar qué está haciendo su configuración en los ficheros de entrada y salida.

Para permitir comentarios en su `sendmail.mc` que *no* sean colocados en el fichero `sendmail.cf`, puede utilizar la orden **m4** divert y las etiquetasdnl. divert(−1) hará que toda la salida se detenga. divert(0) hará que la salida sea restaurada al valor predeterminado. Cualquier salida generada por líneas entre éstos será descartada. En nuestro ejemplo, hemos usado este mecanismo para proporcionar un comentario que aparece sólamente en el fichero `sendmail.mc`. Para conseguir el mismo resultado para una línea sencilla, puede utilizar la etiqueta dnl que significa, literalmente, [176] empezando al principio de la siguiente línea, borrar todos los caracteres hacia arriba incluyendo la nueva línea Hemos usado esto en nuestro ejemplo también.

Éstas son características estándar de **m4** y se puede obtener más información sobre ellas en su página de manual.

18.4.2.2. VERSIONID y OSTYPE

```
VERSIONID(`@(#)sendmail.mc  8.9 (Linux) 01/10/98')
```

La macro `VERSIONID` es opcional, pero es útil para grabar la versión de configuración de sendmail en el fichero `sendmail.cf`. Así que a menudo se encontrará con ello, y nosotros lo recomendamos. En cualquier caso, asegúrese de incluir:

```
OSTYPE(`linux')
```

Esta es probablemente la definición más importante. La macro `OSTYPE` provoca que se incluya un fichero de definiciones que son buenas opciones predeterminadas para su sistema operativo. La mayoría de las definiciones en un fichero macro `OSTYPE` configuran los nombres de ruta de varios ficheros de configuración, transporte de correo [177] y argumentos, y la localización de directorios que usa sendmail para almacenar mensajes. La distribución estándar de código fuente de sendmail incluye un fichero de este tipo para GNU/Linux, que se incluye en el ejemplo previo. Algunas distrbuciones de GNU/Linux, particularmente la distribución Debian, incluye su propio fichero de definiciones que es compatible con la jerarquía de ficheros de GNU/Linux. Cuando su distribución haga ésto, probablemente debería usar su definición en vez de la genérica de GNU/Linux por omisión.

La definición OSTYPE debería ser una de las primeras en aparecer en su fichero `sendmail.mc`, debido a que muchas otras definiciones dependen de ella.

18.4.2.3. DOMAIN

La macro DOMAIN es útil cuando se desea configurar un gran número de máquinas en la misma red de una manera estandarizada. Si se está configurando un pequeño número de anfitrións, probablemente es mejor no preocparse de ésto. Típicamente se configuran items, como el nombre de los anfitrións de retransmisión [178] o concentradores [179] que todos los anfitrións de la red utilizarán.

La instalación estándar continene un directorio de plantillas de macros **m4** utilizadas para dirigir el proceso de configuración. Este directorio se nombra normalmente como `/usr/share/sendmail.cf` o algo por el estilo. Aquí se encuentra un subdirectorio nombrado `domain` que contiene plantillas de configuración específicas para el dominio. Para hacer uso de una macro DOMAIN, se debe crear su propio fichero macro conteniendo las definiciones estándar que se requieran para el sitio, y escribirlas en el subdirectorio `domain`. Sólo se deberían incluir normalmente las definiciones de macro que sean únicas a su dominio aquí, como las definiciones de los anfitrión inteligentes o los anfitrión de retransmisión, pero no limitándose a éstos.

La distribución del código fuente de **sendmail** viene con algunos ficheros de ejemplo de definición de macros que puede usar para crear el suyo propio.

Si salvó su fichero de macros de dominio como `/usr/share/sendmail.cf/domain/vbrew.m4`, debería incluir las definiciones en su `sendmail.mc` usando:

```
DOMAIN(`vbrew')
```

18.4.2.4. FEATURE

La macro FEATURE permite incluir características predefinidas de **sendmail** en su configuración. Estas características de **sendmail** hacen las configuraciones soportadas muy fáciles de usar. Hay un gran número, y a través de este capítulo hablaremos únicamente de unas pocas se pueden encontrar detalles completos de las características disponibles en el fichero `CF` incluido en el paquete de fuentes.

Para usar cualquiera de las características listadas, debería incluir una línea en su fichero `sendmail.mc` que se parezca a esto:

```
FEATURE(nombre)
```

donde *`nombre`* se sustituye con el nombre de la característica. Algunas características toman un parámetro opcional. Si desea usar algún otro distinto al predeterminado, debería utiizar una entrada como ésta:

```
FEATURE(nombre, parámetro)
```

donde *`parámetro`* es el parámetro a suministrar.

18.4.2.5. Definiciones de macros locales

Los ficheros estándar de configuración de **sendmail** proporcionan una buena cantidad de maneras y variables con las que se puede personalizar la configuración. Éstas se llaman **definiciones de macros locales**. Muchas de ellas se listan en el fichero `CF` en el paquete de fuentes de **sendmail**.

La definiciones de macros locales son normalmente invocadas mediante el suministro del nombre de la macro con un argumento representando el valor que se quiere asignar a la variable que gestiona la macro. Otra vez, exploraremos algunas de las definiciones de macro locales más comunes en los ejemplos que presentaremos más tarde en este capítulo.

18.4.2.6. Definición de los protocolos de transporte de correo

Si desea que **sendmail** transporte correo de cualquier otra forma que por entrega local, debe indicarle qué transporte tiene que emplear. La macro `MAILER` hace esto muy fácil. La versión actual de **sendmail** soporta una variedad de protocolos de transporte de correo; algunos de éstos son experimentales, otros son probablemente usados raramente.

En nuestra red necesitamos transporte SMTP para enviar y recibir correo entre los anfitrións en nuestra red local, y el transporte UUCP para enviar y recibir correo desde nuestro anfitrión inteligente. Para lograr ésto, simplemente incluimos ambos transportes de correo, el `smtp` y el `uucp`. El transporte de correo `local` se incluye por omisión, pero puede ser definido por claridad, si o desea. Si se incluyen ambos `smtp` y el `uucp` en su configuración, debe asegurarse siempre de definir primero el agente de correo `smtp`.

Los transportes más comúnmente usados disponibles al usar la macro MAILER se describen en la siguiente lista:

local
: Este transporte incluye tanto el agente de entrega local usado para enviar correo al buzón de los usuarios en esta máquina como el programa prog transporte de entrega usado para enviar mensajes a programas locales. Este transporte se incluye por omisión.

smtp
: Este transporte implementa el Protocolo Simple de Transporte de Correo (SMTP), que es el medio más usual de transporte de correo en Internet. Cuando se incluye este transporte, se configuran cuatro transportes de correo: `smtp` (SMTP básico), `esmtp` (SMTP Extendido), `smtp8` (SMTP binario plano de 8 bits), y `relay` (específicamente diseñado para hacer de transporte a modo de pararela entre anfitriones).

uucp
: El transporte uucp proporciona soporte para dos transportes de correo: uucp−old, que es el UUCP tradicional, y uucp−new, que permite manipular múltiples buzones en una transferencia.

usenet
: Este transporte de correo permite enviar mensajes directamente a redes de noticias del estilo Usenet. Cualquier mensaje local dirigido a una dirección de *news.group.usenet* será introducido en la red de noticias para el grupo de noticias *news.group*.

fax
: Si tiene el software Hylafax instalado, este transporte permitirá dirigir correo electrónico a él, para que así pueda construir una pasarela de correo−fax. Esta característica era experimental cuando se escribió este documento. Puede obtenerse más información en: http://www.vix.com/hylafax/.

Hay otros como pop, procmail, mail11, phquery, y cyrus que son útiles, pero menos comunes. Si le pica la cusiosidad, puede leer sobre éstos en el libro de sendmail o en la documentación suministrada en el paquete de fuentes.

18.4.2.7. Configuración de encaminamiento de correo para anfitriones locales

La configuración de la Cervecera Virtual es probablemente más compleja que lo que muchos sitios requieren. La mayoría de los sitios de hoy usarían transporte SMTP únicamente y no tendrían que tratar con UUCP para nada. En nuestra configuración hemos configurado un anfitrión inteligente que es usado para tratar todo el correo saliente. Debido a que estamos usando un transporte SMTP en nuestra red local debemos indicarle a **sendmail** que no envíe correo local por el anfitrión inteligente. La macro `LOCAL_NET_CONFIG` permite introducir reglas sendmail directamente en el fichero de salida `sendmail.cf` para modificar la manera en que el correo local es manipulado. Hablaremos más sobre reglas de reescritura más tarde, pero por el momento se debería aceptar que la regla que hemos suministrado en nuestro ejemplo especifica que cualquier correo destinado a anfitriones dentro del dominio vbrew.com deberían entregarse directamente al anfitrión objetivo usando el transporte de correo SMTP.

18.5. Generando el Fichero sendmail.cf

Cuando haya completado la edición del fichero de configuración **m4**, debe procesarlo para producir el fichero `/etc/mail/sendmail.cf` leído por **sendmail**. Esto se hace directamente como se ilustra en el siguiente ejemplo:

```
# cd /etc/mail
# m4 /usr/share/sendmail.cf/m4/cf.m4 vstout.uucpsmtp.mc >sendmail.cf
```

Esta orden invoca el procesador de macros **m4** suministrándole el nombre de dos ficheros de definición de macros para procesar. **m4** procesa los ficheros en el orden dado. El primer fichero es una plantilla de macro de **sendmail** estándar suministrada en el paquete de fuentes de **sendmail**, el segundo, por supuesto, es el fichero que contiene nuestras propias definiciones de macros. La salida de la orden es dirigida al fichero `/etc/mail/sendmail.cf`, que es nuestro fichero objetivo.

Puede ahora iniciar **sendmail** con la configuración nueva.

18.6. Interpretación de las Reglas de Escritura − Reescritura

Es argumentable que la característica más poderosa de **sendmail** es la regla de reescritura. Las reglas de reescritura son usadas por **sendmail** para determinar cómo procesar un manseje de correo recibido. **sendmail** pasa las direcciones desde las *cabeceras* de un mensaje de correo a través de colecciones de reglas de reescritura llamadas *conjuntos de reglas*. Las reglas de reescritura transforman una dirección de correo de una forma a otra y puede pensar en ellas como algo similar a una orden en su editor que reemplaza todo el texto que encaje en un patrón especificado con otro.

Cada regla tiene un lado izquierdo y un lado derecho, separados por al menos un carácter de tabulación. Cuando **sendmail** está procesando correo, busca a través de las reglas de reescritura intentando encontrar una coincidencia con el lado izquierdo. Si una dirección coincide con una de las reglas del lado izquierdo, la dirección es reemplazada por la del lado derecho y es procesada de nuevo.

18.6.1. Las Órdenes R y S de sendmail.cf

En el fichero `sendmail.cf`, los conjuntos de reglas son definidos usando órdenes codificadas como `S`*n*, donde *n* especifica el conjunto de reglas que se considera el actual.

Las reglas por sí mismas aparecen en órdenes codificadas como **R**. Cuando cada orden **R** es leída , se añade al conjunto de reglas actual.

Si está tratando sólo con el fichero `sendmail.mc`, no necesita preocuparse acerca de las órdenes **S** para nada, ya que las macros construirán éstas por usted. Necesitará codificar manualmente las reglas **R**.

Un conjunto de reglas de **sendmail** entonces tiene la siguiente apariencia:

```
Sn
    Rlhs rhs
    Rlhs2 rhs2
```

18.6.2. Algunas Definiciones de Macro Útiles

sendmail utiliza internamente unas cuantas definiciones de macro estandarizadas. Las más útiles de éstas en la escritura de conjuntos de reglas son:

$j
: El nombre completamente cualificado de este anfitrión (FQDN).

$w
: El componente del anfitrión del FQDN.

$m
: El componente del dominio del FQDN.

Podemos incorporar estas definiciones de macros en nuestras propias reglas de reescritura. Nuestra configuración de la Cervecera Virtual utiliza la macro `$m`.

18.6.3. El Lado Izquierdo

A la izquierda de una regla de reescritura, hay que especificar un patrón que coincida con una dirección que desee transformar. La mayoría de los caracteres se les hace coincidir literalmente, pero hay un número de caracteres que tienen significado especial; estos se describen en la lista siguiente. Las reglas de reescritura para el lado izquierdo son:

`$@`
: Coinciden exactamente cero símbolos

`$*`
: Coinciden cero o más símbolos

`$+`
: Coincide uno o más símbolos

`$-`
: Coincide exactamente un símbolo

`$=x`
: Coindice cualquier frase en la clase *`x`*

`$~x`
: Coincide con cualquier palabra que no esté en la clase *`x`*

Un símbolo es una tira de caracteres delimitados por espacios. No hay manera de incluir espacios en un símbolo, o no es necesario como los patrones de expresiones son suficientemente flexibles para atajar esta necesidad. Cuando una regla coincide con una dirección, el texto que coincide con cada uno de los patrones en la expresión será asignado a variables especiales que se usarán en la parte derecha. La única excepción a esto es con el literal `$@`, que no coincide con ningún símbolo y entonces nunca generará texto para ser usado en el lado derecho.

18.6.4. El Lado Derecho

Cuando el lado izquierdo de una regla de reescritura coincide con una direción, el texto original se borra y se reemplaza por lo que haya en el lado derecho de la regla. Todos los símbolos en el lado derecho son copiados literalmente , a no ser que comiencen por el signo del dólar. De la misma manera que en el lado izquierdo, unos cuantos metasímbolos pueden usarse en el lado derecho. Estos son descritos en la siguiente lista. Las reglas de reescritura para el lado derecho son:

`$n`
: Este metasímbolo es reemplazado por la expresión *`n`*ésima del lado izquierdo.

`$[nombre$]`
: Este metasímbolo resuelve el nombre del anfitrión a nombre canónico. Es reemplazado por la forma canónica del nombre del anfitrión suministrado.

`$(map_clave $@argum $:omisión $)`
: Esta es la forma más general de búsqueda. La salida es un resultado de mirar la *`clave`* en el mapa nombrado *map* pasándole *`argum`* como argumentos. El *mapa* puede ser cualquiera de los mapas que **sendmail** soporta como el `virtusertable` que describimos un poco más tarde. Si la búsqueda es infructuosa, *`por omisión`* será la salida. Si no se suministra nada por omisión y la búsqueda falla, la entrada no se altera y la *`clave`* es la salida.

`$>n`
: Esto hará que el resto de esta lína sea analizada y entonces dada al conjunto de reglas *`n`* para ser evaluada. La salida del conjunto de reglas llamado se escribirá como salida a esta regla. Éste es el mecanismo que permite a las reglas invocar otras reglas.

`$#mailer`
: Este metasímbolo hace que la evaluación del conjunto de reglas se detenga y especifica el transporte que deberá usarse para transportar este mensaje en el siguiente paso de su entrega. Este metasímbolo debería ser llamado sólo desde el conjunto de reglas 0 o una de sus subrutinas. Esta es la parte final del análisis de direcciones y debería ser acompañado de los dos siguientes metasímbolos.

`$@host`
: Este metasímbolo especifica el anfitrión al que este mensaje será reenviado. Si el anfitrión destinatario es el anfitrión local, puede omitirse. El *`host`* puede ser una lista de anfitriones de destino separada por dos puntos (:) que a los que se intentará entregar el mensaje en secuencia.

`$:user`
: Este metasímbolo especifica el usuario destinatario para el mensaje de correo.

Una regla de reescritura que coincide se intenta repetidamente hasta que falla una coincidencia, entonces el análisis continúa en la sigiente regla. Este comportamiento puede cambiarse precediendo el lado derecho con uno de dos metasímbolos especiales descritos en la siguiente lista. Las reglas de reescritura para el control del bucle del lado derecho son:

`$@`
: Este metasímbolo causa que el conjunto de regles retorne con el resto del lado derecho como el valor. Ninguna otra regla del conjunto se evalúa.

`$:`
: Este metasímbolo causa que esta regla finalice inmediatamente, pero el resto del conjunto de reglas actual es evaluado.

18.6.5. Un Ejemplo de Patrón de Reglas Sencillo

Para ver mejor cómo funcionan las macros de sustitución de patrones, considere la siguiente regla de lado izquierdo:

```
    $* < $+ >
```

Esta regla coincide con Cero o más símbolos, seguidos por el carácter <, seguidas a su vez por una o más símbolos, seguidos por el carácter >.

Si esta regla fuese aplicada a `brewer@vbrew.com` o `Head Brewer < >`, la regla no coincidiría. La primera cadena no coincidiría porque no incluye el carácter <, y la segunda fallaría porque `$+` coincide con *uno o más* símbolos y no hay símbolos entre los caracteres <>. En cualquier caso en que una regla co coincida, el lado derecho de la regla no se usa.

Si la regla fuera aplicada a `Head Brewer < brewer@vbrew.com >`, la regla coincidiría, y en el lado derecho `$1` sería sustituido por `Head Brewer` y `$2` sería sustituido por `brewer@vbrew.com`.

Si la regla fuese aplicada a `< brewer@vbrew.com >` la regla coincidiría porque `$*` coincide con *cero* o más símbolos, y en el lado derecho `$1` podría ser sustituido por la cadena vacía.

18.6.6. Semántica del Conjunto de Reglas

Cada uno de los conjuntos de reglas de **sendmail** se les llama para realizar una tarea distinta en el procesado del correo. Cuando se están escribiendo reglas, es importante entender qué se espera que cada uno de los conjuntos de reglas haga. Vamos a echar un vistazo a cada uno de los conjuntos de reglas que los guiones de configuración **m4** nos permiten modificar:

LOCAL_RULE_3

: El conjunto 3 es responsable de convertir una dirección en un formato arbitrario en un formato común que**sendmail** procesará. El formato de salida esperado es el aspecto familiar *`parte-local@especificación-anfitrión-dominio`*.

  El conjunto 3 debería poner la parte del nombre del anfitrión de la dirección convertida entre los caracteres < y > para hacer el análisis de las siguientes reglas más fácil. El conjunto de reglas 3 se aplica antes que **sendmail** haga cualquier otro procesamiento de una dirección de correo, así que si quiere que **sendmail** haga de pasarela de correo desde algún sistema que utilice algún formato de dirección poco usual, se debería añadir una regla usando la macro `LOCAL_RULE_3` para convertir direcciones en el formato común.

LOCAL_RULE_0 y LOCAL_NET_CONFIG

: El conjunto 0 se aplica por **sendmail** a las direcciones del destinatario tras el conjunto de reglas 3. La macro `LOCAL_NET_CONFIG` provoca que las reglas sean introducidas en la *mitad inferior* del conjunto 0.

  El conjunto 0 se espera que realice la entrega del mensaje al destinatario, así que debe resolver un triplete que especifica el correo, el anfitrión y el usuario. Las reglas serán colocadas antes de cualquier definición de anfitrión inteligente que quiera incluir, así que si añade reglas que resuelvan direcciones apropiadamente cualquier dirección que coincida con una regla no será tratada por el anfitrión inteligente. Así es como tratamos los **smtp** directos para los usuarios de nuestra red local en nuestro ejemplo.

LOCAL_RULE_1 y LOCAL_RULE_2

: El conjunto 1 se aplica a todas las direcciones de remite y el conjunto 2 de aplica a todas las direcciones de destino. Ambos están normalmente vacíos.

18.6.6.1. Interpretación de la regla en nuestro ejemplo

Nuestro ejemplo en Ejemplo 18−3 usa la macro `LOCAL_NET_CONFIG` para declarar una regla local que asegure que cualquier correo dentro de nuestro dominio se entregue directamente usando el transporte de correo **smtp**. Ahora que sabe cómo se construyen las reglas de reescritura, es capaz de entender cómo funciona esta regla. Echémosle un vistazo.

Ejemplo 18−3. Regla de reescritura desde vstout.uucpsmtp.m4

```
LOCAL_NET_CONFIG
# Esta regla se asegura de que todo correo local sea entregado usando el
# transporte smtp, todo lo demás irá por el anfitrión inteligente.
R$* < @ $* .$m. > $*  $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
```

Sabemos que la macro `LOCAL_NET_CONFIG` hará que la regla se introduzca en algún lugar cerca del final del conjunto de reglas 0, pero antes de cualquier definición del anfitrión inteligente. Sabemos también que el conjunto 0 es el último conjunto en ser ejecutado y que debería resolver un triplete especificando transporte de correo, usuario y anfitrión.

Podemos ignorar las dos líneas de comentariso; no hacen nada útil. La regla en sí misma es la línea que comienza con `R`. Sabemos que la `R` es una instrucción de **sendmail** y que añade esta regla al conjunto de reglas actual, en este caso el conjunto `0`. Miremos al lado izquierdo y al lado derecho que devuelve.

El lado izquierdo es como éste: `$* < @ $* .$m. > $*`.

El conjunto 0 espera los caracteres < y > porque es alimentado por el conjunto 3. El conjunto 3 convierte direcciones en una forma común y para hacer el análisis más fácil, coloca la parte del anfitrión de la dirección de correo entre < y >.

Esta regla coincide con cualquier dirección que parecezca como: `'UsuarioDestino < @ cualquieranfitrión.nuestrodominio. > Algún Texto'`. Esto es, coincide con el correo de cualquier usuario y de cualquier anfitrión dentro de nuestro dominio.

Recordará que el texto que coincide con los metasímbolos en el lado izquierdo de una regla de reescritura se asigna a definiciones de macro para su uso en el lado derecho. En nuestro ejemplo, el primer `$*` coincide con todo el texto desde el inicio de la dirección hasta el carácter <. Todo este texto se asigna al `$1` para su uso en el lado derecho. Similarmente, el segundo `$*` en nuestra regla de reescritura se asigna a `$2`, y el último se asigna a `$3`.

Ahora tenemos suficiente para entender el lado izquierdo. Esta regla coincide con el correo de cualquier usuario en cualquier anfitrión dentro de nuestro dominio. Asigna el nombre de usuario a `$1`, el nombre del anfitrión a `$2`, y cualquier texto subsiguiente a `$3`. El lado derecho se invoca entonces para procesar éstos.

Echemos un vistazo a aquello que estamos esperando ver a la salida. El lado derecho de nuestra regla de reescritura de ejemplo es semejante a: `$#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3`.

Cuando la regla del lado derecho de nuestro conjunto de reglas se procesa, se interpreta cada uno de los metasímbolos y se realizan las sustituciones correspondientes.

El metasímbolo `$#` hace que esta regla resuelva un transporte específico, *smtp* en nuestro caso.

El `$@` resuelve el anfitrión objetivo. En nuestro ejemplo, el anfitrión objetivo se especifica como `$2.$m.`, el cual es el nombre completamente cualificado del anfitrión en nuestro dominio. El NDCC se construye con el componente del nombre del anfitrión asignado a `$2` desde nuestro lado izquierdo con nuestro nombre de dominio (`.$m.`) concatenado.

El metasímbolo `$:` especifica el usuario objetivo, el cual se captura otra vez del lado izquierdo y se almacena en `$1`.

Preservamos los contenidos de la sección `<>` y cualquier texto acompañante, usando los datos que recogimos desde el lado izquierdo de la regla.

Debido a que esta regla resuelve a un transporte de correo, el mensaje es reenviado al transporte para su entrega. En nuestro exemplo, el mensaje sería reenviado al anfitrión de destino usando el protocolo SMTP.

18.7. Configuración de las Opciones de Sendmail

sendmail tiene un número de opciones que permiten personalizar la manera en que realiza ciertas tareas. Hay un número mayor de éstas, así que sólo hemos listado unas pocas de las más usadas normalmente en la lista siguiente:

Para configurar cualquiera de estas opciones, se pueden definirlas en el fichero de configuración **m4**, que es el método preferible, o puede insertarlas directamente en el fichero `sendmail.cf`. Por ejemplo, si deseásemos que **sendmail** se bifurque en un proceso nuevo para cada mensaje de correo que tenga que entregarse, quizá añadamos la siguiente línea a nuestro fichero de configuración **m4**:

```
define(confSEPARATE_PROC,true)
```

La correspondiente entrada creada en `sendmail.cf` es:

```
O ForkEachJob=true
```

La siguiente lista describe opciones usuales de *sendmail m4* (y las equivalentes en `sendmail.cf`):

`confMIN_FREE_BLOCKS (MinFreeBlocks)`
: Hay ocasiones cuando un problema quizá impida la entrega inmediata de mensajes de correo, haciendo que los mensajes esperen en la cola de envío. Si su anfitrión de correo procesa un gran volumen de correo, es posible que la cola de correo crezca hasta tal tamaño que llene completamente el sistema de ficheros en que la cola se encuentra. Para prevenir esto, **sendmail** proporciona esta opción para especificar el número mínimo de bloques de disco libres que deben existir antes de que un mensaje de correo sea aceptado. Esto permite asegurarse de que **sendmail** nunca provoca que el sistema de ficheros de la cola de correo se llene (Predeterminado a: 100).

`confME_TOO (MeToo)`
: Cuando un objetivo de correo como un alias de correo se expande, es a veces posible para el remitente aparecer en la lista de destinatarios. Esta opción determina si los originadores de un mensaje de correo electrónico recibirán una copia si aparecen en la lista de destinatarios expandida. Los valores válidos son true y false (Predeterminado a: false).

`confMAX_DAEMON_CHILDREN (MaxDaemonChildren)`
: Cuando **sendmail** recibe una conexión SMTP desde un anfitrión remoto, inicia una copia nueva de sí mismo para tratar el mensaje entrante. De esta manera, es posible para **sendmail** estar procesando múltiples mensajes entrantes simultáneamente. Aunque esto es útil, cada nueva copia de **sendmail** consume memoria en la computadora anfitrióna. Si se recibe un número inusualmente grande de conexiones entrantes, por ejemplo, debido a un problema o a un ataque malicioso, es posible que los demonios de **sendmail** consuman toda la memoria del sistema. Esta opcioón le permite limitar el número máximo de demonios hijos que serán iniciados. Cuando este número se alcance, las conexiones nuevas se deniegan hasta que uno de los demonios hijos haya terminado. (Predeterminado a: undefined(indefinido)).

confSEPARATE_PROC (ForkEachJob)
: Cuando se procesa una cola de correo y se envían mensajes de correo, **sendmail** procesa un mensaje de correo cada vez. Cuando esta opción está activada, **sendmail** se bifurcará en copias nuevas de sí mismo para cada mensaje a ser entregado. Esto es particularmente útil cuando hay varios mensajes de correo atascados en la cola porque hay un problema con el anfitrión objetivo. (Predeterminado: false).

confSMTP_LOGIN_MSG (SmtpGreetingMessage)
: Cuando se hace una conexión a **sendmail**, se envía un mensaje de bienvenida. Por omisión, este mensaje contiene el nombre del anfitrión, el nombre del agente de transporte de correo, el número de versión de sendmail, el número de versión local, y la fecha actual. El RFC821 especifica que la primera palabra del saludo debería ser el nombre completamente cualificado del anfitrión, pero el resto del saludo, puede ser configurado a su gusto. Puede especificar macros de sendmail aquí y serán expandidas y usadas. La única gente que verá este mensaje serán los administradores de sistemas que estén diagnosticando problemas de envío de correo, o gente fuertemente curiosa interesada en descubrir cómo está configurada su máquina. Puede aliviar algo del tedio de su tarea personalizando el mensaje de bienvenida con algunos chistes; sea amable. La palabra EMSTP será introducida por **sendmail** entre la primera palabra y la segunda, como señal para los anfitriones remotos de que soportamos el protocolo ESMTP. (Predeterminado: `$j Sendmail $v/$Z; $b`).

18.8. Algunas configuraciones útiles para Sendmail

Hay una miríada de posibles configuraciones de **sendmail**. En este espacio se ilustrarán sólo unos cuantos tipos de configuración de importancia que serán muy útiles para muchas instalaciones de **sendmail**.

18.8.1. Confiar en los usuarios para que pongan el campo From:

En ocasiones es útil sobreescribir el campo `From:` de un mensaje de correo que va hacia afuera. Supongamos que se tiene un programa basado en web que genera correo electrónico. Normalmente el mensaje de correo aparecerá como proviniente del usuario que es el dueño del proceso del servidor de web. Pero se quiere especificar alguna otra dirección remitente de tal forma que el correo parezca ser originado por otro usuario o dirección en esa máquina. **Sendmail** permite especificar en qué usuarios del sistema se puede confiar para que tengan la habilidad de hacer esto.

La opción `use_ct_file` habilita la posibilidad de especificar y dar un fichero que liste los nombres de los usuarios de confianza. Por omisión, un pequeño número de usuarios del sistema son de confianza de **sendmail**, por ejemplo `root`. El nombre del fichero por omisión para esta opción es `/etc/mail/trusted-users` en los sistemas en los que el directorio de configuración es `/etc/mail` y en `/etc/senmdail.ct` en donde es el otro tipo de configuración. Se puede especificar el nombre y lugar del fichero sobreescribiendo la definición `confCT_FILE`.

Escriba FEATURE(use_ct_file) en el fichero `sendmail.mc` para habilitar esta opción.

18.8.2. Managing Mail Aliases

Los alias de correo son una poderosa opción que permite que el correo sea dirigido a otros apartados postales que son nombres alternativos de usuarios o procesos en un servidor destinatario. Por ejemplo, es una práctica común tener retroalimentación o comentarios con respecto a un servidor de Web y que estén dirigidos a webmaster. Con frecuencia no hay un usuario llamado webmaster. en el servidor, en vez de ello, hay un alias a otro usuario del sistema. Otro uso común para los alias de correo es utilizarlos por los programas de gestión de listas de correo en los cuales un alias dirige todos los mensajes que ingresan al programa de gestión de la lista para que sea interpretado.

El fichero `/etc/aliases` es el lugar en donde los alias se almacenan. El programa **sendmail** consulta este fichero cuando está determinando cómo manejar un mensaje que ingresa. Si encuentra una línea en este fichero que coincide con el usuario a quien va dirigido el mensaje, lo redirige al lugar que indica dicha línea.

De forma específica, hay tres cosas que los alias permiten:

- Otorgan un nombre corto o bien conocido para el correo que será dirigido hacia una o más personas.
- Pueden invocar a un programa con el mensaje de correo como entrada hacia dicho programa.
- Pueden mandar el correo a un fichero.

Todos los sistemas requieren de alias para el Postmaster y el MAILER−DAEMON para cumplir con el RFC.

Se debe ser especialmente cuidadoso con la seguridad cuando se definan alias que invoquen o escriban a programas, ya que **sendmail** por lo general se ejecuta con los permisos de root.

Se pueden encontrar más detalles con respecto a los alias de correo en la página de manual `aliases(5)`. Una ejemplo del fichero `aliases` se muestra en Ejemplo 18−4.

Ejemplo 18−4. Ejemplo del fichero aliases

```
#
# Los siguientes dos alias deben estar presentes para cumplir con el RFC.
# Es importante resolverlos en 'una persona' que lea su correo con regularidad.
#
postmaster:    root                             # línea indispensable
MAILER-DAEMON: postmaster                       # línea indispensable
#
#
# demuestra los tipos más comunes de alias
#
usenet:        janet                            # alias para una persona
admin:         joe,janet                        # alias para varias personas
newspak-users: :include:/usr/lib/lists/newspak # lee a los destinatarios desde un fichero
changefeed:    |/usr/local/lib/gup              # alias que invoca a un programa
complaints:    /var/log/complaints              # alias que escribe el correo a un fichero
#
```

Cada vez que actualice el fichero `/etc/aliases`, se debe asegurar de ejecutar el programa:

```
# /usr/bin/newaliases
```

para reconstruir la base de datos que **sendmail** utiliza internamente. La orden **/usr/bin/newaliases** es un vínculo simbólico al ejecutable de **sendmail** y, cuando se invoca de esta forma, se comporta exactamente como si hubiese sido invocado así:

```
# /usr/lib/sendmail −bi
```

La orden **newaliases** es una forma alternativa y más adecuada para hacer esto.

18.8.3. Cómo usar un anfitrión inteligente

Algunas veces un anfitrión encuentra correo que no puede entregar directamente a un sitio remoto. Con frecuencia es conveniente tener un único sitio en una red que tenga el papel de gestionar la transmision del correo a sitios remotos que son difíciles de alcanzar, en vez de que cada sitio local intente hacer esto por sí mismo.

Hay algunas buenas razones para que se tenga un solo sitio encargado de la gestión del correo. Se simplifica la gestión al tener sólo un sitio con una configuración cuidadosa del correo que sepa cómo manejar todos los tipos de transporte de correo, tales como UUCP, Usenet, etc. Todos los otros sitios lo único que necesitan es un solo protocolo de transporte para enviar su correo a este anfitrión central. Los sitios que cumplen este papel de encaminadores centrales y reenviadores se llaman *anfitriones inteligentes*. Si se tiene un anfitrión inteligente que acepte correo de usted, se puede enviar correo de cualquier tipo y él se encargará de gestionar el encaminamiento y la transmisión de ese correo a todos los sitios remotos deseados.

Otra buena aplicación para la configuración de anfitriones remotos es gestionar la transmisión del correo a través de un cortafuegos privado. Una organización puede elegir instalar una red IP privada y utilizar sus propias direcciones IP no registradas. La red privada se puede conectar a Internet mediante un cortafuegos. El enviar el correo desde y hacia los diversos anfitriones dentro de la red privada hacia el mundo exterior utilizando SMTP no será posible en una configuración convencional debido a que los sitios locales no pueden establecer una conexión directa de red a los sitios que están en Internet. En cambio, la organización puede optar por que el cortafuegos tenga la función de anfitrión inteligente. El anfitrión inteligente que se ejecute en el cortafuegos será capaz de establecer conexiones directas de red con los sitios que se encuentran tanto en el interior de la red privada como en el exterior de ella. El anfitrión inteligente puede aceptar correo de ambos anfitriones, de los que están en la red privada y de los que están en Internet, el correo se guarda en un almacenamiento local y luego se gestiona la retransmisión de ese correo directamente al sitio adecuado.

Los anfitriones inteligentes se utilizan en general cuando todos los otros métodos de entrega han fallado. En el caso de una organización con una red privada, es perfectamente razonable que los anfitriones primero intenten entregar el correo directamente, y si eso falla, entonces los envían al anfitrión inteligente. Esto descarga mucho el tráfico que va hacia el anfitrión inteligente debido a que los otros anfitriones pueden enviar correo directamente a otros anfitriones dentro de la red privada.

sendmail provee de un método simple para configurar un anfitrión inteligente utilizando la opción `SMART_HOST`; para implementarlo en la configuración de la Cervecera Virtual, se hace exactamente esto. Las porciones relevantes de nuestra configuración que definen al anfitrión inteligente son:

```
define(`SMART_HOST', `uucp-new:moria')
LOCAL_NET_CONFIG
# Esta regla asegura que todo el correo local se entrega utilizando
# el transporte smtp, todo lo demás se va a través del anfitrión inteligente.
R$* < @ $* .$m. > $*        $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
```

La macro `SMART_HOST` permite que se especifique el anfitrión que reenviará todo el correo de salida que no se pueda entregar directamente y el protocolo de transporte de correo que se debe utilizar para ello.

En nuestra configuración se está usando el transporte `uucp-new` hacia el sitio UUCP **moria**. Si se quisiera configurar **sendmail** para que utilice un anfitrión inteligente con SMTP, se debería escribir algo como lo siguiente:

```
define(`SMART_HOST', `mail.isp.net')
```

No se necesita especificar que el transporte es SMTP, ya que está dicho por omisión.

¿Puede adivinar lo que la macro `LOCAL_NET_CONFIG` y la regla de reescritura podría estar haciendo?

La macro `LOCAL_NET_CONFIG` permite agregar reglas de reescritura directamente a la configuración de **sendmail** que definirá qué correo se deberá quedar dentro del sistema local. En nuestro ejemplo, se ha utilizado una regla en la que cualquier correo electrónico cuyo dominio coincida con el dominio de nuestro anfitrión (`.$m.`) se reescribe para ser enviado directamente usando el transporte `SMTP`. Esto asegura que cualquier mensaje enviado hacia un anfitrión dentro de nuestro dominio local será redirigido inmediatamente al transporte SMTP y enviado directamente a ese anfitrión en vez de pasar a través de nuestro anfitrión inteligente, que es el tratamiento por omisión.

18.8.4. Gestionando correo no deseado o no solicitado (Spam)

Si se ha suscrito a una lista de correo, publicado su dirección de correo electrónico en un sitio web, o enviado un artículo a UseNet, lo más probable es que comience a recibir correo electrónico no solicitado con anuncios. Son los lugares comunes en donde la gente que ronda por la red busca las direcciones de correo para agregarlas a listas de correo que luego venden a compañías que buscan anunciar sus productos. A este tipo de correo masivo se le conoce como spam.

El diccionario gratuito de la computación en línea ofrece una definición con respecto al correo de spam que dice: [180]

> 2. (En un sentido más estricto que 1, arriba) El envío indiscriminado de grandes cantidades de correo electrónico no solicitado para promocionar un producto o un servicio. El spam, en este sentido, es una especie equivalente electrónico de el correo basura enviado al "inquilino."
>
> En los años 90, con el crecimiento del interés comercial en la red, hay algunas personas sin escrúpulos[181] que ofrecen el uso del spam como un "servicio" a las compañías que quieren anunciarse en la red. Ello lo consiguen al enviar mensajes a grandes colecciones de direcciones de correo, foros de noticias de Usenet o listas de correo. Dichas prácticas han causado furia y reacción agresiva de muchos usuarios de la red en contra de dichos individuos.

Por fortuna, **sendmail** tiene algunos mecanismos que pueden ayudar a tratar al correo no solicitado.

18.8.4.1. Las Listas Negras en Tiempo Real (RBL)

Las listas de exclusión [182] en tiempo real (RBL, Real−time Blackhole List) es una lista pública que ayuda a reducir el volumen de anuncios no solicitados con los que se tiene que tratar. Algunas fuentes de correo electrónico están en listadas en una base de datos consultable a través de Internet. Ellos han sido incluidos allí por la gente que recibe anuncios no solicitados de alguna dirección de correo. Los grandes dominios, en ocasiones están en dicha lista debido a algún resbalón que les impidió detener el spam. Mientras que alguna gente se queja de alguna selección en particular hecha por los mantenedores de la lista, aún sigue siendo muy popular y los errores se arreglan con rapidez. Todos los detalles de la operación de cómo opera el servicio están en el sitio web del Sistema de Protección contra el Abuso del Correo en *http://maps.vix.com/rbl/*.

Si se habilita esta opción de **sendmail**, se buscará la dirección de origen de cada mensaje que llegue en la base de datos de la Lista Negra en tiempo real para determinar si se acepta o no el mensaje. Si se tiene un gran sitio con muchos usuarios, esta opción podría ahorrarles una gran cantidad de espacio en disco. Esta opción acepta como parámetro especificar el nombre del servidor que se va a utilizar. El servidor principal por omisión es rbl.maps.vix.com.

Para configurar la opción de "listas negras en tiempo real", se debe agregar la siguiente declaración de macro en el fichero `sendmail.mc`:

```
FEATURE(rbl)
```

Si se quiere especificar otro servidor de RBL, la declaración que se debe escribir debe ser como la siguiente:

```
FEATURE(rbl,`rbl.host.net')
```

18.8.4.2. La base de datos de acceso

Un sistema alternativo que tiene gran flexibilidad para el control a cambio del costo que implica una configuración manual es la opción access_db. La base de datos de acceso permite configurar qué anfitriones o usuarios serán aceptables para enviar correo y quiénes pueden utilizarlo como puente.

Gestionar a quiénes se les permitirá reenviar el correo es muy importante ya que el reenvío es una técnica de uso común para mandar correo basura a los anfitriones que tienen sistemas como el RBL del que se comentó anteriormente para evitar la basura. En vez de enviar el correo directamente, los 'spammers' utilizarán el reenvío a través de un anfitrión que, ingenuamente, lo permita. La conexión entrante de SMTP no provendrá del anfitrión conocido por enviar basura, sino de quien lo reenvió. Para garantizar que nuestro anfitrión no sea utilizado de esta forma, sólo se debe reenviar el correo de los sitios autorizados. Las versiones de **sendmail** que son 8.9.0 o posteriores, tienen el reenvío deshabilitado por omisión, así que para ellos será necesario utilizar la base de datos de acceso para habilitar a los sitios locales para que puedan reenviar sus mensajes.

La idea general es muy sencilla. Cuando se recibe una conexión de entrada por SMTP, **sendmail** toma la información del encabezado de entrada y luego consulta la base de datos de acceso para ver si aceptará el contenido del mensaje.

La base de datos de acceso es una colección de reglas que describen qué acciones se deben tomar para los mensajes recibidos de los anfitriones nombrados. El fichero de control de acceso por omisión se llama `/etc/mail/access`. La tabla tiene un formato muy simple. Cada línea de la tabla contiene una regla de acceso. El lado izquierdo de cada regla es un patrón utilizado para comparar con el remitente de un mensaje de correo de entrada. Puede ser una dirección de correo completa, un nombre de anfitrión o una dirección IP. El lado derecho es la acción que se deberá tomar. Hay cinco tipos de acciones que se pueden seleccionar y son:

OK
: Aceptar el mensaje.

RELAY
: Aceptar los mensajes para este anfitrión o usuario aún si no provienen de nuestro anfitrión; esto es, aceptar que los mensajes sean reenviados hacia otros anfitriones desde este anfitrión.

REJECT
: Rechazar el correo con un mensaje genérico.

DISCARD
: Descartar el mensaje utilizando la propiedad `$#discard` del sistema de correo.

cualquier texto
: Contestar con un mensaje de error utilizando `###` como código de error (el cual deberá cumplir con el RFC−821) y cualquier texto será el mensaje.

Como ejemplo, el fichero `/etc/mail/access` podría ser como este:

```
friends@cybermail.com   REJECT
aol.com                 REJECT
207.46.131.30           REJECT
postmaster@aol.com      OK
linux.org.au            RELAY
```

Este ejemplo rechazará cualquier correo que se reciba desde friends@cybermail.com, cualquier anfitrión en el dominio **aol.com** y el anfitrión **207.46.131.30**. La siguiente regla aceptará correo electrónico desde postmaster@aol.com a pesar del hecho de que el dominio en sí mismo tiene una regla de rechazo. Y la última regla permite el reenvío de correo de cualquier anfitrión en el dominio **linux.org.au**.

Para habilitar la opción de la base de datos de acceso, se debe utilizar la siguiente declaración en su fichero `sendmail.mc`:

```
FEATURE(access_db)
```

La definición por omisión construye la base de datos con `hash -o /etc/mail/access`, lo que genera una base de datos con formato hash a partir de un fichero de texto simple. Esto es perfectamente adecuado en la mayor parte de las instalaciones. Hay otras opciones que se deben considerar si se intenta tener una gran base de datos de acceso. Consulte el libro de *sendmail* u otra documentación de **sendmail** para más detalles.

18.8.4.3. Exclusión usuarios de la recepción de correo

Si tiene usuarios o procesos automatizados que envían correo pero nunca necesitan recibirlo, es a veces útil no aceptar el correo destinado a ellos. Esto salva espacio de disco malgastado en almacenar correo que nunca será leído. la característica blacklist_recipients cuando se usa en combinación con la característica access_db le permite desactivar la recepción de correo para usuarios locales.

Para activar la característica, agregue las siguientes líneas al fichero `sendmail.mc`, si no están ya allí:

```
FEATURE(access_db)
FEATURE(blacklist_recipients)
```

Para desactivar la recepción de correo de un usuario local, simplemente añada sus datos a la base de datos de acceso. Normalmente podría usar el estilo de entrada ### que devolvería un mensaje de error con significado al remitente para que así sepa por qué el correo no se ha entregado. Esta característica se aplica igualmente bien a usuarios en dominios de correo virtuales, y debe incluir el dominio virtual de correo en la especificación de la base de datos de acceso. Algunas entradas de `/etc/mail/access` podrían semejarse a esto:

```
daemon           550 El demonio no acepta ni lee correo.
flacco           550 El correo de este usuario ha sido desactivado administrativamente.
grump@dairy.org 550 Correo desactivado para este destinatario.
```

18.8.5. Configurando el Hospedaje Virtual de Correo

El hospedaje virtual de correo proporciona a un anfitrión la capacidad de aceptar y entregar correo en nombre de varios dominios diferentes aunque estén en varios hospedajes de correo separados. Normalmente son los Proveedores de Aplicación de Internet los que explotan el hospedaje virtual en combinación con hospedaje virtual de webs, pero es sencillo de configurar y nunca se sabrá cuando tendrá la necesidad de hospedar virtualmente una lista de correo para su proyecto favorito de GNU/Linux, así que lo describiremos aquí.

18.8.5.1. Aceptar correo para otros dominios

Cuando **sendmail** recibe un mensaje de correo electrónico, compara el anfitrión de destino en las cabeceras del mensaje con el nombre del anfitrión local. Si coinciden, **sendmail** acepta el mensaje para entrega local; si difieren, **sendmail** puede decidir aceptar el mensaje e intentar reenviarlo al destino final (Vea Sección 18.8.4.2 más tarde en este capítulo para detalles sobre cómo configurar **sendmail** para aceptar correo para reenvío).

Si deseásemos configurar hospedeaje virtual de correo, la primera cosa que necesitamos hacer es convencer a **sendmail** de que deba aceptar también correo para los dominios que estamos hospedando. Afortunadamente, esto es muy sencillo de hacer.

La característica de **sendmail** use_cw_file nos permite especificar el nombre de un fichero donde almacenamos nombres de dominio para los que **sendmail** acepta correo. Para configurar la característica, añada la declaración de la característica a su fichero `sendmail.mc`:

```
FEATURE(use_cw_file)
```

El nombre predeterminado del fichero será `/etc/mail/local-host-names` para distribuciones que usen el directorio de configuración `/etc/mail/` o `/etc/sendmail.cw` para aquellas que no. Alternativamente, puede especificar el nombre y la localización del fichero anulando la macro `confCW_FILE` utilizando una variacion en:

```
define(`confCW_FILE',`/etc/virtualnames')
```

Para seguir con el nombre del fichero predeterminado, si deseásemos ofrecer hospedaje virtual a los dominios **bovine.net**, **dairy.org**, y **artist.org**, crearíamos un fichero `/etc/mail/local-host-names` semejante a:

```
bovine.net
dairy.org
artist.org
```

Cuando esto está hecho, y asumiendo que los registros DNS apropiados existen y apuntan éstos nombres de dominio a nuestro anfitrión, **sendmail** aceptará los mensajes de correo para estos dominios como si estuviesen dirigidos a nuestro propio nombre de dominio real.

18.8.5.2. Reenvío de correo hospedado virtualmente a otros destinos

La característica de **sendmail** virtusertable configura el soporte para la tabla de usuarios virtuales, donde configuramos el hospedaje de correo virtual. Las tablas de usuarios virtuales mapean el correo entrante destinado a algunos *usuarios@anfitrión* a algunos *otro_usuario@otro_anfitrión*. Puede pensar en esto como una característica de alias avanzada; una que opera usando no sólo el usuario de destino, sino también el dominio de destino.

Para configurar la característica virtusertable, añada la característica a su configuración `sendmail.mc` como se muestra:

```
FEATURE(virtusertable)
```

Por omisión, el fichero conteniendo las reglas para efectuar las traducciones será `/etc/mail/virtusertable`. Puede anular éste mediante el suministro de un argumento a la definición de la macro; consulte una referencia detallada de **sendmail** para aprender acerca de qué opciones están disponibles.

El formato de la tabla de usuarios virtuales es muy sencillo. En el lado izquierdo de cada línea hay un patrón representando el la dirección de destino original; al lado derecho, un patrón representando la dirección de correo a la que la se mapeará la dirección virtual hospedada.

El siguiente ejemplo muestra tres posibles tipos de entradas:

```
samiam@bovine.net      colin
sunny@bovine.net       darkhorse@mystery.net
@dairy.org             mail@jhm.org
@artist.org            $1@red.firefly.com
```

En este ejemplo, estamos hospedando virtualmente tres dominios: **bovine.net**, **dairy.org**, y **artist.org**.

La primera entrada reenvía el correo dirigido a un usuario en el dominio virtual **bovine.net** a un usuario local en la máquina. La segunda entrada reenvía el correo a un usuario en el mismo dominio virtual a un usuario en otro dominio. El tercer ejemplo reenvía todo el correo dirigido a cualquier usuario dentro del dominio virtual **dairly.org** a una sola dirección de correo remota. Finalmente, la última entrada reenvía cualquier correo a un usuario en el dominio virtual **artist.org** al mismo usuario en otro dominio; por ejemplo, julie@artists.org sería reenviado a julie@red.firefly.com.

18.9. Probando la Configuración

La orden **m4** procesa los ficheros de definición de la macro de acuerdo a sus propias reglas de sintaxis sin entender nada sobre la corrección de la sintaxis de **sendmail**; así que no debería haber ningún mensaje de error si tiene algo equivocado en el fichero de definición de macros. Por esta razón, es muy importante que pruebe su configuración. Afortunadamente, **sendmail** proporciona una manera relativamente fácil de hacer esto.

sendmail soporta un modo de prueba de direcciones que nos permite probar nuestra configuración e identificar cualquier error. En este modo de operación, invocamos **sendmail** desde la línea de órdenes, y él mismo nos pide una especificación del conjunto de reglas y una dirección de destino. **sendmail** entonces procesa esa dirección de destino usando las reglas especificadas, mostrando la salida de cada regla de reescritura mientras se realiza. Para poner **sendmail** en este modo, lo invocamos con el argumento `bt`:

```
# /usr/sbin/sendmail −bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
>
```

El fichero de configuración usado por omisión es el fichero `/etc/mail/sendmail.cf`; puede especificar uno alternativo usando el argumento `C`. Para probar nuestra configuración, necesitamos seleccionar varias direcciones para procesar que nos dirán que cada uno de los requerimientos de manipulación del correo se encuentran. Para ilustrar esto, trabajaremos a través de nuestra configuración UUCP más complicada mostrada en Ejemplo 18−2.

Primero, probaremos que **sendmail** es capaz de entregar correo a los usuarios locales del sistema. En estas pruebas, todas las direcciones serán reescritas al transporte de correo **local** en esta máquina:

```
# /usr/sbin/sendmail −bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac
rewrite: ruleset   3   input: isaac
rewrite: ruleset  96   input: isaac
rewrite: ruleset  96 returns: isaac
rewrite: ruleset   3 returns: isaac
rewrite: ruleset   0   input: isaac
rewrite: ruleset 199   input: isaac
rewrite: ruleset 199 returns: isaac
rewrite: ruleset  98   input: isaac
rewrite: ruleset  98 returns: isaac
rewrite: ruleset 198   input: isaac
rewrite: ruleset 198 returns: $# local $: isaac
rewrite: ruleset   0 returns: $# local $: isaac
```

Esta salida nos muestra cómo **sendmail** procesa el correo dirigido a **isaac** en este sistema. cada línea nos muestra qué información ha sido suministrada a un conjunto de reglas o el resultado obtenido del procesamiento por un conjunto de reglas. Le dijimos a **sendmail** que deseábamos emplear el conjunto de reglas 3 y 0 para procesar la dirección. El conjunto 0 es lo que se invoca normalmente y nosotros forzamos el conjunto 3 porque no se comprueba por omisión. La última línea nos muestra que el resultado del conjunto 0 en efecto reenvía el correo a **isaac** al transporte de correo **local**.

Lo siguiente que comprobaremos es el correo dirigido a nuestra dirección SMTP: **isaac@vstout.vbrew.com**. Deberíamos ser capaces de producir el mismo resultado final como en nuestro ejemplo último:

```
# /usr/sbin/sendmail −bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vstout.vbrew.com
rewrite: ruleset   3   input: isaac @ vstout . vbrew . com
```

```
rewrite: ruleset  96   input: isaac < @ vstout . vbrew . com >
rewrite: ruleset  96 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   3 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   0   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198 returns: $# local $: isaac
rewrite: ruleset   0 returns: $# local $: isaac
```

Otra vez la prueba se pasó. Lo siguiente es probar el correo a nuestra dirección estilo UUCP: **vstout!isaac**.

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 vstout!isaac
rewrite: ruleset   3   input: vstout ! isaac
rewrite: ruleset  96   input: isaac < @ vstout . UUCP >
rewrite: ruleset  96 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   3 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   0   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198 returns: $# local $: isaac
rewrite: ruleset   0 returns: $# local $: isaac
```

Esta prueba también se pasó. Estas pruebas confirman que cualquier correo recibido para los usuarios locales en nuestra máquina será entregado apropiadamente sin importar cómo está formateada la dirección. Si ha definido cualquier alias en su máquina, como hospedajes virtuales, debería repetir estas pruebas para cada uno de los nombres alternativos por los que este anfitrión se conoce para asegurarse que también funcionan correctamente.

Después, probaremos que el correo dirigido a otros anfitriones en el dominio **vbrew.com** se entregan directamente a ese anfitrión usando el transporte de correo SMTP:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vale.vbrew.com
rewrite: ruleset   3   input: isaac @ vale . vbrew . com
rewrite: ruleset  96   input: isaac < @ vale . vbrew . com >
rewrite: ruleset  96 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset   3 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset   0   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 199   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 199 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset  98   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset  98 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset 198   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 198 returns: $# smtp $@ vale . vbrew . com . /
    $: isaac < @ vale . vbrew . com . >
rewrite: ruleset   0 returns: $# smtp $@ vale . vbrew . com . /
    $: isaac < @ vale . vbrew . com . >
```

Podemos ver que esta prueba ha dirigido el mensaje al transporte SMTP para ser reenviado directamente al anfitrión **vale.vbrew.com** y especifica el usuario **isaac**. Esta prueba confirma que la definición `LOCAL_NET_CONFIG` funciona correctamente. Para que esta prueba sea satisfactoria, el nombre del anfitrión de destino debe ser resuelto correctamente, así que debe tener una entrada en nuestro fichero `/etc/hosts`, o en nuestro DNS local. Podemos ver qué ocurre si el nombre del anfitrión de destino no es capaz de resolverse especificando intencionadamente un anfitrión desconocido:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vXXXX.vbrew.com
rewrite: ruleset   3   input: isaac @ vXXXX . vbrew . com
rewrite: ruleset  96   input: isaac < @ vXXXX . vbrew . com >
vXXXX.vbrew.com: Name server timeout
rewrite: ruleset  96 returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset   3 returns: isaac < @ vXXXX . vbrew . com >
== Ruleset 3,0 (3) status 75
rewrite: ruleset   0   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 199   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 199 returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  98   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  98 returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 198   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  95   input: < uucp-new : moria > isaac </
    @ vXXXX . vbrew . com >
rewrite: ruleset  95 returns: $# uucp-new $@ moria $: isaac </
    @ vXXXX . vbrew . com >
rewrite: ruleset 198 returns: $# uucp-new $@ moria $: isaac </
    @ vXXXX . vbrew . com >
rewrite: ruleset   0 returns: $# uucp-new $@ moria $: isaac </
    @ vXXXX . vbrew . com >
```

Este resultado es muy diferente. Primero el conjunto de reglas 3 devuelve un mensaje de error indicando que el nombre del anfitrión no se pudo resolver. segundo, tratamos esta situación delegando en la otra característica clave de nuestra configuración: el anfitrión inteligente. El anfitrión inteligente estará para manipular cualquier correo que no se pueda entregar de otra manera. El nombre del anfitrión que especificamos en esta prueba era incapaz de ser resuelto y los conjuntos de reglas determinaron que el correo debería ser reenviado a nuestro anfitrión inteligente **moria** usando el transporte de correo **uucp−new**. Nuestro anfitrión inteligente quizá esté mejor conectado y sepa qué hacer con la dirección.

Nuesta prueba final asegura que cualquier correo dirigido a un anfitrión que no esté dentro de nuestro dominio se entrega a nuestro anfitrión inteligente. Esto debería producir un resultado similar a nuestro ejemplo previo:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@linux.org.au
rewrite: ruleset   3   input: isaac @ linux . org . au
rewrite: ruleset  96   input: isaac < @ linux . org . au >
rewrite: ruleset  96 returns: isaac < @ linux . org . au . >
rewrite: ruleset   3 returns: isaac < @ linux . org . au . >
rewrite: ruleset   0   input: isaac < @ linux . org . au . >
rewrite: ruleset 199   input: isaac < @ linux . org . au . >
rewrite: ruleset 199 returns: isaac < @ linux . org . au . >
rewrite: ruleset  98   input: isaac < @ linux . org . au . >
rewrite: ruleset  98 returns: isaac < @ linux . org . au . >
rewrite: ruleset 198   input: isaac < @ linux . org . au . >
rewrite: ruleset  95   input: < uucp-new : moria > isaac </
    @ linux . org . au . >
rewrite: ruleset  95 returns: $# uucp-new $@ moria $: isaac </
    @ linux . org . au . >
rewrite: ruleset 198 returns: $# uucp-new $@ moria $: isaac </
    @ linux . org . au . >
rewrite: ruleset   0 returns: $# uucp-new $@ moria $: isaac </
    @ linux . org . au . >
```

Los resultados de esta prueba indican que el nombre del anfitrión se resolvió, y que el mensaje podría ser encaminado a nuestro anfitrión inteligente. Esto prueba que nuestra definición `LOCAL_NET_CONFIG` funciona correctamente y que manejó ambos casos correctamente. Esta prueba es también exitosa, así que podemos felizmente asumir que nuestra configuración es correcta y usarla.

18.10. Ejecución de Sendmail

El demonio **sendmail** puede ejecutarse de dos formas. Una manera es ejecutarlo desde el demonio **inetd** y la otra, más usual, es ejecutar **sendmail** como un demonio en solitario. También es común para los programas de transporte invocar **sendmail** como una orden de usuario para aceptar el envío del correo generado localmente.

Cuando se ejecuta **sendmail** de modo solitario, se pone la instrucción en un fichero `rc`para que se inicie en tiempo de arranque. La sintaxis usada es normalmente:

```
/usr/sbin/sendmail -bd -q10m
```

El argumento `-bd` le dice a **sendmail** que se ejecute como un demonio. Se bifurcará y se ejecutará en segundo plano. El argumento `-q10m` le indica a **sendmail** que verifique su cola cada diez minutos. Puede escoger un tiempo diferente para verificar la cola.

Para ejecutar **sendmail** desde el demonio de red **inetd** podría usar una entrada como ésta:

```
smtp  stream  tcp nowait  nobody  /usr/sbin/sendmail -bs
```

El argumento `-bs` aquí le dice a **sendmail** que utilice el protocolo SMTP en stdin/stdout, que se requiere para utilizarlo con **inetd**.

La orden **runq** es usualmente un enlace simbólico al binario de **sendmail** y es una forma más conveniente que:

```
# sendmail -q
```

Cuando **sendmail** se invoca de esta manera, procesa cualquier correo esperando en la cola para ser transmitido. Cuando se ejecuta **sendmail** desde **inetd** debe crear un trabajo en **cron** que ejecute la orden **runq** periódicamente para asegurarse que el correo se sirve periódicamente.

Una tabla de **cron** útil sería similar a :

```
# Ejecutar la entrega de correo cada quince minutos
0,15,30,45     *     *     *     *     /usr/bin/runq
```

En muchas instalaciones **sendmail** procesa la cola cada 15 minutos como se muestra en nuestra tabla `crontab` de ejemplo, intentando transmitir cualquier mensaje allí.

18.11. Pistas y Trucos

Hay varias cosas que se pueden hacer para que la gestión de un sitio **sendmail** sea eficiente. Se proporcionan varias herramientas en el paquete **sendmail**; echemos un vistazo a las más importantes de éstas.

18.11.1. Gestión de la Cola de Correo

El correo es encolado en el directorio `/var/spool/mqueue` antes de ser transmitido. Este directorio se llama la cola de correo[183]. El programa **sendmail**proporciona un método para mostrar una lista formateada de todos los mensajes de correo en la cola y su estado.

La orden **/usr/bin/mailq** es un enlace simbólico al ejecutable de **sendmail** y se comporta idénticamente a:

```
# sendmail -bp
```

La salida muestra el identificador del mensaje (ID), su tamaño, la hora en que fue colocado en la cola, quién lo envió y un mensaje indicando su estado actual. El siguiente ejemplo muestra un mensaje de correo atascado en la cola con un problema:

```
$ mailq
                Mail Queue (1 request)
```

```
--Q-ID-- --Size-- -----Q-Time----- ------------Sender/Recipient------------
RAA00275      124 Wed Dec  9 17:47 root
                  (host map: lookup (tao.linux.org.au): deferred)
                                   terry@tao.linux.org.au
```

Este mensaje está todavía en la cola de correo porque la dirección IP del anfitrión de destino no se ha podido resolver.

Podemos forzar **sendmail** para procesar la cola ahora mediante la instrucción `/usr/bin/runq`.

La orden **runq** no produce ninguna salida. **sendmail** comenzará procesando la cola de correo en segundo plano.

18.11.2. Forzar a un Anfitrión Remoto a Procesar su Cola de Correo

Si usa una conexión a Internet temporal mediante marcado telefónico con una dirección IP *fija* y delega en un anfitrión MX para recibir su correo mientras está desconectado, encontrará útil forzar al anfitrión MX para que procese su cola de correo tan pronto como se establezca una conexión

Un pequeño programa **perl** se incluye con la distribución de **sendmail** que hace esto simple para los anfitriones de correo que lo soportan. El guión **etrn** tiene como mucho el mismo efecto en un anfitrión remoto como la instrucción **runq** tiene en el nuestro. Si invocamos la orden como se indica en este ejemplo:

```
# etrn vstout.vbrew.com
```

forzaremos al anfitrión **vstout.vbrew.com** que procese cualquier correo en espera para nuestra máquina local.

Típicamente añadiría esta instrucción al guión PPP `ip-up` de manera que se ejecute tan pronto como la conexión de red sea establecida.

18.11.3. Análisis de las Estadísticas de Correo

sendmail almacena datos de los volúmenes de tráfico de correo y alguna información sobre los anfitriones a los cuáles ha entregado correo. Hay dos instrucciones disponibles para mostrar esta información, **mailstats**, y **hoststat**.

18.11.3.1. mailstats

La orden **mailstats** muestra estadísticas del volumen de correo procesado por **sendmail**. La hora a la que la recogida de datos comenzó se imprime primero, seguida de una tabla con una columna para cada transporte configurado y una mostrando un resumen total de todo el correo. Cada línea presenta ocho puntos de información:

Campo	Significado
`M`	El número del transporte (protocolo de transporte)
`msgsfr`	El número de mensajes desde el transporte
`bytes_from`	Los Kbytes de correo desde el transporte
`msgsto`	El número de mensajes enviados al transporte
`bytes_to`	Los Kbytes de correo enviados al transporte
`msgsreg`	El número de mensajes rechazados
`msgsdis`	El número de mensajes descartados
`Mailer`	El nombre del transporte

Un ejemplo de la salida de la orden **mailstats** se muestra en Ejemplo 18−5.

Ejemplo 18−5. Ejemplo de la Salida de la Instrucción mailstats

```
# /usr/sbin/mailstats
    Statistics from Sun Dec 20 22:47:02 1998
     M   msgsfr  bytes_from   msgsto    bytes_to  msgsrej msgsdis  Mailer
     0        0          0K       19        515K        0       0  prog
     3       33        545K        0          0K        0       0  local
     5       88        972K      139       1018K        0       0  esmtp
    =============================================================
     T      121       1517K      158       1533K        0       0
```

Estos datos son recogidos si la opción *StatusFile* está activada en el fichero `sendmail.cf` y el fichero de estado existe. Típicamente añadiría lo siguiente a su fichero `sendmail.cf`:

```
    # status file
    O StatusFile=/var/log/sendmail.st
```

Para reiniciar la recogida de estadísticas, necesita hacer el fichero de estadísticas de longitud cero:

```
    > /var/log/sendmail.st
```

y reiniciar **sendmail**.

18.11.3.2. Hoststat

La instrucción **hoststat** muestra información sobre el estado de los anfitriones a los que **sendmail** ha intentado enviar correo. La instrucción **hoststat** es equivalente a invocar **sendmail** como:

```
sendmail −bh
```

La salida presenta cada anfitrión en su propia línea, y para cada uno la hora desde que se intentó la entrega a él, y el mensaje de estado recibido a esa hora.

Ejemplo 18−6 muestra la clase de salida que podría esperarse de la instrucción **hoststat**. Nótese que la mayor parte de los resultados indican una entrega exitosa. El resultado para **earthlink.net**, por otra parte, indica que la entraga no fue exitosa. El mensaje de estado puede algunas veces ayudar a determinar la causa del fallo. En este caso, la conexión expiró, probablemente porque el anfitrión estaba caído o inalcanzable a la hora en que la entrega se intentó.

Ejemplo 18−6. Ejemplo de Salida de la Instrucción hoststat

```
# hosts at
     −−−−−−−−−−−−− Anfitriónname −−−−−−−−−− How long ago −−−−−−−−−Results−−−−−−−−−
     mail.telstra.com.au                     04:05:41 250 Message accepted for
     scooter.eye−net.com.au               81+08:32:42 250 OK id=0zTGai−0008S9−0
     yarrina.connect.com.a                53+10:46:03 250 LAA09163 Message acce
     happy.optus.com.au                   55+03:34:40 250 Mail accepted
     mail.zip.com.au                         04:05:33 250 RAA23904 Message acce
     kwanon.research.canon.com.au         44+04:39:10 250 ok 911542267 qp 21186
     linux.org.au                         83+10:04:11 250 IAA31139 Message acce
     albert.aapra.org.au                     00:00:12 250 VAA21968 Message acce
     field.medicine.adelaide.edu.au       53+10:46:03 250 ok 910742814 qp 721
     copper.fuller.net                    65+12:38:00 250 OAA14470 Message acce
     amsat.org                             5+06:49:21 250 UAA07526 Message acce
     mail.acm.org                         53+10:46:17 250 TAA25012 Message acce
     extmail.bigpond.com                  11+04:06:20 250 ok
     earthlink.net                        45+05:41:09 Deferred: Connection time
```

La instrucción **purgestat** vacía los datos del anfitrión recogidos y es equivalente a invocar sendmail como:

```
# sendmail -bH
```

Las estadísticas continuarán creciendo hasta que usted las purge. Quizá quiera ejecutar periódicamente la instrucción **purgestat** para hacer más fácil buscar y encontrar entradas recientes, especialmente si tiene un sitio con mucho tráfico. Podría poner la orden en un fichero `crontab` para que se ejecute automáticamente, o tan sólo hacerlo usted mismo ocasionalmente.

Capítulo 19. Poner Eximen marcha

En este capítulo se ofrece una rápida introducción a la instalación de Exim y un vistazo a su funcionalidad. Aunque Exim es muy compatible con **sendmail** en su comportamiento, sus ficheros de configuración son completamente diferentes.

El principal fichero de configuración se llama normalmente `/etc/exim.conf` o `/etc/exim/config` en la mayoría de las distribuciones de GNU/Linux, o `/usr/lib/exim/config` en configuraciones más antiguas. Puede averiguar dónde se encuentra el fichero de configuración ejecutando la orden:

```
$ exim -bP configure_file
```

Quizá tenga que editar el fichero de configuración para reflejar los valores específicos de su sistema. En las configuraciones más comunes no hay mucho que cambiar, y una configuración que funcionase raramente tendría que modificarse.

De manera predeterminada, Exim procesa y envía todo el correo al instante. Si sufre un tráfico relativamente alto, también puede hacer que Exim recoja todos los mensajes en la *cola de correo* y sólo los procese a intervalos regulares.

Cuando gestiona correo en una red TCP/IP, Exim se ejecuta frecuentemente en modo demonio: durante el arranque del sistema `/etc/init.d/exim`lo invoca.[184] y se coloca en segundo plano, donde espera conexiones TCP entrantes por el puerto SMTP (normalmente el puerto 25). Esto resulta beneficioso cuando se tiene un tráfico significativo porque así Exim no tiene que iniciarse para cada conexión entrante. De manera alternativa, **inetd** puede gestionar el puerto SMTP y lanzar a Exim cuando haya una conexión en ese puerto. Esta configuración puede resultar útil si dispone de una memoria limitada y volúmenes bajos de tráfico.

Exim posee un complicado conjunto de opciones por línea de órdenes, incluyendo muchas que coinciden con las de sendmail. En vez de intentar reunir todas las opciones que se ajusten exactamente a sus necesidades, puede implementar los tipos más comunes de operaciones invocando órdenes tradicionales como **rmail** o **rsmtp**. Se trata de enlaces simbólicos a Exim (y si no, puede enlazarlos fácilmente). Cuando ejecute una de las órdenes, Exim comprobará el nombre que usó para invocarlo y él mismo usará las opciones adecuadas.

Hay dos enlaces a Exim que debería tener bajo cualquier circunstancia: **/usr/bin/rmail** y **/usr/sbin/sendmail**.[185] Cuando compone y envía un mensaje de correo electrónico con un cliente como **elm**, el mensaje se traslada a **sendmail** o a **rmail** para que lo envíen, que es por lo que **/usr/sbin/sendmail** y **/usr/bin/rmail** deberían apuntar a Exim. La lista de receptores para el mensaje se le pasa a Exim por la línea de órdenes.[186] Lo mismo sucede con el correo que entra por UUCP. Puede configurar los nombres de las rutas requeridas para que apunten a Exim introduciendo lo siguiente en el indicador del intérprete de órdenes:

```
$ ln -s /usr/sbin/exim /usr/bin/rmail
$ ln -s /usr/sbin/exim /usr/sbin/sendmail
```

Si quiere profundizar en los detalles de la configuración de Exim, debería consultar la especificación de Exim al completo. Si su distribución de GNU/Linux favorita no la incluye, puede obtenerla de las fuentes de Exim, o leerla en línea desde el sitio web de Exim en: http://www.exim.org.

19.1. Ejecución de Exim

Para ejecutar Exim, primero debe decidir si quiere que gestione los mensajes por SMTP entrantes ejecutándose como un demonio separado, o si quiere que **inetd** se encargue del puerto SMTP invocando a Exim sólo cuando se solicite una conexión SMTP desde el cliente. Normalmente preferirá que funcione como demonio porque eso cargará menos el anfitrión que iniciar Exim una y otra vez en cada conexión. Como el servidor de correo traslada la mayoría del correo entrante directamente a los usuarios, debería encargarle la gestión a **inetd** en la mayoría del resto de anfitriones.

Sea cual sea el modo de funcionamiento para cada anfitrión, tiene que asegurarse de tener la siguiente entrada en su fichero /etc/services:

```
smtp            25/tcp          # Simple Mail Transfer Protocol
```

Esto define el número del puerto TCP que se usará en las comunicaciones SMTP. El puerto 25 es el estándar definido por el RFC de Números Asignados (RFC−1700).

Cuando se ejecuta como demonio, Exim se coloca en segundo plano y espera conexiones por el puerto SMTP. Cuando se da una conexión se bifurca y el proceso hijo lleva a cabo una conversación SMTP con el proceso compañero del anfitrión que llama. El demonio Exim se inicia normalmente invocándolo desde el guión rc durante el arranque por medio de la siguiente orden:

```
/usr/sbin/exim −bd −q15m
```

La opción bd activa el modo demonio y q15m hace que procese los mensajes que se hayan acumulado en la cola durante los últimos quince minutos.

Si quiere usar **inetd** su fichero /etc/inetd.conf debería contener una línea como ésta:

```
smtp    stream  tcp nowait  root  /usr/sbin/exim  in.exim −bs
```

Recuerde que tiene que hacer que **inetd** relea inetd.conf enviándole una señal HUP tras realizar cualquier cambio.[187]

Los modos demonio e **inetd** se excluyen mutuamente. Si ejecuta Exim como demonio, debería asegurarse de descomentar cualquier línea de inetd.conf para el servicio smtp. De manera equivalente, cuando **inetd** se encargue de Exim, asegúrese de que ningún guión rc inicie el demonio Exim.

Puede comprobar si Exim está instalado correctamente para recibir mensajes SMTP entrantes conectándose por telnet al puerto SMTP de su anfitrión. Una conexión con éxito al servidor SMTP sería algo así como esto:

```
$ telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 richard.vbrew.com ESMTP Exim 3.13 #1 Sun, 30 Jan 2000 16:23:55 +0600
quit
221 richard.brew.com closing connection
Connection closed by foreign host.
```

Si esta prueba no genera la leyenda del SMTP (la línea que comienza con el código 220), compruebe si está ejecutando un demonio Exim o si **inetd** está configurado correctamente. Si eso no lo resuelve mire en los ficheros de bitácora de Exim (que se describen a continuación) en caso de que haya algún error en el fichero de configuración de Exim.

19.2. Si el correo no llega a su destino

Hay disponibles algunas características por si tiene problemas con la instalación. El primer lugar donde mirar es en los ficheros de registro de Exim. En los sistemas GNU/Linux normalmente se encuentran en /var/log/exim/log y se llaman exim_mainlog, exim_rejectlog y exim_paniclog. En otros sistemas operativos, a menudo se guardan en /var/spool/exim/log. Puede averiguar dónde se encuentran estos ficheros mediante la orden:

```
exim −bP log_file_path
```

El registro principal lista todas las transacciones, el registro de rechazos contiene mensajes que se han rechazado por cuestiones de políticas y el registro de avisos mensajes relacionados con errores de configuración y problemas similares.

Abajo se muestran entradas típicas del registro principal. Cada entrada del registro es una única línea de texto que comienza con una fecha y una hora. Aquí se han separado en varias líneas para que quepan en la página:

```
2000-01-30 15:46:37 12EwYe-0004WO-00 <= jack@vstout.vbrew.com
  H=vstout.vbrew.com [192.168.131.111] U=exim P=esmtp S=32100
  id=38690D72.286F@vstout.vbrew.com
2000-01-30 15:46:37 12EwYe-0004WO-00 => jill <jill@vbrew.com>
  D=localuser T=local_delivery
2000-01-30 15:46:37 12EwYe-0004WO-00 Completed
```

Estas entradas muestran que un mensaje desde jack@vstout.vbrew.com para jill@vbrew.com se envió con éxito a un buzón del anfitrión local. Las llegadas de mensajes se marcan con <= y los envíos con =>.

Hay dos tipos de errores de envío: permanente y temporal. Un error de envío permanente aparece en una entrada de registro como ésta marcado con `**`:

```
2000-01-30 14:48:28 12EvcH-0003rC-00 ** bill@lager.vbrew.com
  R=lookuphost T=smtp: SMTP error from remote mailer after RCPT TO:
  <bill@lager.vbrew.com>: host lager.vbrew.com [192.168.157.2]:
  550 <bill@lager.vbrew.com>... User unknown
```

Tras un error como éste, Exim envía un informe sobre el error en el envío, a menudo conocido como *mensaje de rebote* de vuelta al remitente.

Los errores temporales se marcan con ==:

```
2000-01-30 12:50:50 12E9Un-0004Wq-00 == jim@bitter.vbrew.com
  T=smtp defer (145): Connection timed out
```

Este error es típico de situaciones en las que Exim reconoce adecuadamente que un mensaje debería enviarse a un anfitrión remoto pero no es capaz de conectar con el servicio SMTP en ese anfitrión. El anfitrión puede no encontrarse operativo o puede haber un problema en la red. Cuando un mensaje se *pospone* como en este caso, permanece en la cola de Exim y se reintenta su envío a intervalos. De todos modos, si no consigue enviarse durante bastante tiempo (normalmente varios días), ocurre un error permanente y el mensaje se devuelve.

Si no es capaz de localizar su problema a partir de los mensajes de error que genera Exim, quizá quiera activar los mensajes de depuración. Puede hacer esto por medio de la variable `d`, seguida opcionalmente por un número que explica el nivel de verbosidad (un valor de 9 le proporciona el máximo de información). Exim muestra entonces por pantalla un informe sobre sus operaciones, que quizá le den más pistas sobre lo que que pueda ir yendo mal.

19.3. Compilar Exim

Exim aún se encuentra en desarrollo activo. La versión de Exim que se incluye en las distribuciones de GNU/Linux probablemente no sea la última. Si necesita una prestación o un error corregido en las últimas versiones, tendrá que obtener una copia del código fuente y compilarlo usted mismo. Puede encontrar la última versión de Exim por medio de la página web de Exim en http://www.exim.org.

GNU/Linux es uno de los muchos sistemas operativos que soportan las fuentes de Exim. Para compilar Exim bajo GNU/Linux debería editar el fichero `src/EDITME` y colocar el resultado en un fichero llamado `Local/Makefile`. Hay comentarios en `src/EDITME` que le dirán para qué son los diferentes ajustes. Ejecute entonces **make**. Acuda al manual de Exim para información más detallada sobre cómo construir Exim a partir de las fuentes.

19.4. Modos de Envío de Correo

Como ya se ha dicho anteriormente, Exim es capaz de enviar los mensajes de inmediato o guardarlos en una cola para procesarlos más tarde. Todo el correo entrante se guarda en el directorio`input` bajo `/var/spool/exim`. Cuando el encolado no se encuentra activo, se inicia un proceso de envío para cada mensaje en cuanto llega. De otro modo, se queda en la cola hasta que un proceso *queue−runner* lo recoge. El encolado puede ajustarse como incondicional mediante `queue_only` en el fichero de configuración, o puede estar condicionado por la carga del sistema minuto a minuto mediante:

```
queue_only_load = 4
```

que hace que los mensajes se encolen si la carga del sistema excede 4.[188]

Si su anfitrión no está conectado permanentemente a Internet, quizá quiera activar el encolado para direcciones remotas, permitiendo al mismo tiempo que Exim realice los envíos locales de inmediato. Puede hacer esto poniendo:

```
queue_remote_domains = *
```

en el fichero de configuración.

Si activa cualquier tipo de encolado tiene que asegurarse de que las colas se comprueba de manera regular, probablemente cada 10 ó 15 minutos. Aún sin opciones explícitas de encolado, las colas necesitan comprobarse por si se hubieran pospuesto mensajes a causa de fallos temporales de envío. Si ejecuta Exim como demonio tendrá que añadir la opción `q15m` en la línea de órdenes para procesar la cola cada 15 minutos. También puede invocar a **exim q** desde **cron** a estos intervalos.

Puede mostrar la cola de correo invocando a Exim con la opción `bp`. De manera equivalente, puede enlazar **mailq** con Exim e invocar **mailq**:

```
$ mailq
 2h   52K 12EwGE-0005jD-00 <sam@vbrew.com>
        D bob@vbrew.com
          harry@example.net
```

Esto muestra un único mensaje desde sam@vbrew.com para dos receptores en la cola de mensajes. Se ha enviado con éxito a bob@vbrew.com pero aún no se ha enviado a harry@example.net aunque ha estado en la cola dos horas. El tamaño del mensaje es de 52K y la identificación que usa Exim para este mensaje es `12EwGE-0005jD-00`. Puede averiguar por qué aún no se ha completado el envío mirando en el fichero de registro individual del mensaje, que se guarda en el directorio `msglog` dentro del directorio de la cola de Exim. La opción`Mvl` es una manera sencilla de hacer esto:

```
$ exim Mvl 12EwGE-0005jD-00
2000-01-30 17:28:13 example.net [192.168.8.2]: Connection timed out
2000-01-30 17:28:13 harry@example.net: remote_smtp transport deferred:
  Connection timed out
```

Los ficheros de registro individuales mantienen una copia de las entradas del registro para cada mensaje por lo que puede inspeccionarlas fácilmente. Puede extraer esa misma información del fichero de registro principal mediante la utilidad **exigrep**:

```
$ exigrep 12EwGE-0005jD-00 /var/log/exim/exim_mainlog
```

Eso puede llevarle bastante tiempo, especialmente en un sistema ocupado en el que los ficheros de registro pueden hacerse bastante grandes. La utilidad **exigrep** es apropiada cuando se busca información sobre más de un mensaje. Su primera variable es una expresión regular, y toma todas las líneas del registro de cualquier mensaje en las que coincida al menos una línea con la expresión. Por esto, puede usarse para consultar todos los mensajes en busca de una dirección o de un anfitrión específicos.

Puede echar un vistazo general a lo que está haciendo un Exim en ejecución mediante la orden **tail** sobre el fichero de registro principal. Otra manera de hacer esto es ejecutar la utilidad **eximon** que viene con Exim. Se trata de una aplicación que muestra una pantalla donde aparece el registro principal en tiempo real, además de mostrarle una lista con los mensajes que estén esperando ser enviados, así como unas gráficas de barras acerca de la actividad de envío.

19.5. Otras opciones de configuración

He aquí unas pocas de las opciones más utiles que puede ajustar en el fichero de configuración:

message_size_limit
: Esta opción limita el tamaño de los mensajes que Exim aceptará.

return_size_limit
: Esta opción limita la cantidad de un mensaje entrante que Exim devolverá como parte de un mensaje de rebote.

deliver_load_max
: Si la carga del sistema excede el valor dado a esta opción, se suspenden todos los envíos de correo aunque seguirán aceptándose mensajes para enviar.

smtp_accept_max
: Éste es el número máximo de llamadas SMTP entrantes simultáneas que Exim está preparado para aceptar.

log_level
: Esta opción controla la cantidad de información que se escribe en el registro. Hay también algunas opciones cuyos nombres comienzan con `log_` que controlan el registro de información específica.

19.6. Encaminamiento y envío de mensajes

Exim divide el envío de correo en tres tareas diferentes: el encaminado, el direccionamiento y el transporte. Hay un número de módulos de código para para cada tarea, siendo cada uno configurable por separado. En el fichero de configuración normalmente se instalan un número de diferentes encaminadores, direccionadores y transportes.

Los encaminadores resuelven direcciones remotas, determinando a qué anfitrión debería enviarse un mensaje y qué transporte debería usarse. En los anfitriones conectados a Internet a menudo hay un solo encaminador, que lleva a cabo la resolución buscando el dominio en el DNS. De forma alternativa, puede haber un encaminador que se encargue de las direcciones destinadas a los anfitriones de una LAN local, y un segundo para enviar cualquier otra dirección a un *anfitrión inteligente*; por ejemplo, el servidor de correo de un PSI.

Las direcciones locales se envían a los direccionadores, de los que suele haber una gran cantidad, que se encargan de la gestión de los "alias" y de los reenvíos así como de la identificación de los buzones locales. Las listas de correo pueden gestionarse mediante direccionadores de "aliasing" o de reenvío. Si una dirección se renombra o se reenvía, cada dirección generada la gestionan los encaminadores o los direccionadores, según sea necesario, de manera independiente. El caso más común, de lejos, será el envío a un buzón, pero los mensajes también pueden enviarse a una orden o adjuntarse a un fichero diferente al buzón predeterminado.

Un transporte es responsable de implementar un método de envío; por ejemplo, enviar el mensaje mediante una conexión SMTP o añadirlo a un buzón específico. Los encaminadores y los direccionadores eligen qué transporte usar para cada dirección receptoras. Si un transporte falla, Exim genera un mensaje de rebote o pospone el envío para intentar realizarlo más tarde.

Con Exim goza de una gran libertad para configurar estas tareas. Hay disponibles controladores para cada una de ellas, entre los que puede escoger aquellos que necesite. Sólo tiene que describírselos a Exim en diferentes secciones de su fichero de configuración. Primero se definen los transportes, seguidos de los direccionadores y después los encaminadores. No hay nada integrado de manera predeterminada, aunque Exim se distribuye con un fichero de configuración predeterminada que cubre casos sencillos. Si quiere cambiar la política de encaminado de Exim o modificar un transporte, lo más sencillo será partir del fichero de configuración predeterminada para realizar los cambios en vez de intentar crear una configuración completa desde cero.

19.6.1. Mensajes de Encaminamiento

Cuando se da una dirección de envío, Exim comprueba primero si el dominio es uno de los que se maneja en el anfitrión con una lista en la opción de configuración `local_domains`. Si no se ha configurado esta opción, se usa el nombre del anfitrión local como el único dominio local. Si el dominio es local, la dirección la manejan los direccionadores. De otro modo, se pasa a los encaminadores para que averigüen a qué anfitrión reenviar el mensaje.[189]

19.6.2. Enviar mensajes a direcciones locales

De manera más común, una dirección local es simplemente el nombre de "login" de un usuario, en cuyo caso el mensaje se envía al buzón del usuario, `/var/spool/mail/`*`nombre-de-usuario`*. Otros casos incluyen los alias, los nombres de listas de correo y los reenvíos del usuario. En estos casos, la dirección local se expande en una nueva lista de direcciones que pueden ser a su vez locales o remotas.

Aparte de estas direcciones normales, Exim puede manejar otro tipo de destinos para los mensajes locales, como nombres de ficheros y órdenes de "pipe". Cuando se envía a un fichero, Exim adjunta el mensaje creando ese fichero si es necesario. Los destinos de fichero y "pipe" no son direcciones propiamente dichas por lo que no puede enviar correo a, pongamos por caso, /etc/passwd@vbrew.com y esperar que se sobreescriba el fichero de las contraseñas; los envíos a un fichero específico sólo son válidos si vienen de fichero de reenvío o de alias. Tenga en cuenta, no obstante, que /etc/passwd@vbrew.com es una dirección de correo sintácticamente válida, pero si Exim la recibe buscaría (típicamente) un usuario cuyo "login" fuese /etc/passwd, y al no encontrar ninguno el mensaje rebotaría.

En una lista de alias o en un fichero de reenvío, un *nombre de fichero* es cualquier cosa que comience con una barra (/)y que no pueda entenderse como una dirección de correo plenamente cualificada. Por ejemplo, `/tmp/junk` en un fichero de reenvío o de alias es interpreta como un nombre de fichero, pero /tmp/junk@vbrew.com es una dirección de correo, aunque no parece muy útil. De todas maneras, pueden verse direcciones válidas de este tipo cuando se envía correo por medio de pasarelas X.400 porque las direcciones X.400 comienzan con una barra.

De manera similar, una *orden de "pipe"* puede ser una orden de Unix precedida por el símbolo (|), a menos que esta cadena pueda entenderse como una dirección de correo con dominio válida. A menos que haya cambiado la configuración, Exim no usa consola alguna para ejecutar la orden; sino que la divide en un nombre de orden y en sus variables y las ejecuta directamente. El mensaje se emplea como entrada estándar para esa orden.

Por ejemplo, para conducir una lista de correo a un grupo de noticias local, puede usar un guión de shell de nombre **gateit**, e instalar un alias local que envíe todos los mensajes de esta lista al guión usando `|gateit`. Si la línea de órdenes contiene una coma, ella y el símbolo de "pipe" precedente deben entrecomillarse.

19.6.2.1. Usuarios locales

Una dirección local denota comúnmente un buzón de usuario. Éste normalmente se encuentra en `/var/spool/mail` y tiene el nombre del usuario, quien también es el propietario del fichero. Si no existe, Exim lo crea.

En algunas configuraciones, el grupo se cambia al grupo del usuario y el modo es 0600. En estos casos, los procesos de envío se ejecutan como un usuario, y el usuario puede borrar el buzón completamente. En otras configuraciones, el grupo del buzón es mail, y tiene el modo 660; los procesos de envío se ejecutan bajo un uid y un grupo del sistema mail, y los usuarios no pueden borrar sus buzones, aunque sí pueden vaciarlos.

Tenga en cuenta que aunque `/var/spool/mail` es habitualmente el lugar estándar en el que colocar los ficheros de los buzones, algunos programas de correo pueden compilarse para usar rutas diferentes, por ejemplo, `/usr/spool/mail`. Si el envío a los usuarios de su anfitrión falla de manera consistente, debería ver si le sirve de algo crear un enlace simbólico a `/var/spool/mail`.

Las direcciones MAILER−DAEMON y postmaster deberían aparecer normalmente en su fichero de alias, expandiéndose en la dirección de correo del administrador del sistema. MAILER−DAEMON lo usa Exim como dirección del remitente en los mensajes de rebote. También se recomienda que root se instale como un alias para el administrador, especialmente cuando los envíos se ejecutan bajo permisos de los usuarios receptores para evitar que no se ejecute ningún envío como root.

19.6.2.2. Reenvío

Los usuarios pueden redirigir su correo a direcciones alternativas creando un fichero `.forward` en sus directorios home. Éste contiene una lista de receptores separados por comas y/o nuevas líneas. Se leen e interpetan todas las líneas del fichero. Puede usarse cualquier tipo de dirección. Un ejemplo práctico de un fichero `.forward` para las vacaciones podría ser:

```
    janet, "|vacation"
```

En otras descripciones de ficheros `.forward`, puede ver el nombre de usuario al comienzo precedido por una barra invertida. Esto era necesarios en algunos MTAs antiguos para detener la búsqueda de `.forward` para un nombre nuevo, lo que podía conducir a un bucle infinito. La barra invertida no es necesaria en Exim, que evita automáticamente bucles de este tipo.[190] De todos modos, se permite una barra invertida y, de hecho, supone una diferencia en configuraciones en las que se manejan muchos dominios de una vez. Sin la barra invertida, un nombre de usuario no cualificado se cualifica con un dominio predeterminado; con una barra invertida se preserva el dominio entrante.

La primera dirección del fichero .forward envía el mensaje entrante al buzón de **janet**, mientras que la orden **vacation** devuelve una breve notificación al remitente. [191]

Además de soportar ficheros de reenvío tradicionales, Exim puede configurarse para que soporte ficheros más complejos conocidos como *filtros*. En vez de ser simplemente una lista con direcciones de reenvío, un fichero de filtro puede contener pruebas sobre el contenido de los mensajes entrantes de manera que, por ejemplo, pueda hacerse que se reenvíen únicamente aquellos mensajes cuyo título contenga la palabra urgente. Los administradores de sistemas deben decidir si permitirán o no esta flexibilidad a los usuarios.

19.6.3. Ficheros de alias

Exim es capaz de gestionar ficheros de alias compatibles con los ficheros de alias del **sendmail** de Berkeley. Las entradas del fichero de alias pueden tener la siguiente forma:

```
alias: receptores
```

`receptores` es una lista de direcciones separadas por comas que se sustituirán por el alias. La lista de receptores puede continuarse en nuevas líneas si la línea siguiente comienza con un espacio en blanco.

Una propiedad especial permite a Exim gestionar listas de correo que se hayan especificado separadamente en el fichero de alias: si especifica `:include:`*`nombre de fichero`* como un receptor, Exim lee el fichero especificado y sustituye su contenido como una lista de receptores. Más adelante en este capítulo en Sección 19.6.4se muestra una manera alternativa de gestionar listas de correo.

El principal fichero de alias es /etc/aliases. Si hace que cualquiera o que cualquier grupo pueda modificar este fichero, Exim rechazará usarlo y pospondrá los envíos locales. Puede controlar la prueba que realiza con los permisos del fichero poniendo modemask en el direccionador system_aliases.

Esto es un fichero aliases de ejemplo:

```
# vbrew.com fichero /etc/aliases
hostmaster: janet
postmaster: janet
usenet: phil
# La lista de correo de desarrollo.
development: joe, sue, mark, biff,
        /var/mail/log/development
owner-development: joe
# Los anuncios de interés general se envían a todo
# el equipo
announce: :include: /etc/Exim/staff,
        /var/mail/log/announce
owner-announce: root
# la lista de correo ppp se traslada a un grupo local de noticias
ppp-list: "|/usr/local/bin/gateit local.lists.ppp"
```

Cuando hay nombres de ficheros y órdenes por "pipe" en un fichero de alias, como aquí, Exim necesita que le digan bajo qué usuario ha de ejecutar los envíos. La opción user (y posiblemente group, también debe proporcionarse en el fichero de configuración de Exim, ya sea en el direccionador que maneja los alias o en los transportes a los que dirige estos elementos.

Si ocurre un error al enviarse a una dirección generada a partir del fichero aliases, Exim enviará un mensaje de rebote al remitente del mensaje, como es habitual, pero esto quizá no resulte apropiado. Puede usar la opción errors_to para especificar que los mensajes de rebote se envíen a otro usuario: por ejemplo, al administrador del correo.

19.6.4. Listas de correo

En vez de con el fichero aliases, las listas de correo también pueden gestionarse mediante un direccionador forwardfile. Las listas se mantienen en un único directorio como /etc/exim/lists/, y un fichero lists/nag-bugs describe una lista de correo llamada nag–bugs. Esto debería contener las direcciones de los miembros separadas por comas o por nuevas líneas. Las líneas que comienzan con una almohadilla (#) se tratan como comentarios. Un sencillo direccionador para usar esos datos sería como sigue:

```
lists:
  driver = forwardfile
  file = /etc/exim/lists/${local_part}
  no_check_local_user
  errors_to = ${local_part}-request
```

Cuando se ejecuta este direccionador, los valores de las opciones file y errors_to se *expanden*. La expansión hace que ciertas partes de las cadenas que comienzan con un símbolo del dólar se sustituyan cada vez que se usa la cadena. El tipo de expansión más sencillo es la inserción del valor de una de las variables de Exim, y esto es lo que está sucediendo aquí. La subcadena ${local_part} sustituye el valor de $local_part, que es la parte local de las direcciones que se estén procesando.

Para cada lista de correo, debería existir un usuario (o un alias o una lista de correo) llamado *listname-request*; se informa a esta dirección de cualquier error al resolver una dirección o al enviar un mensaje a un miembro de la lista.

19.7. Protegerse contra el "spam"

El `spam` o correo con fines comerciales no solicitado es un molesto problema para muchos usuarios. Se ha formado un proyecto para acabar con este problema conocido como Sistema de Protección contra los Abusos en el Correo (MAPS), y se ha construido un mecanismo que reduce el problema, conocido como Lista de Agujeros Negros en Tiempo Real (RBL). Puede obtener información sobre el funcionamiento de MAPS RBL a partir de su documentación en línea en http://maps.vix.com/rbl/. La idea es sencillamente simple. Los sitios que se encuentran generando spam se añaden a una base de datos y agentes de transferencia de correo como Exim son capaces de consultar la base de datos para confirmar si un remitente es o no un "spammer" antes de aceptar correo de él.

Desde el advenimiento de la RBL, se han creado muchas otras listas. Una de las más útiles es la Lista de Marcado (DUL), que lista las direcciones IP de anfitriones conectados a la red mediante acceso por llamada telefónica. Éstas deberían enviar normalmente el correo a los servidores de correo de sus PSIs. Muchos sitios bloquean el correo desde llamadas externas porque cuando un anfitrión de este tipo evita al servidor de su propio PSI normalmente no se trata de algo bueno.

Exim ofrece soporte para listas en tiempo real y otras listas negras. Esto se configura de manera muy sencilla. Para activarlo, añada las siguientes líneas a su fichero `/etc/exim.conf`:

```
# Vixie / MAPS RBL (http://maps.vix.com/rbl)
rbl_domains = rbl.maps.vix.com : dul.maps.vix.com
```

Este ejemplo comprueba tanto la RBL como la DUL, rechazando cualquier mensaje desde anfitriones que se encuentren en cualquiera de esas listas. La opción `rbl_hosts` le permite especificar grupos de anfitriones a los que se aplica (o no) la comprobación RBL. La configuración predeterminada es:

```
rbl_hosts = *
```

lo que significa que todos los anfitriones se encuentran sujetos a la comprobación de la RBL. Si quisiera saltarse la lista negra y aceptar correo de un anfitrión específico sin que se realizara la comprobación RBL podría usar, por ejemplo:

```
rbl_hosts = ! nocheck.example.com : *
```

El signo de exclamación antes del primer elemento de la lista indica un elemento negado: si el anfitrión que llama es nocheck.example.com, coincidirá con este elemento. Pero a causa de la negación, la comprobación RBL no se lleva a cabo. Cualquier otro anfitrión coincidirá con el segundo elemento de la lista.

19.8. Instalación UUCP

Exim no posee código específico para transportar correo mediante UUCP ni soporta prohibición de direcciones UUCP. De todos modos, si se usa el direccionamiento de dominios, Exim puede interactuar con UUCP de una manera bastante sencilla. He aquí un fragmento de configuración para enviar ciertos dominios a UUCP tomado de una instalación real:

```
# Transporte
uucp:
  driver = pipe
  user = nobody
  command = "/usr/local/bin/uux -r - \
    ${substr_-5:$host}!rmail ${local_part}"
  return_fail_output = true

# Encaminador
uucphost:
  transport = uucp
  driver = domainlist
  route_file = /usr/exim/uucphosts
  search_type = lsearch
```

En un fichero de configuración completo, el transporte se insertaría entre los otros transportes, y el encaminador definido probablemente como el primer encaminador. El fichero `/usr/exim/uucphosts` contiene entradas como ésta:

```
darksite.example.com:          darksite.UUCP
```

que se interpreta como, Enviar el correo dirigido al dominio **darksite.example.com** al anfitrión UUCP **darksite**. Puede realizar esta configuración de manera más sencilla sin que el encaminador añada el sufijo .UUCP a **darksite** sólo para que el transporte se lo quite de nuevo, pero este método resulta útil porque así queda más clara la distinción entre el nombre del dominio **darksite.example.com** y el nombre del anfitrión UUCP **darksite**.

Cuando el encaminador se encuentre con un dominio presente en el fichero de rutado, enviará la dirección al transporte UUCP, que se la enviará a su vez a la orden **uux** (descrita en Capítulo 16). Si ocurre algún problema **uux** generará alguna salida y terminará con un código de error distinto a cero. Use la opción `return_fail_output` para asegurarse de que la salida se devuelve al remitente.

Si los mensajes UUCP entrantes se agrupan en ficheros en formato SMTP por lotes, pueden pasarse directamente a Exim usando una orden como ésta:

```
exim -bS </var/uucp/incoming/001
```

De toda formas, no todo es tan sencillo. Cuando Exim recibe un mensaje localmente, insiste en que el remitente es el usuario conectado que lo haya enviado, pero para un lote UUCP queremos que los remitentes se tomen de los mensajes entrantes. Exim hará esto si el proceso que lo llama se está ejecutando como un *usuario en el que se confía*. Si especifica que el correo UUCP entrante lo gestione por ejemplo un usuario llamado uucp, necesitará especificar:

```
trusted_users = uucp
```

en el fichero de configuración de Exim para asegurarse de que las direcciones de los remitentes se tratan de manera correcta.

Capítulo 20. Noticias

Netnews, o Usenet News, sigue siendo uno de los servicios más importantes y favorablemente valorados en las redes de computadoras de hoy. Rechazado por algunos como un fango de correo electrónico comercial no solicitado y pornografía, Netnews todavía mantiene varios casos de los grupos de discusión de alta calidad que lo hizo un recurso crítico antes de la existencia de la web. Incluso en los tiempos actuales de un billón de páginas web, Netnews todavía es una fuente para la ayuda en línea y a la comunidad en muchos temas.

20.1. Historia de Usenet

La idea de noticias en la red nació en 1979 cuando dos estudiantes graduados, Tom, Truscott y Jim Ellis, pensaron en el uso de UUCP para conectar las máquinas con la finalidad de intercambiar información entre los usuarios de Unix. Instalaron una pequeña red compuesta de sólo tres máquinas en Carolina del Norte.

El tráfico inicialmente, fue manejado por algunos guiones de shell (después se reescribieron en C), pero nunca se dieron al público. Fueron reemplazados rápidamente por A News, la primera edición pública de software para noticias.

"ANews" no fue diseñado para manejar más de unos pocos artículos por grupo y día. Cuando el volumen de información creció, fue vuelto a escribir por Mark Horton y Matt Glickman que lo llamaron versión B (llamado también BNews). La versión 2.1 de BNews en 1982 fue la primera edición pública. Fue creciendo gradualmente, con la adición de nuevas prestaciones. La versión actual es BNews. Lentamente se está quedando obsoleta, su último mantenedor oficial se cambió a INN.

Geoff Collyer y Henry Spencer reescribieron BNews y lo lanzaron en 1987; ésta es la versión C, o CNews. Desde esta versión, ha habido varios parches para CNews, siendo el más notorio el CNews Performance Release. En sitios que llevan gran número de grupos, la sobrecarga que involucraba la frecuente invocación de **relaynews** , que es el responsable de despachar los artículos entrantes a otros hosts, es significativa. La Performance Release agrega una opción a **relaynews** que le permite ejecutarse en modo *daemon*, es decir, que se pone a sí misma como tarea de fondo. En la mayoría de las distribuciones actuales de GNU/Linux se incluye la Performance Release de CNews. Describimos CNews en detalle en Capítulo 21.

Todas las versiones hasta la "C" están principalmente diseñadas para su uso en redes UUCP aunque también sirven para otros entornos. La transferencia eficaz de noticias sobre redes tipo TCP/IP o DECNet requiere un nuevo esquema. Así en 1986, se introdujo el protocolo NNTP, *Network News Transfer Protocol*. Está basado en las conexiones de la red y especifica varias instrucciones para transferir y recuperar interactivamente los artículos.

En la Red hay disponibles varias aplicaciones basadas en NNTP. Una de ellas es el **nntpd** de Brian Barber y Phil Lapsley, que puede usar para proporcionar servicio de lectura a varios hosts dentro una red local. **nntpd** se diseñó para complementar paquetes de noticias como BNews y C news y darles prestaciones de NNTP. Si usted quiere usar NNTP con el servidor CNews, debe leer Capítulo 22 que explica cómo configurar el demonio **nntpd** y lo ejecuta con CNews.

Un paquete alternativo de apoyo a NNTP es INN, o *INternet News*. No es solamente una interfaz, sino un sistema de noticias por derecho propio. Comprende un sofisticado demonio de noticias que se puede mantener eficientemente y es por ello el servidor de noticias de elección por muchos sitios de Internet. Lo discutimos en detalle en Capítulo 23.

20.2. Pero, ¿qué es Usenet después de todo?

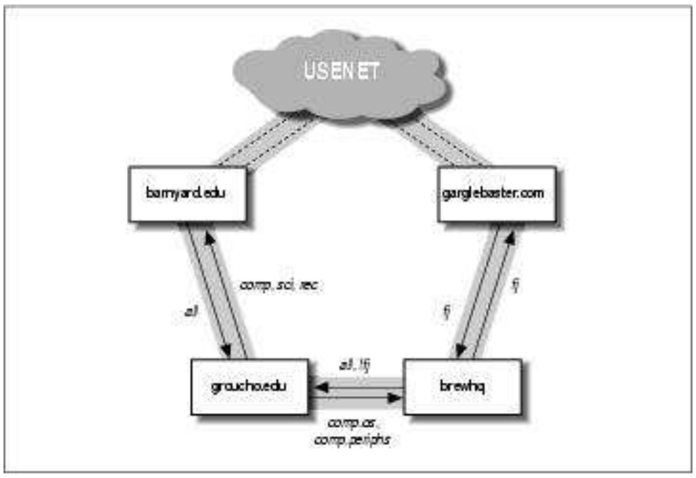
Uno de los hechos más asombrosos sobre Usenet es que no es parte de ninguna organización, ni tiene ninguna clase de dirección centralizada. De hecho, es parte de la erudición de Usenet que, salvo una descripción técnica, no se puede definir *qué* es; a riesgo de parecer tonto, uno podría definir Usenet como la colaboración de servidores separados que intercambian las noticias de Usenet. Para ser un sitio de Usenet, todo lo que se tiene que hacer es encontrar otro servidor y acordar con sus dueños y administradores el intercambio. Proporcionar noticias a otro sitio se llama *feeding* o alimentación, de ello resulta otro axioma común de la filosofía de Usenet, Consigue quien te alcance noticias y ya eres parte.

La unidad básica de noticias de Usenet es el *artículo*. Éste es un mensaje que un usuario escribe y pone en la red. Para hacer posible que los sistemas de noticias lo manejen, están precedidos por información administrativa llamada cabecera del artículo. Es muy similar al formato de las cabeceras de los mensajes de correo descritos en el estándar de correo de Internet el cuál consiste en varias líneas de texto, cada una se inicia con el nombre de un campo terminado en dos puntos seguidos por el valor del campo. [192]

Los artículos son enviados a uno o más *grupos de noticias*. Uno puede considerar a los grupos de noticias como foros para artículos con relación a un asunto común.Todos los grupos de noticias están organizados en una jerarquía, el nombre de cada grupo indica su lugar en la jerarquía. Frecuentemente esto facilita ver acerca de qué trata cada grupo. Por ejemplo, cualquiera puede ver por el nombre del grupo de noticias que comp.os.linux.announce se usa para anuncios que conciernen a un sistema operativo para computadoras llamado GNU/Linux.

Estos artículos son intercambiados entonces entre todos los sitios de Usenet que llevan las noticias para este grupo. Cuando dos sitios acuerdan el intercambio, son libres para intercambiar cualquier grupo de noticias que gusten, e incluso pueden agregar sus propias jerarquías de noticias locales. Por ejemplo, groucho.edu puede tener enlace de noticias a barnyard.edu, el cual es un gran alimentador de noticias, y varios enlaces a servidores menores a los cuáles alimenta. Ahora, el Colegio Barnyard podría recibir todos los grupos de Usenet mientras la GMU solo quiere unas pocas jerarquías mayores como sci, comp, o rec. Algunos servidores más abajo, digamos un servidor UUCP llamado brewhq, querrán aún menos grupos, porque no tienen tantos recursos de red o hardware. Por otro lado, brewhq puede querer recibir grupos de la jerarquía fj que no tiene la GMU. Por consiguiente, mantiene otro eslabón con gargleblaster.com, el cual tiene todos los grupos fj y alimenta con ellos a brewhq. El flujo de noticias se muestra en Figura 20−1.

Figura 20−1. Tráfico de noticias a través de la Universidad Groucho Marx



Las etiquetas en las flechas que se originan en brewhq pueden requerir alguna explicación. Por omisión, este servidor quiere todas noticias generadas localmente para que sean enviadas a groucho.edu. Sin embargo, como groucho.edu no tiene los grupos fj, no existe razón para enviar ningún mensaje de estos grupos. Por consiguiente, la etiqueta de la alimentación de brewhq a la GMU es `all,!fj`, lo que significa que todos los grupos, con excepción de los que están bajo fj se envían.

20.3. ¿Cómo maneja Usenet las noticias?

Actualmente, Usenet ha crecido a enormes dimensiones. Los servidores que llevan todos los grupos usualmente transfieren algo como 60 MB diarios. [193] Por supuesto, esto requiere mucho más que mezclar ficheros. Vamos a dar una mirada a la manera en la mayoría de los sistemas Unix manejan las noticias de Usenet.

Las noticias empiezan cuando los usuarios crean y publican los artículos. Cada usuario introduce un mensaje en una aplicación especial llamada lector de noticias, el cual lo formatea apropiadamente para su transmisión al servidor de noticias local. En entornos Unix el lector de noticias normalmente emplea la instrucción **inews** para transmitir artículos al servidor de noticias usando el protocolo TCP/IP. Sin embargo, también es posible escribir el artículo directamente en un fichero dentro de un directorio especial llamado cola de noticias. Una vez que la publicación se entrega al servidor local de noticias, éste toma la responsabilidad de entregar el artículo a otros usuarios de noticias.

Las noticias son distribuidas a través de la red por varios transportes. El medio acostumbraba a ser UUCP, pero hoy el tráfico principal se lleva por sitios de Internet. El algoritmo de encaminamiento usado se llama *flooding*.[194] Cada sitio mantiene varios enlaces (*news feeds*) a otros servidores. Cualquier artículo generado o recibido por el sistema de noticias local es reenviado a ellos, a menos que ya haya pasado por ellos, en cuyo caso será descartado. Un sitio puede averiguar todos los sitios por los que ha pasado el artículo observando el campo `Path:` de la cabecera. Este campo contiene una lista de todos los sistemas que ha atravesado el artículo, separados por un signo de admiración[195].

Para distinguir los artículos y reconocer los duplicados, los artículos de Usenet llevan un identificador de mensajes, (especificado en el campo `Message-Id:` de la cabecera), el cuál es una combinación del nombre del servidor y un número de serie. `<`*`serial@site`*`>`. Para cada artículo procesado, los sistemas de noticias registran su identificador en un fichero llamado *history* contra el cual se cotejan los artículos recién llegados.

El flujo entre dos servidores cualquiera puede ser limitado por dos criterios. Uno, al artículo se le asigna una distribución (en el campo `Distribution:` de la cabecera), que puede ser usado para confinarlo dentro de un determinado grupo de servidores. Por otro lado, los grupos de noticias intercambiados pueden ser limitados por ambos sistemas, el remitente y el receptor. El conjunto de grupos de noticias y distribuciones que le es permitido transmitir a un servidor se mantienen usualmente en el fichero `sys`.

El número de artículos normalmente requiere que se hagan mejoras al esquema anterior. En redes UUCP los sistemas recogen los artículos en un periodo de tiempo y los combinan en un único fichero el cuál es comprimido y enviado al servidor remoto. Esto se llama *procesado por lotes*[196].

Una técnica alternativa es la del protocolo *ihave/sendme* que previene la transmisión de artículos duplicados en primer lugar, así se ahorra ancho de banda de la red. En lugar de poner todos los artículos en un bloque y enviarlo, sólo se envían al servidor remoto los IDs combinados en un gran mensage llamado ihave. El servidor remoto lee este mensaje, lo compara con su fichero "history" y retorna la lista de artículos que quiere en un mensaje sendme. Sólo los artículos requeridos son enviados.

Claro, el protocolo ihave/sendme sólo tiene sentido si involucra dos grandes servidores que reciben noticias de varias fuentes independientes entre sí, y que intercambian noticias con la frecuencia suficiente como para generar un flujo de noticias eficiente.

Los servidores de Internet generalmente confían en el software basado en TCP/IP que usa el Protocolo de Transferencia de Noticias (NNTP). NNTP se describe en el RFC−977; el cuál es responsable de transferir las noticias entre servidores nuevos y provee acceso a Usenet a usuarios individuales en nodos remotos.

Se conocen tres maneras diferentes de transferir las noticias con NNTP. Una es la versión en tiempo real de ihave/sendme, también conocida como *push[197]* las noticias. La segunda técnica es llamada *pull[198]* las noticias, en la cuál el cliente requiere una lista de artículos de un grupo de noticias o jerarquía determinado que han llegado al servidor después de una fecha especificada y elige aquellas que no encuentra en su fichero "history". La tercera técnica es la lectura interactiva de noticias y le permite a usted o a su lector de noticias recuperar artículos de un grupo especificado, también colocar artículos con la información de cabecera incompleta.

Cada servidor guarda las noticias en una jerarquía de directorios bajo `/var/spool/news`, cada artículo en un fichero separado y cada grupo en un directorio separado. El nombre del directorio se construye a partir del nombre del grupo, cuyos componentes son los componentes de la ruta. De este modo, los artículos de comp.os.linux.misc se guardan en `/var/spool/news/comp/os/linux/misc`. Los artículos de un grupo reciben números de acuerdo a su orden de llegada. Este número sirve como nombre del fichero. El rango de los números de los ficheros vigentes se conserva en un fichero llamado `active` el cual al mismo tiempo sirve como la lista de grupos del sistema.

Toda vez que el espacio en disco es un recurso finito, se tiene que empezar a desechar artículos después de un tiempo. [199] A esto se llama *expiración*. Usualmente los artículos de un determinado grupo y jerarquía expiran al cabo de un número fijo de días después de llegar. El autor puede invalidar esta fecha de expiración especificando una fecha de expiración en el campo `Expires:` de la cabecera del artículo.

Ahora usted tiene bastante información para escoger qué leer después. Los usuarios de UUCP pueden leer sobre C−News en Capítulo 21. Si usted está usando una red TCP/IP, lea acerca de NNTP en Capítulo 22. Si necesita transferir volúmenes moderados de noticias sobre TCP/IP, el servidor descrito en ese capítulo puede ser suficiente. Para instalar un servidor de noticias pesado que pueda manejar grandes volúmenes de material, vaya a leer acerca de Internet News en Capítulo 23.

Capítulo 21. C−News

Uno de los paquetes de software más populares para las NetNews es C−News. Fue diseñado para servidores que llevan noticias sobre enlaces UUCP. Este capítulo discutirá los conceptos centrales de C−News, y las tareas de instalación básica y de mantenimiento.

C−News almacena sus ficheros de configuración en el directorio `/etc/news`, y la mayoría de sus ficheros binarios en `/usr/lib/news/`. Los artículos se guardan en `/var/spool/news`. Debe asegurarse de que todos los ficheros de esos directorios son propiedad del usuario news o el grupo news. La mayoría de los problemas surgen por la inaccesibilidad a los ficheros por parte de C−News. Use la instrucción **su** para trabajar como usuario news antes de tocar nada del directorio. La única excepción es el programa **setnewsids**, que se usa para asignar las ID reales del usuario de algunos programas de noticias. Éste debe ser propiedad del root y tener el bit setuid activado.

En este capítulo, trataremos en detalle todos los ficheros de configuración de C−News y le mostraremos lo que tiene de hacer para mantener su servidor en funcionamiento.

21.1. Enviando noticias

Los artículos pueden ser suministrados a C−News de varias formas. Cuando un usuario envía un artículo, el lector de noticias normalmente lo pasa al programa **inews**, que completa la información de la cabecera. Las noticias que llegan al servidor, ya sea un solo artículo o un lote de ellos, son pasadas por el programa **rnews**, que las guardará en el directorio `/var/spool/news/in.coming`, desde donde más tarde serán recogidas por **newsrun**. Sin embargo, con cualquiera de estas dos técnicas el artículo se pasará finalmente por el programa **newsrun**.

La orden **relaynews** comprueba si un artículo ya se encuentra en el servidor buscando el ID del mensaje en el fichero `history`. Los artículos duplicados son eliminados. Entonces **relaynews** mira la línea de la cabecera `Newsgroups:` para saber si el servidor local solicita artículos de cualquiera de estos grupos. Si es así, y el grupo de noticias aparece en el fichero `active`, **relaynews** intenta almacenar el articulo en su directorio correspondiente en el área de cola de noticias. Si el directorio no existe, es creado. El ID del mensaje que tiene el artículo es registrado en el fichero `history`. Si no, **relaynews** elimina el artículo.

Algunas veces **relaynews** falla al guardar un artículo entrante porque el grupo al que ha sido enviado no está listado en su fichero `active`. En este caso, el artículo se mueve al grupo junk.[200] **relaynews** también busca artículos sin referenciar y los rechaza. Los lotes entrantes que fallan por cualquier razón se mueven a `/var/spool/news/in.coming/bad`, y se registra un mensaje de error.

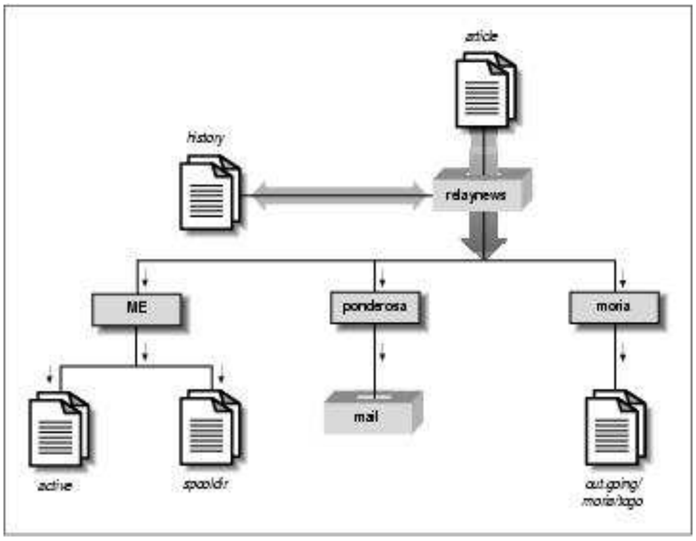
Después de esto, el artículo se transmite a todos los otros servidores que soliciten noticias de estos grupos, usando el transporte especificado para cada servidor . Para asegurarse de que un artículo no es enviado a un servidor que ya lo tiene, cada servidor de destino se compara con el campo `Path:` de la cabecera, que contiene la lista de servidores por los que el artículo ha pasado, escrito al estilo UUCP bang−path source−routing descrito en Capítulo 17. Si el nombre del servidor destinatario no aparece en esta lista, el articulo se le envía.

C−News se usa comúnmente para transmitir noticias entre servidores UUCP, aunque es también posible usarlo bajo un entorno NNTP. Para entregar noticias a un servidor remoto UUCP, tanto un solo artículo como lotes enteros, se usa **uux** para ejecutar el programa **rnews** en un servidor remoto y entregarle el artículo o lote por su entrada estándar. Consulte en Capítulo 16 para más información sobre el UUCP.

Proceso por lotes es un término usado para describir el envío de grandes cantidades de artículos individuales en una sola transmisión. Cuando el procesamiento por lotes se activa en un servidor, C−News no envía ningún artículo entrante inmediatamente; en vez de eso, añade su localización a un fichero, normalmente `out.going/site/togo`. Periódicamente, el programa **cron** ejecuta un programa desde una entrada del **crontab**, que lee este fichero y mete todos los artículos listados en uno o más ficheros, opcionalmente comprimiéndolos y enviándolos a **rnews** en el servidor remoto. [201]

Figura 21−1 muestra las noticias fluyendo a través de **relaynews**. Los artículos deben ser transmitidos al servidor local (indicado por ME), a un servidor llamado ponderosa vía email, y a un servidor llamado moria, para el cual el proceso por lotes está activado.

Figura 21−1. Flujo de noticias mediante relaynews



21.2. Instalación

C−News suele estar disponible empaquetado para cualquier distribución moderna de GNU/Linux, por lo que la instalación será fácil. Si no es así, o quiere instalarlo desde la distribución del código original, por supuesto que también puede.[202] No importa como lo instale, necesitará editar los ficheros de configuración de C−News. Sus formatos serán descritos en la siguiente lista:

`sys`
: El fichero `sys` controla que grupos de noticias recibe y reenvía su grupo de noticias. Hablaremos de esto en detalle en la siguiente sección.

`active`
: No es editado normalmente por la administración; contiene las órdenes para manejar los artículos en cada grupo de noticias que el servidor maneja.

`organization`
: Este fichero debe de contener el nombre de su organización, por ejemplo, Cervecera Virtual, Inc. En su máquina de casa, introduzca servidor privado, o cualquier nombre que desee. La mayoría de la gente no dirá que su servidor está configurado correctamente hasta que no haya configurado este fichero.

`newsgroups`
: Este fichero es una lista de todos los grupos de noticias, con una línea para describir el propósito de cada uno. Estas descripciones son usadas frecuentemente por los lectores de noticias cuando muestran la lista de todos los grupos a los que está suscrito.

`mailname`

El nombre de su servidor de correo, por ejemplo, vbrew.com.

whoami
: El nombre para su servidor de noticias. Muy a menudo, se usa el nombre del servidor de UUCP, por ejemplo, vbrew.

explist
: Probablemente deberá editar este fichero para reflejar sus tiempos de expiración predeterminados para grupos de noticias especiales. El espacio en disco puede jugar un papel importante en sus elecciones.

Para crear una jerarquía inicial de grupos de noticias, obtenga los ficheros `active` y `newsgroups` del servidor que le provee. Instálelos en `/etc/news`, asegurándose de que son propiedad de news y tienen un modo de protección 644, usando la orden **chmod**. Borre todos los grupos to.* del fichero active, y añada to.my−site, to.feed−site, junk, y control. Los grupos to.* se usan normalmente para intercambiar mensajes tipo ihave/sendme, pero deben listarlos tanto si planea usar este tipo de mensajes como si no. Después, sustituya todos los números de los artículos en el segundo y tercer campo de `active` usando las siguientes instrucciones:

```
# cp active active.old
# sed 's/ [0−9]* [0−9]* / 0000000000 00001 /' active.old > active
# rm active.old
```

La segunda orden invoca el editor **sed**. Esta invocación reemplaza dos cadenas de ceros y la cadena `000001`, respectivamente.

Finalmente, cree el directorio de cola de noticias y los subdirectorios usados para las noticias entrantes y salientes:

```
# cd /var/spool
# mkdir news news/in.coming news/out.going news/out.master
# chown −R news.news news
# chmod −R 755 news
```

Si esta usando una versión precompilada del lector de noticias procedente de una distribución diferente a la del servidor C−News que estáa ejecutando, puede encontrase con que alguno de ellos espera la cola de noticias en `/usr/spool/news` en vez de `/var/spool/news`. Si su lector de noticias no parece encontrar ningún artículo, cree un enlace simbólico de `/usr/spool/news` a `/var/spool/news` como éeste:

```
# ln −sf /usr/spool/news /var/spool/news
```

Ahora estáa preparado para recibir noticias. Recuerde que no tiene que crear directorios de cola para cada grupo individual de noticias. Ccada vez que C−News recibe un artículo de un grupo para el que todavía no hay directorio de cola, lo crea.

En particular, éesto le ocurre a *todos* los grupos a los que se ha enviado un artículo. Así que, después de un cierto tiempo, encontrará su cola de noticias llena con directorios para grupos de noticias a los que Ud. nunca se ha subscrito, como alt.lang.teco. Puede evitar esto tanto borrando los grupos no deseados del `active`, como ejecutando regularmente un guión que borre todos los directorios vacíos de `/var/spool/news` (excepto `out.going` y `in.coming`, por supuesto).

C−News necesita un usuario a quien mandar los mensajes de error y los informes de estado. Por omisión, éste es usenet. Si usa el valor por omisión, tendrá que establecer un alias para que reenvíe todos los correos a una o más personas responsables. Puede también evitar ésto estableciendo la variable de entorno NEWSMASTER al nombre apropiado. Tiene que hacer ésto en el fichero `crontab` de news, así como cada vez que invoque manualmente una herramienta administrativa, por lo que instalar un alias es probablemente más fácil. Los alias para los mails se describen en Capítulo 18 y Capítulo 19.

Mientras esté hackeando `/etc/passwd`, asegúrese que cada usuario tiene su nombre real en el campo pw_gecos del fichero de contraseñas (éste es el cuarto campo). Es una cuestión de etiqueta en Usenet que el nombre real del remitente aparezca en el campo `From:` del artículo. Por supuesto, de cualquier forma querrá hacerlo cuando use el correo.

21.3. El fichero sys

El fichero `sys`, situado en `/etc/news`, controla que jerarquías recibe y reenvía a otros servidores. Aunque hay herramientas de mantenimiento llamadas **addfeed** y **delfeed**, pensamos que es mejor mantener este fichero a mano.

El fichero `sys` contiene entradas para cada servidor con el que intercambia noticias, además tiene una lista de los grupos que usted acepta. La primera línea es una entrada ME que describe tu sistema. Es una apuesta segura usar lo siguiente:

```
    ME:all/all::
```

Además, tendrá que añadir una línea por cada servidor al que envía noticias. Cada línea se parece a esta:

```
sitio[/exclusiones]:listagrupo[/listdistribución][:opciones[:órdenes]]
```

Las entradas pueden continuar a lo largo de varias líneas usando la barra invertida (\). Un símbolo (#) denota un comentario.

`sitio`
: Éste es el nombre del servidor al que se aplica la entrada. Una de las elecciones más usuales, es el nombre del servidor UUCP. También debe existir una entrada con su servidor en el fichero `sys` , de lo contrario no recibirá ninguno de los artículos.

  El nombre especial de servidor ME indica su servidor. La entrada ME define todos los grupos de noticias que Ud. está preparado para almacenar localmente. Los artículos que no concuerden con la línea ME irán al grupo junk.

  C–News rechaza cualquier artículo que ya haya pasado a través de su servidor para evitar que los artículos regresen. C–News realiza ésto asegurándose de que el nombre del servidor local no aparece en el `Path:` del artículo. Algunos servidores usan su nombre de dominio completamente calificado en este campo, o un alias como `news.`*`sitio.dominio`*. Para prevenir que cualquier artículo regrese a estos servidores, es importante añadir todos los alias a la lista de exclusión, separados por comas.

  Para la entrada aplicada al servidor moria, por ejemplo, el campo *`site`* debe contener moria/moria.orcnet.org. Ahora, si moria fuese un alias del servidor news.orcnet.org, el campo *`site`* debe contener la siguiente expresión moria/moria.orcnet.org,news.orcnet.org.

`listagrupo`
: Ésta es una lista de suscripción, separada por comas, de grupos y jerarquías para ese servidor en particular. Una jerarquía debe especificarse dando el prefijo de la jerarquía (como comp.os para todos los grupos cuyos nombres empiezan con este prefijo), seguido opcionalmente por la palabra clave all (por ejemplo, comp.os.all).

  Para excluir una jerarquía o grupo de reemisión, debe ser precedido con un símbolo de exclamación. Si un grupo de noticias encaja con más de una definición de la lista, la coincidencia más larga se aplica.} Por ejemplo, si *`listagrupo`* contiene:

  ```
      !comp,comp.os.linux,comp.folklore.computers
  ```

  ningún grupo de la jerarquía comp excepto comp.folklore.computers y todos los grupos bajo comp.os.linux serán suministrados a ese servidor.

Si el servidor requiere que se le envíen todas las noticias que usted recibe, introduzca all como *listagrupo*.

listadistribución
: Este valor está separado de *grouplist* por una barra invertida, y contiene una lista de distribuciones para ser reenviada. Usted puede, nuevamente, excluir ciertas distribuciones precediéndolas con un símbolo de exclamación. Todas las distribuciones se denotan con all. Omitir *listadistrribución* implica una lista de all.

Por ejemplo, puede usar una lista de distribución de all,!local para impedir que las noticias de uso sólo local sean enviadas a servidores remotos.

Usualmente existen al menos dos distribuciones: world, que es a menudo la distribución predeterminada usada cuando el usuario no especifica nada, y local. Puede haber otras distribuciones que se empleen para una cierta región, estado, país, etc. Finalmente hay dos distribuciones usadas sólamente por C−News; éstas son sendme y ihave, y son usadas para el protocolo sendme/ihave.

El uso de distribuciones es materia de debate. El campo de distribución en un artículo de noticias puede ser creado de forma arbitraria, pero para que la distribución sea efectiva, los servidores de noticias en la red deben conocer ésto. Para unos, algunos lectores de noticias crean falsas distribuciones simplemente usando la jerarquía de alto nivel, por ejemplo comp cuando se envía un mensaje a comp.os.linux.networking. Las distribuciones que se emplean en regiones son a menudo también cuestionables, porque las noticias deben viajar fuera de su región cuando son enviadas a través de Internet. [203] Sin embargo, las distribuciones empleadas por una organización, son muy significativas, por ejemplo para evitar la salida de información confidencial de la red de la compañía. No obstante, este propósito generalmente se consigue mejor creando un grupo de noticias o una jerarquía separados.

opciones
: Este campo describe ciertos parámetros para el suministro. Puede estar vacío, o ser una combinación de los siguientes:

F
: Esta opción permite el proceso por lotes.

f
: Ésta es casi idéntica a la opción `F`, permite a C−News calcular el tamaño de los lotes salientes con más precisión, y debe ser usada preferentemente antes que F.

I
: Ésta opción hace que C−News produzca una lista de artículo apta para ser usada por el protocolo ihave/sendme. Hay que hacer modificaciones al fichero `sys` y a `batchparms` para habilitar ihave/sendme.

n
: Ésta opción crea ficheros por lotes para clientes de transferencia NNTP activa como **nntpxmit** (ver el Capítulo 22). Los procesos por lotes contienen el nombre del fichero del artículo junto con el identificador de mensaje.

L
: Indica a C−News que sólo transmita los mensajes generados en su servidor. Éste flag puede ir seguido por un número decimal *n*, el cual le indica a C−News que sólo transfiera los artículos generados a *n* saltos desde su servidor. C−News determina el número de saltos a partir del campo *Path:*.

u
: Éste flag hace que C−News realice el proceso por lotes sólo de los artículos que se encuentran en los grupos sin moderar.

m
: Inverso al anterior, C−News procesa los artículos de los grupos moderados.

Se debe utilizar al menos uno de los parámetros `F`, `f`, `I`, o `n`.

cmds
: Éste campo contiene una orden a ser ejecutada por cada artículo, a menos que el proceso por lotes esté habilitado. El artículo será suministrado a través de su entrada estándar. Esto sólo debería usarse para suministros muy pequeños; de otra manera, la carga en ambos servidores sería demasiado alta.

La instrucción predeterminada es:

```
uux − −r −z sistema−remoto!rnews
```

Esto invoca a **rnews** en el sistema remoto, y le suministra el artículo en su entrada estándar.

La ruta de búsqueda predeterminada para las órdenes indicadas en este campo es `/bin:/usr/bin:/usr/lib/news/batch`. El último directorio contiene un cierto número de guiones del intérprete de órdenes cuyos nombres empiezan por **via** estos serán brevemente descritos luego, en este mismo capítulo.

Si el proceso por lotes está habilitado utilizando algún flag como `F`, `f`, `I`, o `n`, C−News espera encontrar un nombre de fichero en este campo en vez de una instrucción. Si el nombre del fichero no empieza con una barra inclinada (/), se asume que es relativo a `/var/spool/news/out.going`. Si el campo esta vacío, su valor predeterminado es `sistema−remoto/togo`. Se espera que este fichero tenga el mismo formato que el fichero `sistema−remoto/togo` y contenga una lista de los artículos a transmitir.

Cuando configure C−News, probablemente tendrá que escribir su propio fichero `sys`. Aquí se muestra un ejemplo para vbrew.com, del cual puede copiar lo que necesite:

```
# Tomamos todo lo que se nos suministra.
ME:all/all::
# Enviamos todo lo que recibimos a moria, excepto los artículos locales
# y relacionados con la cervecera. Se utiliza proceso por lotes.
moria/moria.orcnet.org:all,!to,to.moria/all,!local,!brewery:f:
# Enviamos comp.risks a jack@ponderosa.uucp
ponderosa:comp.risks/all::rmail jack@ponderosa.uucp
# swim obtiene un suministro reducido.
swim/swim.twobirds.com:comp.os.linux,rec.humor.oracle/all,!local:f:
# Guardar los artículos de mapas de correo para procesarlos luego.
usenet−maps:comp.mail.maps/all:F:/var/spool/uumaps/work/batch
```

21.4. El Fichero active

El fichero `active` está situado en `/etc/`, y lista todos los grupos conocidos en su servidor, y los artículos disponibles actualmente. Rara vez tendrá que tocarlo, sin embargo, lo explicamos por amor a la perfección. Las entradas toman el siguiente formato:

```
grupo_noticias high low perm
```

`grupo_noticias` es el nombre del grupo. *`low`* y *`high`* contienen los números más bajo y más alto de los artículos actualmente disponibles. Si no hay ningún artículo en ese momento, *`low`* es igual a *`high`*+1.

Al menos, eso e lo que el campo `low` pretende hacer. Sin embargo, por razones de eficiencia, C−News no actualiza este campo. Esto no sería una gran pérdida si no hubiera algunos lectores que dependen de él. Por ejemplo, **trn** comprueba este campo para ver si es posible purgar cualquier artículo de su base de datos de hilos. Para actualizar el campo bajo `low`, tiene por lo tanto que ejecutar regularmente la orden **updatemin** (o, en versiones más antiguas de C−News, es guión **upact**).

`perm` es un parámetro que detalla el tipo de acceso que los usuarios tienen concedido en el grupo. Toma uno de los siguientes valores:

y
: Se permite a los usuarios publicar en este grupo.

n
: Los usuarios no pueden publicar en este grupo. Sin embargo, todavía puede ser leído.

x
: Éste grupo ha sido deshabilitado localmente. Esto ocurre algunas veces cuando los administradores de noticias (o sus superiores) se ofenden por algún artículo publicado en ciertos grupos.

  Los artículos recibidos para estos grupos no son almacenados localmente aunque son reenviados a los servidores que los piden.

m
: Esto denota un grupo moderado. Cuando un usuario intenta enviar un artículo a este grupo, un lector de noticias inteligente le notificará al usuario la respectiva condición del grupo, y enviará el artículo al moderador del grupo. La dirección del moderador se toma del fichero `moderators` en el directorio `/var/lib/news`.

`=real-group`
: Esto marca a *`newsgroup`* como un alias local para otro grupo, a saber *`real-group`*. Todos los artículos publicados en *`newsgroup`* serán redirigidos a él.

En C−News, generalmente no tendrá que acceder directamente a este fichero. Los grupos deben ser añadidos o borrados localmente usando **addgroup** y **delgroup** (vea la sección Sección 21.10 luego, en este capítulo). Un mensaje de control newgroup crea un grupo para todo Usenet, mientras que un mensaje rmgroup elimina un grupo. *¡Nunca envíe Ud. un mensaje de este tipo!* Para saber como crear un grupo de noticias, lea los mensajes enviados mensualmente a news.announce.newusers.

El fichero `active.times` está estrechamente relacionado con el fichero `active`. Cada vez que se crea un grupo, C−News registra un mensaje en este fichero, conteniendo el nombre del grupo creado, la fecha de creación, si fue creado por un mensaje de control newgroup localmente, y quién lo hizo. Esto es para facilitar la vida a los lectores de noticias, quienes pueden notificar al usuario los grupos recién creados. También lo usa la orden **NEWGROUPS** de NNTP.

21.5. Procesar Artículos por Lotes

Los lotes de noticias siguen un formato particular, el cual es el mismo para Bnews, C−News e INN. Cada artículo está precedido por una línea como esta:

```
#! rnews count
```

donde *`count`* es el número de bytes en el artículo. Cuando se usa la compresión de lotes, el fichero resultante es comprimido como un todo, y precedido por otra línea, que indica el mensaje a ser usado por la compresión. la herramienta de compresión estándar es compress, la cual se indica con:

```
    #! cunbatch
```

Algunas veces, cuando hay que enviar los lotes usando un software de correo que elimina el octavo bit de todos los datos, se puede proteger un lote usando un método de codificación llamado *codificación c7*(c7−encoding, en inglés); estos lotes serán marcados por **c7unbatch**.

Cuando se le suministra un lote **rnews** en el servidor remoto, éste comprueba esas marcas y procesa el lote apropiadamente. Algunos servidores también usan otras herramientas de compresión tales como **gzip**, y en su lugar, preceden a sus ficheros comprimidos con **zunbatch**. C−News no reconoce cabeceras no estándares como esas; Ud. deberá modificar el código fuente para darle soporte.

En C−News, el proceso por lotes de ficheros lo realiza `/usr/lib/news/batch/sendbatches`, el cual recoge la lista de artículos del fichero `site/togo` y los pone en varios lotes de noticias. Debería ejecutarse una vez cada hora, o incluso más a menudo, dependiendo del volumen de tráfico. Esta operación es controlada por el fichero `batchparms` situado en `/var/lib/news`. Este fichero describe el máximo tamaño de lote permitido para cada servidor, el tipo de proceso por lotes y opcionalmente el programa de compresión usado, además del método de transporte para entregarlo al servidor remoto. Puede especificar los parámetros del proceso por lotes para cada servidor, además de un conjunto de parámetros predeterminados para servidores no mencionados explícitamente.

Cuando instale C−News, seguramente hallará un fichero de nombre `batchparms` en su distribución que contenga una entrada predeterminada, con valores razonables, así que es muy probable que no tenga que tocar el fichero. No obstante, describimos su formato por si acaso. Cada línea consta de seis campos, separados por espacios o tabuladores:

```
sitio tamaño max batcher muncher transporte
```

sitio
: *sitio* es el nombre del servidor al que se aplica la entrada. El fichero `togo` para este servidor debe residir en `out.going/togo` bajo la cola de noticias. Un servidor llamado /default/ denota la entrada predeterminada y coincide con cualquier servidor que no sea especificado con una única entrada.

tamaño
: *tamaño* es el tamaño máximo de los lotes creados (antes de la compresión). Para aquellos artículos que son mayores que este valor, C−News hace una excepción y los pone en un lote a ellos solos.

max
: *max* es el máximo número de lotes creados y programados para la transferencia antes de que el proceso por lotes se interrumpa para este servidor en particular. Esto es útil en el caso de que el servidor remoto no esté disponible durante un largo período de tiempo, porque previene que C−News ateste sus directorios de cola UUCP con millones de lotes de noticias.

  C−News determina el número de lotes que hay en cola usando el guión **queuelen** situado en `/usr/lib/news/`. Si ha instalado C−News desde paquetes, el guión no necesita retoques, pero si elige lugares diferentes para los directorios de cola, por ejemplo para Taylor UUCP, deberá crear su propia versión del guión. Si no le importa el número de ficheros de cola (porque Ud. es la única persona usando el ordenador, y no escribe artículos gigantes), puede reemplazar el contenido del guión por una simple sentencia **exit 0**.

batcher
: El campo *batcher* contiene la orden usada para producir un lote a partir de la lista de artículos del fichero `togo`. Para las fuentes habituales, éste es generalmente **batcher**. Puede que se proporcionen otros empaquetadores para otros propósitos. Por ejemplo, el protocolo ihave/sendme requiere que la lista de artículos sea convertida en mensajes de control *ihave* o *sendme*, los cuales se envían al grupo to.site. Esto lo realizan las instrucciones **batchih** y **batchsm** respectivamente.

muncher
: El campo `compresión` especifica la orden a usar para la compresión de los lotes. Generalmente, se usa compcun, un guión que produce lotes comprimidos[204]. Alternativamente, puede proporcionar un muncher que use **gzip**, para comprimir, digamos **gzipcun** (para ser claros: tiene que escribirlo usted mismo). Debe asegurarse de que **uncompress** en el servidor remoto esté parcheado para reconocer ficheros comprimidos con **gzip**.

  Si el servidor remoto no tiene una orden **uncompress**, debe especificar **nocomp**, lo que implica que no se realice compresión alguna.

transporte
: El último campo, *transporte*, describe el transporte a ser utilizado. Hay disponibles varias órdenes estándar para diferentes transportes cuyos nombres empiezan con **via**. **sendbatches** les pasa el nombre del servidor de destino en la línea de órdenes. Si la entrada `batchparms` no era /default/, **sendbatches** deriva el nombre del servidor del campo *sitio* suprimiendo cualquier cosa después e incluyendo el primer punto o barra inclinada. Si la entrada batchparms es /default/, se usa el nombre del directorio introducido en `out.going`.

Para llevar a cabo el proceso por lotes para un servidor específico, se invoca como:

```
# su news −c "/usr/lib/news/batch/sendbatches site"
```

Cuando se invoca sin argumentos, **sendbatches** maneja todas las colas de lotes. La interpretación de todas depende de una entrada predeterminada en `batchparms`. Si se encuentra una, se comprueban todos los directorios de `/var/spool/news/out.going`; si no, **sendbatches** recorre todas las entradas de `batchparms`, procesando solamente los lugares encontrados. Note que **sendbatches**, cuando explora el directorio `out.going` toma sólo aquellos directorios que no contienen ningún punto o símbolo arroba (@)como nombre de servidor.

Hay dos órdenes que usan **uux** para ejecutar **rnews** en el servidor remoto: **viauux** y **viauuxz**. El último establece el parámetro `z` para **uux**, el cual, previene que versiones más antiguas de uux devuelvan mensajes de éxito por cada artículo entregado. Otra orden, **viamail**, envía los lotes de artículos al usuario rnews en el servidor remoto vía correo. Por supuesto, esto requiere que el sistema remoto administre de alguna manera todo el correo para rnews a su sistema local de noticias. Para obtener una lista completa de estos transportes, refiérase a las páginas del manual de `newsbatch`.

Todas las órdenes de los tres últimos campos deben estar situados, bien en `out.going/site` o en `/usr/lib/news/batch`. La mayoría de ellos son guiones, de tal forma que pueda confeccionar fácilmente nuevas herramientas para sus necesidades personales y son invocadas a través de tuberías (pipes). La lista de artículos se suministra al batches por su entrada estándar, la cual produce el lote en su salida estándar. Esto a su vez, se vuelve a entubar en el muncher, y así sucesivamente.

Aquí hay un ejemplo:

```
# fichero batchparms para la cervecera
# site         | size    |max    |batcher   |muncher    |transport
#-------------+--------+-------+---------+-----------+-----------
/default/       100000  22      batcher   compcun     viauux
swim             10000  10      batcher   nocomp      viauux
```

21.6. Caducando Noticias

En B News, la caducidad de las noticias solía realizarse mediante el programa **expire**, que tomaba como argumento una lista de los grupos de noticias, junto con una especificación del tiempo después del cual los artículos caducaban. Para hacer que diferentes jerarquías caducasen en momentos distintos, tenía que escribir un guión que invocase a **expire** por cada uno de ellos en forma individual. C−News ofrece una solución más conveniente a esto. En un fichero llamado `explist`, puede especificar los grupos de noticias y los respectivos intervalos. Una orden llamada **doexpire** se activa diariamente desde **cron** y procesa todos los grupos acorde a esta lista.

Ocasionalmente, Ud. puede querer mantener artículos de ciertos grupos incluso después de que hayan caducado; por ejemplo, podría querer mantener los programas enviados a comp.sources.unix. A esto se le llama *archivado*. `explist` le permite marcar grupos para el archivado.

Una entrada en `explist` tiene el siguiente formato:

```
grouplist perm times archive
```

`grouplist` es una lista separada por comas de los grupos de noticias a los que aplica la entrada. Se pueden especificar jerarquías completas indicando el prefijo del nombre del grupo, añadiendo, opcionalmente all. Por ejemplo, para indicar una entrada que se aplique a todos los grupos de comp.os, o tambien **`comp.os`** o **`comp.os.all`**.

Cuando se van a *caducar* las noticias de un grupo, se constata el nombre del grupo con todas las entradas de `explist` en el orden dado. La primer entrada que coincida es la que se aplica. Por ejemplo, para eliminar la mayoría de comp después de cuatro días, excepto comp.os.linux.announce, que desea mantener por una semana, debe simplemente tener una entrada para esto, que especifique un período de caducidad de siete días, seguida por una para comp, que especifique cuatro días.

El campo *`perm`* detalla si la entrada se aplica a grupos moderados, no moderados, o a cualquier grupo. Debe tomar uno de los valores m, u, o x, lo que designa la condición de moderado, no moderado o cualquier tipo.

El tercer campo, *`times`*, usualmente contiene un solo número. Éste es el número de días después de los cuales caducarán los artículos si no se les ha asignado una fecha de caducidad artificial en el campo `Expires:` de la cabecera del artículo. Debe darse cuenta de que éste es el número de días contando desde la *llegada* a su servidor, no desde la fecha de publicación.

Sin embargo, el campo *`times`* puede ser más complejo que eso. Puede ser una combinación de hasta tres números separados unos de otros por un guión (−). El primero designa el número de días que tienen que pasar antes de que el artículo sea considerado candidato para estar caduco, incluso si el campo `Expires:` ya haya indicado esta condición. Rara vez es útil usar otro valor que no sea cero. El segundo campo, es el valor mencionado arriba, es decir, el número predeterminado de días después de los cuales caducará. El tercero es el número de días después de los cuales un artículo caducará incondicionalmente, sin tomar en cuenta si tiene un campo `Expires:` o no. Si sólo se indica el número de en medio, los otros dos toman valores por omisión. Éstos pueden especificarse usando la entrada especial /bounds/, la cual se describe un poco más abajo.

El cuarto campo, *`archive`*, designa si el grupo de noticias tiene que archivarse, y dónde. Si no desea archivarlo, debería usar un guión. De lo contrario, use la ruta completa (apuntando a un directorio), o use una arroba (`@`). La arroba designa el directorio de fichero por omisión cuyo valor debe darse a **doexpire** usando el parámetro `a` en la línea de órdenes. Éste directorio debe ser propiedad del usuario news. Cuando **doexpire** archiva un artículo de, digamos, comp.sources.unix, lo almacena en el directorio comp/sources/unix bajo el directorio de fichero, creándolo si no existe. Sin embargo, no se creará el propio directorio de fichero.

Hay dos entradas especiales en el fichero `explist` de las que depende **doexpire**. En vez de una lista de grupos de noticias, tienen las palabras clave /bounds/ y /expired/. La entrada /bounds/ contiene los valores predeterminados usados por el campo *`times`* descrito anteriormente.

El campo /expired/ determina cuánto tiempo guardará C−News las entradas del fichero `history`. C−News no borrará una línea del fichero de historial una vez que el (los) artículo(s) hayan caducado, pero lo guardará por si acaso llega un duplicado tras esa fecha. De lo contrario, un par de semanas es un valor aconsejable para las redes UUCP, dependiendo de los retrasos que experimente con los artículos de esos servidores.

A continuación se reproduce un fichero `explist` de ejemplo con unos intervalos de expiración bastante ajustados:

```
    # Mantiene las líneas de historial dos semanas.
    # Ningún artículo consigue más de tres meses.
    /expired/                       x       14      -
    /bounds/                        x       0-1-90  -
    # grupos que queremos mantener más tiempo que el resto.
    comp.os.linux.announce          m       10      -
    comp.os.linux                   x       5       -
    alt.folklore.computers          u       10      -
    rec.humor.oracle                m       10      -
    soc.feminism                    m       10      -
    # Archiva los grupos *.sources
    comp.sources,alt.sources        x       5       @
    # Valores predeterminados para los grupos de tecnología.
    comp,sci                        x       7       -
    # Suficiente para un fin de semana largo.
    misc,talk                       x       4       -
    # desecha rápidamente lo inservible
    junk                            x       1       -
    # los mensajes de control, también son de escaso interés
    control                         x       1       -
    # para el resto de ellos, la entrada comodín
    all                             x       2       -
```

Hay un cierto número de problemas potenciales con la caducidad en C−News. Uno es que su lector de noticias puede depender del tercer campo del fichero *active* descrito anteriormente, el cuál contiene el número de artículo más bajo disponible. Cuando C−News caduca artículos, no actualiza este campo. Si necesita (o quiere) que este campo represente la situación real, necesita ejecutar un programa llamado **updatemin** después de cada ejecución de **doexpire**. (En versiones anteriores de C−News, existe un guión llamado **upact** que realiza este trabajo.)

C−News no caduca los artículos examinando el directorio de los grupos de noticias, sino que simplemente comprueba en el fichero `history` si el artículo debe caducar.[205] Si el fichero history consigue de alguna manera estar fuera de sincronismo, sus artículos pueden permanecer en su disco duro para siempre, porque C−News los ha olvidado literalmente.[206] Puede reparar esto usando el guión **addmissing** que se encuentra en `/usr/lib/news/maint`, el cuál añadirá los artículos perdidos al fichero `history` o a **mkhistory**, el cuál reconstruye el fichero desde cero. No olvide ser news antes de invocarlo, o de lo contrario terminará con un fichero `history` imposible de leer por C−News.

21.7. Ficheros Diversos

Hay algunos ficheros que controlan el uso de C−News, pero que no son esenciales para su funcionamiento. Todos ellos residen en `/etc/news`. Los describiremos brevemente:

`newsgroups`
: Éste acompaña al fichero `active` y contiene una lista de nombres de grupos de noticias, junto con una descripción, en una sola línea, de su tema principal. Se actualiza automáticamente cuando C−News recibe un mensaje de control checknews.

`localgroups`
: Si tiene muchos grupos locales, puede hacer que C−News no se queje cada vez que recibe un mensaje `checkgroups`, poniendo sus nombres y descripciones en este fichero, justo como aparecerían en el fichero `newsgroups`.

`mailpaths`
: Este fichero contiene la dirección del moderador para cada grupo moderado. Cada línea contiene el

nombre del grupo, seguido por la dirección de correo electrónico del moderador (separado por un tabulador).

Se proporcionan dos entradas especiales por omisión: backbone e internet. Ambas proporcionan − en notacion UUCP de signos de admiración − el camino al servidor principal más cercano y el servidor que reconoce direcciones del estilo RFC−822 (user@host). Las entradas predeterminadas son:

```
    internet            backbone
```

No tendrá que cambiar la entrada internet si tiene instalado **exim** o **sendmail** ya que entienden el direccionamiento RFC−822.

La entrada backbone se usa cada vez que un usuario publica un mensaje en un grupo moderado cuyo moderador no esté listado explícitamente. Si el nombre del grupo de noticias es alt.sewer, y la entrada backbone contiene path!%s, C−News enviará por correo el artículo a path!alt−sewer, esperando que la máquina backbone pueda reenviar el artículo. Para averiguar que camino usar, pregunte al administrador de su proveedor de noticias. Como último recurso, puede usar también uunet.uu.net!%s.

distributions
: Este no es un fichero real de C−News, pero se usa por algunos lectores de noticias y por **nntpd**. Contiene la lista de distribuciones reconocida y una descripción de su efecto (deseado). Por ejemplo, la Cervecera Virtual tiene el siguiente fichero:

```
    world         cualquier lugar del mundo
    local         sólo local a este servidor
    nl            sólo Holanda
    mugnet        sólo MUGNET
    fr            sólo Francia
    de            sólo Alemania
    brewery       sólo la Cervecera Virtual
```

log
: Este fichero contiene un registro de todas las actividades de C−News. Se recorta regularmente ejecutando **newsdaily**; las copias de ficheros de los registros antiguos se guardan bajo `log.o`, `log.oo`, etc.

errlog
: Este es un registro de todos los mensajes de error generados por C−News. Estos no incluyen artículos desechados debido a grupos incorrectos u otros errores de los usuarios. De no estar vacío al momento de ejecutar **newsdaily**, será enviado por correo al administrador de noticias (usenet por omisión) automáticamente.

newsdaily se encarga de limpiar `errlog`. Las copias antiguas se guardan bajo `errlog.o` y compañía.

batchlog
: Este fichero registra todas las ejecuciones de **sendbatches**. Normalmente no tiene interés su contenido. También es atendido por **newsdaily**.

watchtime
: Este es un fichero vacío que crea **newswatch** cada vez que se ejecuta.

21.8. Mensajes de Control

El protocolo de noticias Usenet reconoce artículos de una categoría especial, los cuales provocan ciertas respuestas o acciones del sistema. Estos son los llamados mensajes de *control*. Se reconocen por la presencia de un campo `Control:` en la cabecera del artículo, el cual contiene el nombre de la operación de control a realizar. Existen varios tipos, y todas ellas son manejadas por guiones del intérprete de órdenes situados en `/usr/lib/news/ctl`.

La mayoría de éstos realizarán su acción automáticamente en el momento en que C−News procese el artículo, sin notificar al administrador de noticias. Por omisión, sólo los mensajes checkgroups serán entregados al administrador de noticias, pero puede cambiar esto editando los guiones.

21.8.1. El mensaje cancel

El mensaje más conocido es cancel, con el cual un usuario puede cancelar un artículo enviado por él en otro momento. Esto borra el artículo de los directorios de cola, si existe. El mensaje cancel se reenvía a todos los servidores que reciben noticias de los grupos afectados, sin reparar si el artículo ha sido visto o no. Esto es para tener en cuenta la posibilidad de que el artículo original se haya retrasado sobre el mensaje de cancelación. Algunos sistemas de noticias permiten a los usuarios cancelar los mensajes de otras personas. Por supuesto esto es algo que no se debería hacer.

21.8.2. newgroup y rmgroup

Dos mensajes que se ocupan de la creación y borrado de grupos de noticias son los mensajes newgroup y rmgroup. Los grupos de noticias bajo la las jerarquías usuales solo pueden ser creados después de que haya mantenido una discusión y voto entre los lectores de Usenet. Las reglas aplicadas a la jerarquía alt permiten algo similar a la anarquía. Para más información, ver los mensajes regulares publicados en news.announce.newusers y en news.announce.newgroups. Nunca envíe un mensaje newgroup o rmgroup usted mismo a menos que sepa con seguridad que tiene permiso para hacerlo.

21.8.3. El Mensaje checkgroups

Los mensajes checkgroups son enviados por los administradores de noticias para hacer que todos los servidores de una red sincronicen sus ficheros `active` con la realidad de Usenet. Por ejemplo, los proveedores de servicio de Internet deberían mandar tal mensaje a los servidores de sus clientes. Una vez al mes, el moderador del grupo comp.announce.newgroups envía el mensaje oficial checkgroups para las principales jerarquías. Sin embargo, se envía como un artículo ordinario, no como un mensaje de control. Para realizar la operación checkgroups , salve este artículo en un fichero, digamos `/tmp/check`, borre todo hasta el principio del mismo mensaje de control, y envíelo al guión checkgroups usando la orden siguiente:

```
# su news −c "/usr/lib/news/ctl/checkgroups" < /tmp/check
```

Esto actualizará su fichero `newsgroups`, añadiendo los grupos listados en `localgroups`. El antiguo fichero `newsgroups` será movido a `newsgroups.bac`. Note que rara vez funciona el enviar el mensaje localmente, ya que **inews**, rechaza un artículo tan grande.

Si C−News encuentra desigualdades entre la lista del fichero checkgroups y el fichero `active`, producirá una lista de órdenes que actualizaría su servidor, y lo enviará por correo al administrado de noticias.

Típicamente la salida se parece a esto:

```
From news Sun Jan 30 16:18:11 1994
Date: Sun, 30 Jan 94 16:18 MET
From: news (News Subsystem)
To: usenet
Subject: Problems with your active file
The following newsgroups are not valid and should be removed.
        alt.ascii−art
        bionet.molbio.gene−org
        comp.windows.x.intrisics
```

```
          de.answers
    You can do this by executing the commands:
           /usr/lib/news/maint/delgroup alt.ascii-art
           /usr/lib/news/maint/delgroup bionet.molbio.gene-org
           /usr/lib/news/maint/delgroup comp.windows.x.intrisics
           /usr/lib/news/maint/delgroup de.answers
    The following newsgroups were missing.
          comp.binaries.cbm
          comp.databases.rdb
          comp.os.geos
          comp.os.qnx
          comp.unix.user-friendly
          misc.legal.moderated
          news.newsites
          soc.culture.scientists
          talk.politics.crypto
          talk.politics.tibet
```

Cuando reciba un mensaje como éste de su sistema de noticias, no lo crea ciegamente. Dependiendo de quién envió el mensaje checkgroups, puede que carezca de unos pocos grupos e incluso jerarquías enteras; por lo tanto, debería tener cuidado al borrar cualquier grupo. Si encuentra grupos listados como no presentes que quiera tener en su servidor, tiene que añadirlos usando el guión **addgroup**. Salve la lista de grupos que le faltan en un fichero y páseselo al siguiente guión:

```
#!/bin/sh
#
WHOIAM=`whoami`
if [ "$WHOIAM" != "news" ]
then
    echo "Debe ejecutar $0 como usuario 'news'" >&2
    exit 1
fi
#
cd /usr/lib/news
while read group; do
    if grep -si "^$group[[:space:]].*moderated" newsgroup; then
        mod=m
    else
        mod=y
    fi
    /usr/lib/news/maint/addgroup $group $mod
done
```

21.8.4. sendsys, version, y senduuname

Finalmente, hay tres mensajes que pueden usarse para averiguar la topología de la red. Estos son sendsys, version, y senduuname. Respectivamente, hacen que C−News devuelva al remitente el fichero `sys`, una cadena con la versión del software, y la salida de **uuname**. C−News es muy lacónica con respecto a los mensajes version; ya que devuelve una simple `C`, sin más adornos.

Nuevamente, *nunca* debería distribuir tales mensajes, a menos que esté seguro de que no pueden salir de su red (regional). Las respuestas a los mensajes sendsys pueden hacer caer rápidamente a una red UUCP. [207]

21.9. C−News en un Entorno NFS

Una manera simple de distribuir noticias en una red local, es guardándolas en un nodo central, y exportar los directorios relevantes vía NFS, de manera que los lectores de noticias puedan examinar los artículos directamente. La ventaja de este método sobre NNTP es que la sobrecarga implicada en recuperar y enhebrar artículos es significativamente más baja. Por otra parte, NNTP gana en una red heterogénea donde el equipamiento varía mucho entre nodos, o donde los usuarios no tienen cuentas equivalentes en la máquina servidora.

Cuando se usa NFS, los artículos enviados al nodo local tienen que ser reenviados a la máquina central, porque de otro modo el acceso a los ficheros administrativos expondría al sistema a condiciones de carrera dejando a los ficheros inconsistentes. También podría querer proteger su área de cola de noticias exportándola con atributos de sólo lectura, lo cual requiere también el reenvío a la máquina central.

C−News maneja esto de forma transparente. Cuando envía un artículo, su lector de noticias normalmente invocará a **inews** para inyectar el artículo al sistema de noticias. Esta orden ejecuta algunas comprobaciones sobre el artículo, completa la cabecera y comprueba el fichero `server` en `/etc/news`. Si el fichero existe y contiene un nombre de nodo diferente al del sistema local, se invoca a **inews** en ese servidor remoto vía **rsh**. Puesto que el guión **inews** usa órdenes binarias y ficheros de apoyo de C−News, debe tener C−News instalado localmente, o montar el software de noticias desde el servidor.

Para que la invocación de **rsh** funcione correctamente, cada usuario debe tener una cuenta equivalente en el sistema del servidor, esto es, una a la que pueda acceder sin necesidad de contraseñas.

Asegúrese de que el nombre del sistema indicado en `server` coincida literalmente con la salida de la orden **hostname** en el servidor, si no C−News entrará en un bucle infinito cuando intente entregar el artículo. NFS se discute en detalle en el Capítulo 14.

21.10. Herramientas y Tareas de Mantenimiento

A pesar de la complejidad de C−News, la vida de un administrador de noticias puede ser bastante fácil, porque C−News proporciona una amplia variedad de herramientas de mantenimiento. Es deseable que algunas de estas sean ejecutadas regularmente desde **cron**, como **newsdaily**. El uso de estos guiones reduce drásticamente los requisitos diarios de cuidado y administración de su instalación de C−News.

A menos que se indique lo contrario, estas órdenes están situadas en `/usr/lib/news/maint`. (Note que usted debe ser el usuario news antes de invocarlas. Ejecutándolas como super−usuario puede volver a estos ficheros inaccesibles a C−News.):

newsdaily
: Es un guión importante que le ayuda a mantener los ficheros de registro pequeños, conservando copias de todos ellos de las últimas tres ejecuciones. También intenta detectar cualquier anomalía, como lotes atascados en los directorios de entrada y salida, envíos a grupos de noticias moderados o desconocidos, etc. Los mensajes de error resultantes serán enviados por correo al administrador de noticias.

newswatch
: Se trata de un guión que debería ejecutarse regularmente para buscar anomalías en el sistema de noticias, una vez cada hora más o menos. Está destinado a detectar problemas que tendrán efectos inmediatos en la operatividad de su sistema de noticias y enviar un informe de problemas al administrador de noticias. Las cosas comprobadas incluyen ficheros de bloqueo pasados que no fueron borrados, lotes de entrada desatendidos y la falta de espacio en disco.

addgroup
: Añade un grupo localmente a su servidor. La forma de invocar al guión de forma correcta es:

```
addgroup groupname y|n|m|=realgroup
```

El segundo argumento tiene el mismo significado que el modificador del fichero `active`, significando que cualquiera puede enviar un artículo al grupo (y), que nadie puede enviar (n), que es moderado (m), o que es un alias para otro grupo (=*realgroup*). Usted podría querer usar **addgroup** cuando los primeros artículos de un grupo recién creado lleguen antes que el mensaje de control newgroup destinado a crearlo.

delgroup

Le permite borrar localmente un grupo. Invóquelo como:

```
delgroup groupname
```

Todavía tiene que borrar los artículos que permanecen en el directorio de cola del grupo de noticias. Aunque se puede dejar esta tarea al proceso natural de expiración de artículos.

addmissing

Añade artículos perdidos al fichero `history`. Ejecute este guión cuando haya artículos que parezcan quedarse para siempre.

newsboot

Este guión se debería ejecutar cuando arranca el sistema. Eliminar cualquier fichero de bloqueo que se dejó atrás cuando se mataron los procesos al apagar, además cierra y ejecuta cualquier lote dejado por alguna conexión NNTP que se cerró cuando se apagó el sistema.

newsrunning

Este guión reside en `/usr/lib/news/input`, y puede ser usado para deshabilitar el desempaquetado de los lotes de noticias entrantes, por ejemplo durante las horas de trabajo. Ud. puede desconectar el desempaquetado invocando:

```
/usr/lib/news/input/newsrunning off
```

Se conecta usando on en vez de off.

Capítulo 22. NNTP y elDemonio nntpd

El Protocolo para la Transferencia de Noticias en Red (NNTP) ofrece un acercamiento al intercambio de noticias muy diferente al de C News y al de otros nuevos servidores sin soporte nativo para NNTP. Antes que confiar en una tecnología por lotes como UUCP para transferir los artículos entre máquinas, permite intercambiar los artículos mediante una conexión de red interactiva. NNTP no es un paquete de software en particular, sino un estándar de Internet descrito en el RFC−977. Está basado en conexiones continuas, normalmente sobre TCP, entre un cliente en cualquier parte de una red y un servidor en una máquina que almacena las noticias en disco. Una conexión de este tipo permite que cliente y servidor negocien de manera interactiva la transferencia de artículos sin apenas retrasos manteniendo al mismo tiempo el número de artículos duplicados muy bajo. Junto con las altas tasas de transferencia de Internet, esto conforma una nueva manera de transportar noticias que supera con creces a las redes UUCP originales. Mientras hace algunos años no era extraño que un artículo tardase dos semanas o más en llegar al último rincón de Usenet; ahora lo hace a menudo en menos de dos días. En cuanto a Internet en sí, eso implica un plazo de apenas unos minutos.

Varias órdenes permiten a los clientes descargar, enviar y publicar artículos. La diferencia entre enviar y publicar es que lo último puede afectar a artículos con información incompleta en las cabeceras; esto significa generalmente que el usuario acaba de escribir el artículo. [208] La descarga de artículos pueden llevarla a cabo tanto los clientes de transferencia de noticias como los lectores de noticias. Esto hace de NNTP una excelente herramienta para dotar de acceso a las noticias a muchos clientes de una red local sin tener que pasar por las complicaciones de usar NFS.

NNTP también proporciona un método activo y otro pasivo de transferir noticias, conocidos coloquialmente como empujar(pushing) y tirar(pulling). Pushing es básicamente lo mismo que el protocolo ihave/sendme que usa C News (descrito en Capítulo 21). El cliente ofrece un artículo al servidor mediante la orden **IHAVE msgid**, y el servidor devuelve un código de respuesta que indica si quiere el artículo o si ya lo tiene. Si el servidor quiere el artículo, el cliente se lo envía haciéndolo acabar con un punto en una línea aparte.

Empujar noticias tiene la única desventaja que supone una gran carga para el sistema servidor, al tener éste que buscar en la base de datos de su historial cada artículo de manera individual.

La técnica opuesta es seleccionar noticias, en la que el cliente solicita una lista con todos los artículos (disponibles) de un grupo que hayan llegado tras una fecha especificada. Esta petición la realiza la orden **NEWNEWS**. De los IDs de los mensajes de la lista devuelta, el cliente elige aquellos artículos que aún no tenga usando la orden **ARTICLE** para cada uno de ellos.

Seleccionar noticias necesita de un estricto control por parte del servidor en lo que se refiere a qué grupos y distribuciones se permite solicitar a un cliente. Por ejemplo, tiene que asegurarse de no enviar material confidencial de un grupo de noticias local a clientes no autorizados.

Hay algunas órdenes convenientes para los lectores de noticias que les permiten descargar las cabeceras y los cuerpos de los artículos de manera separada, o incluso líneas sueltas de las cabeceras de un determinado rango de artículos. Esto le permite mantener todas las noticias en una máquina central, con todos los usuarios de la red (presumiblemente local) usando clientes basados en NNTP para leer y publicar. Esto es una alternativa a exportar los directorios de noticias mediante NFS, tal y como se describe en Capítulo 21.

Un problema global de NNTP es que permite a una persona conocida insertar artículos en el flujo de noticias con las especificaciones del remitente falseadas. A esto se lo conoce como *falsear noticias* o *spoofing*.[209] Una extensión de NNTP le permite requerir una autentificación al usuario para ciertas órdenes, ofreciendo algunas medidas de protección contra la gente que pueda abusar de esta manera de su servidor de noticias.

Existen diferentes paquetes NNTP. Uno de los más conocidos es el demonio NNTP, también conocido como la *implementación de referencia*. Lo escribieron originalmente Stan Barber y Phil Lapsley para ilustrar los detalles del RFC−977. Como la mayoría del software de calidad disponible hoy en día, puede encontrarlo empaquetado para su distribución de Linux, o puede obtener las fuentes para compilarlo usted mismo. Si elige compilarlo usted mismo, necesitará estar familiarizado con su distribución para poder asegurarse de que configura todas las rutas de ficheros de manera correcta.

El paquete **nntpd** tiene un servidor, dos clientes para empujar y seleccionar noticias, y un sustituto de **inews**. Habitan un entorno B News, pero con unos pocos retoques también se contentarán con C News. De todos modos, si piensa usar NNTP para algo más que ofrecer acceso de los clientes de noticias al servidor, la implementación de referencia realmente no es una opción. Sólo discutiremos aquí el demonio NNTP que contiene el paquete **nntpd** prescindiendo de los clientes.

Si piensa poner en marcha un sitio de noticias de gran tamaño, debería echar un vistazo al paquete *InterNet News* o INN, que escribió Rich Salz. Proporciona transporte de noticias basado tanto en NNTP como en UUCP. El transporte de noticias es definitivamente mejor que **nntpd**. Discutiremos INN en detalle en Capítulo 23.

22.1. El Protocolo NNTP

Hemos mencionado dos órdenes NNTP que son clave en cuanto a cómo los artículos de noticias se impulsan o seleccionan entre servidores. Ahora, le echaremos un vistazo a todo esto en una sesión NNTP real para mostrarle cuán sencillo es el protocolo. Para ilustrar nuestros propósitos, usaremos un sencillo cliente **telnet** para conectar con un servidor de noticias basado en INN de la Cervecera Virtual llamado **news.vbrew.com**. El servidor está corriendo una configuración mínima para que los ejemplos sean cortos. Ya veremos cómo completar la configuración de este servidor en Capítulo 23. En nuestras pruebas pondremos especial cuidado en generar artículos sólo en el grupo de noticias *es.pruebas* para evitar molestar a nadie.

22.1.1. Conectar con el servidor de noticias

Conectar con el servidor de noticias es tan sencillo como abrir una conexión TCP con su puerto NNTP. Cuando esté conectado, aparecerá un anuncio de bienvenida. Una de las primeras órdenes que puede probar es help. La respuesta que reciba dependerá generalmente de si el servidor cree que somos un servidor NNTP o un cliente de noticias, al requerirse grupos de órdenes diferentes. Puede cambiar el modo de operación con la orden **mode**; veremos eso en un momento:

```
$ telnet news.vbrew.com nntp
Trying 172.16.1.1...
Connected to localhost.
Escape character is '^]'.
200 news.vbrew.com InterNetNews server INN 1.7.2 08−Dec−1997 ready
help
100 Legal commands
        authinfo
            help
            ihave
            check
            takethis
            list
            mode
            xmode
            quit
            head
            stat
            xbatch
            xpath
            xreplic
For more information, contact "usenet" at this machine.
.
```

Las respuestas a las órdenes NNTP siempre acaban con un punto (.) en una línea separada. Los números que ve en el listado de salida son *códigos de respuesta* que usa el servidor para indicar éxito o fallo de la orden. Los códigos de respuesta se describen en el RFC–977; hablaremos de los más importantes conforme vayamos avanzando.

22.1.2. Empujar un artículo de noticias a un servidor

Mencionamos la orden **IHAVE** cuando hablamos de hacer "pushing" de noticias a servidores. Fijémonos ahora en cómo funciona realmente la orden **IHAVE**:

```
ihave <123456@gw.vk2ktj.ampr.org>
    335
    From: terry@gw.vk2ktj.ampr.org
Subject: mensaje de prueba enviado con ihave
Newsgroups: es.pruebas
Distribution: mundo
Path: gw.vk2ktj.ampr.org
Date: 26 Abril 1999
Message-ID: <123456@gw.vk2ktj.ampr.org>
Body:

Esto es un mensaje de prueba enviado usando la orden NNTP IHAVE.

.
    235
```

Ninguna orden NNTP es sensible al uso de mayúsculas o minúsculas, por lo que resulta indiferente cómo se introduzcan las órdenes. La orden **IHAVE** toma una variable obligada, siendo el ID del mensaje lo que se impulsa. Se asigna un ID de mensaje único a cada artículo cuando se crea. La orden **IHAVE** proporciona al servidor NNTP una manera de decir qué articulos tiene cuando quiere impulsar artículos a otro servidor. El servidor que envíe usará una orden **IHAVE** para cada artículo que desee impulsar. Si el código de respuesta de la orden que genere el servidor NNTP receptor se encuentra en el rango 3xx, el servidor NNTP remitente transmitirá el artículo completo, incluyendo toda su cabecera y haciéndolo terminar con un punto en una línea aparte. Si el código de respuesta se encuentra en el rango 4xx, el servidor receptor ha elegido no aceptar el artículo, posiblemente porque ya lo tiene o a causa de algún problema, como que se esté quedando sin suficiente espacio en disco.

Cuando se ha transmitido el artículo, el servidor receptor emite otro código de respuesta indicando si la transferencia del artículo ha concluido con éxito.

22.1.3. Cambiar el modo de lectura NNRP

Los lectores de noticias usan sus propios grupos de órdenes cuando se comunican con los servidores de noticias. Para activar estas órdenes, el servidor de noticias tiene que estar operando en modo *de lectura*. La mayoría de servidores de noticias funcionan en modo de lectura de manera predeterminada, a menos que la dirección IP de la máquina con la que conecten se encuentre lista como un punto de reenvío de noticias. En cualquier caso, NNTP incluye una orden para cambiar explícitamente al modo de lectura:

```
mode reader
    200 news.vbrew.com InterNetNews NNRP server INN 1.7.2 08-Dec-1997 ready/
        (posting ok).
    help
    100 Legal commands
      authinfo user Name|pass Password|generic <prog> <args>
      article [MessageID|Number]
      body [MessageID|Number]
      date
      group newsgroup
      head [MessageID|Number]
```

```
      help
      ihave
      last
      list [active|active.times|newsgroups|distributions|distrib.pats|/
          overview.fmt|subscriptions]
      listgroup newsgroup
      mode reader
      newgroups yymmdd hhmmss ["GMT"] [<distributions>]
      newnews newsgroups yymmddhhmmss ["GMT"] [<distributions>]
      next
      post
      slave
      stat [MessageID|Number]
      xgtitle [group_pattern]
      xhdr header [range|MessageID]
      xover [range]
      xpat header range|MessageID pat [morepat...]
      xpath MessageID
    Report problems to <usenet@vlager.vbrew.com>
    .
```

En el modo de lectura de NNTP se dispone de muchas órdenes. La mayoría de ellas están diseñadas para facilitar la vida de un lector de noticias. Mencionamos antes que hay órdenes para pedir al servidor que envíe la cabecera y el cuerpo de los artículos de manera separada. También hay órdenes que listan los grupos y artículos disponibles, y otras que permiten publicar, un medio alternativo de enviar noticias a un servidor.

22.1.4. Listar los grupos disponibles

La orden **list** lista diferentes tipos de información; de manera notable los grupos que soporta el servidor:

```
list newgroups
    215 Descriptions in form "group description".
    control                  News server internal group
    junk                     News server internal group
    local.general            General local stuff
    local.test               Local test group
    .
```

22.1.5. Listar grupos activos

`list active` muestra cada grupos soportado y ofrece información sobre ellos. Los dos números de cada línea de la salida son las marcas de agua alta y bajaes decir, el artículo numerado más alto y el artículo numerado más bajo en cada grupo. El lector de noticias es capaz de hacerse una idea del número de artículos del grupo a partir de éstos. Hablaremos un poco más sobre estos números en un momento. El último campo de la salida muestra variables que controlan si se permite publicar en el grupo, si el grupo está moderado y si los artículos publicados tienen o no que guardarse. Estas variables se describen con detalle en Capítulo 23. Éste es el aspecto de un ejemplo:

```
list active
    215 Newsgroups in form "group high low flags".
    control 0000000000 0000000001 y
    junk 0000000003 0000000001 y
    alt.test 0000000000 0000000001 y
    .
```

22.1.6. Publicar un artículo

Hemos mencionado que había una diferencia entre impulsar un artículo y publicar uno. Cuando se impulsa un artículo, se asume implícitamente que el artículo ya existe, que tiene un identificador del mensaje que el servidor ha asignado al que se publicó originalmente, y que tiene un juego completo de cabeceras. Cuando se publica un artículo, se crea el artículo por primera vez y las únicas cabeceras que se aportan son las que puedan tener algún sentido para nosotros, como el título o el grupo de noticias al que se desea enviar el artículo. El servidor de noticias en el que publiquemos el artículo se encargará de añadir el resto de cabeceras y de crear un ID para el mensaje que usará cuando impulse el artículo a otros servidores.

Todo esto significa que publicar un artículo es incluso más sencillo que impulsar uno. Un ejemplo de publicar sería algo como esto:

```
post
    340 Ok
    From: terry@richard.geek.org.au
Subject: mensaje de prueba número 1
Newsgroups: es.pruebas
Body:

Esto es un mensaje de prueba, ignórelo libremente.

.
    240 Article posted
```

Hemos generado dos mensajes más como éste para dotar de algo de realismo a nuestro siguientes ejemplos.

22.1.7. Listar nuevos artículos

Cuando un lector de noticias se conecta por primera vez con un servidor de noticias y el usuario elige leer un grupo de noticias, el lector querrá descargar una lista con los nuevos artículos publicados o recibidos desde la última vez que el usuario entró al sistema. La orden `newnews` se usa con este propósito. Hay variables obligatorias que tienen que proporcionarse: el nombre del grupo o grupos a consultar, la fecha de comienzo y la hora a partir de la cual listar los mensajes. La fecha y la hora se especifican con números de seis dígitos cada una, con la información más significativa primero; *`aammdd`* y *`hhmmss`*, respectivamente:

```
newnews junk 990101 000000
    230 New news follows
    <7g2o5r$aa$6@news.vbrew.com>
    <7g5bhm$8f$2@news.vbrew.com>
    <7g5bk5$8f$3@news.vbrew.com>
    .
```

22.1.8. Elegir un grupo con el que trabajar

Cuando el usuario elige un grupo de noticias a leer, el lector de noticias puede decirle al servidor que se ha elegido ese grupo. Esto simplifica la interacción entre el lector y el servidor de noticias eliminando la necesidad de enviar constantemente el nombre del grupo con cada orden. La orden `group` simplemente toma el nombre del grupo elegido como una variable. Muchas de las órdenes siguientes usan el grupo elegido como el predeterminado, a menos que se especifique otro grupo de noticias explícitamente:

```
group es.pruebas
    211 3 1 3 es.pruebas
```

La orden `group` devuelve un mensaje en el que se indica el número de mensajes activos, la marca de agua baja, la marca de agua alta y el nombre del grupo respectivamente. Tenga en cuenta que mientras que el número de mensajes activos y la marca de agua alta coinciden en nuestro ejemplo, a menudo no es éste el caso; en un servidor de noticias activo, algunos artículos pueden haber expirado o haberse borrado, haciendo descender el número de mensajes activos pero dejando intacta la marca de agua alta.

22.1.9. Listar artículos en un grupo

Para dirigirse a los artículos de noticias, el lector tiene que saber qué números de artículos representan a los artículos activos. La orden `listgroup` ofrece una lista con los números de los artículos activos en el grupo actual o en un grupo explícito si se proporciona el nombre del grupo:

```
listgroup es.pruebas
    211 Article list follows
    1
    2
    3
    .
```

22.1.10. Descargar sólo la cabecera de un artículo

El usuario tiene que disponer de información acerca de un artículo antes de saber si desea leerlo. Hemos mencionado antes que algunas órdenes nos permiten descargar la cabecera y el cuerpo de los artículos de manera separada. La orden `head` se usa para solicitar al servidor que sólo transfiera la cabecera del artículo especificado al lector de noticias. Si el usuario no quiere leer el artículo, no hemos desperdiciado tiempo y ancho de banda transfiriendo innecesariamente el cuerpo de un artículo potencialmente grande.

Puede hacerse referencia a los artículos tanto por su número (el de la orden `listgroup`) como por el identificador del mensaje:

```
head 2
    221 2 <7g5bhm$8f$2@news.vbrew.com> head
    Path: news.vbrew.com!not-for-mail
    From: terry@richard.geek.org.au
    Newsgroups: es.pruebas
    Subject: mensaje de prueba número 2
    Date: 27 Apr 1999 21:51:50 GMT
    Organization: La Cervecera Virtual
    Lines: 2
    Message-ID: <7g5bhm$8f$2@news.vbrew.com>
    NNTP-Posting-Host: localhost
    X-Server-Date: 27 Apr 1999 21:51:50 GMT
    Body:
    Xref: news.vbrew.com es.pruebas:2
    .
```

22.1.11. Descargar sólo el cuerpo de un artículo

Si, por otra parte, el usuario decide que quiere leer el artículo, su lector de noticias necesita una manera de solicitar que se le transfiera el cuerpo del mensaje. La orden `body` se usa con este propósito. Funciona de una manera muy similar a la orden `head`, exceptuando que sólo se devuelve el cuerpo del mensaje.

```
body 2
    222 2 <7g5bhm$8f$2@news.vbrew.com> body
    Esto es otro mensaje de prueba, ignórelo también libremente.

    .
```

22.1.12. Leer un artículo de un grupo

Aunque normalmente es más eficiente transferir las cabeceras y los cuerpos de manera separada, hay ocasiones en las que puede resultarnos mejor descargar el artículo completo. Un buen ejemplo de esto es en aplicaciones a través de las que queramos transferir todos los artículos de un grupo sin ningún tipo de preselección, como cuando usamos un programa con caché NNTP tipo **leafnode**.[210]

Naturalmente, NNTP nos ofrece una forma de hacer esto, y de manera ya poco sorprendente, funciona de manera casi idéntica a como lo hace la orden `head`. La orden `article` también acepta un número de artículo o un ID de mensaje como variable, pero devuelve el artículo completo incluyendo su cabecera:

```
article 1
    220 1 <7g2o5r$aa$6@news.vbrew.com> article
    Path: news.vbrew.com!not-for-mail
    From: terry@richard.geek.org.au
    Newsgroups: es.pruebas
    Subject: mensaje de prueba número 1
    Date: 26 Apr 1999 22:08:59 GMT
    Organization: La Cerveceraa Virtual
    Lines: 2
    Message-ID: <7g2o5r$aa$6@news.vbrew.com>
    NNTP-Posting-Host: localhost
    X-Server-Date: 26 Apr 1999 22:08:59 GMT
    Body:
    Xref: news.vbrew.com es.pruebas:1

    Esto es un mensaje de prueba, ignórelo libremente.

    .
```

Si intenta descargar un artículo desconocido el servidor le devolverá un mensaje con un código de respuestas apropiado y quizá un mensaje de texto legible:

```
article 4
    423 Bad article number
```

En esta sección hemos descrito cómo se usan las órdenes NNTP más importantes. Si está interesado en desarrollar software que implemente el protocolo NNTP, debería acudir a los documentos RFC relevantes; ellos le proporcionarán información al detalle que aquí no podemos incluir.

Veamos ahora a NNTP en acción mediante el servidor **nntpd**.

22.2. Instalar el servidor NNTP

El servidor NNTP (**nntpd**) puede compilarse de dos manera dependiendo de la carga que se espere en el sistema de noticias. No hay versiones compiladas disponibles a causa de los ajustes predeterminados específicos de cada sitio que se integran en el ejecutable. Toda la configuración se lleva a cabo mediante el uso de macros definidas en `common/conf.h`.

nntpd puede configurarse tanto como un servidor independiente que se inicie durante el arranque del sistema desde un fichero **rc** o como un demonio que gestione **inetd**. En el último caso, tendrá la siguiente entrada en el `/etc/inetd.conf`:

```
    nntp    stream  tcp nowait      news    /usr/etc/in.nntpd    nntpd
```

La sintaxis de `inetd.conf` se describe con detalle en Capítulo 12. Si configura **nntpd** como independiente, asegúrese de que una línea de ese tipo en **inetd.conf** se encuentre descomentada. En cualquier caso, asegúrese de que aparece la línea siguiente en `/etc/services`:

```
    nntp    119/tcp   readnews untp    # Network News Transfer Protocol
```

Para guardar temporalmente cualquier artículo entrante, **nntpd**también necesita un directorio `.tmp` en su cola de noticias. Debería crearlo usando las siguientes órdenes:

```
    # mkdir /var/spool/news/.tmp
    # chown news.news /var/spool/news/.tmp
```

22.3. Restringir el acceso con NNTP

El acceso a los recursos NNTP lo rige el fichero `nntp_access` en `/etc/news`. En las líneas de este ficheros se describen los derechos de acceso que se garantizan a las máquinas del exterior. Cada línea tiene el siguiente formato:

```
site   read|xfer|both|no    post|no      [!exceptgroups]
```

Si un cliente conecta con el puerto NNTP, **nntpd** intentará obtener el nombre de dominio completamente cualificado de la máquina a partir de su dirección IP por medio de una búsqueda inversa. El nombre de la máquina del cliente y su dirección IP se cotejan con el campo *`site`* de cada entrada en el orden en el que aparecen en el fichero. Las coincidencias pueden ser parciales o exactas. Si una entrada coincide de manera exacta se acepta; si sólo coincide parcialmente, sólo se acepta si no hay otra coincidencia siguiéndola que sea al menos tan buena como ella. *`site`* puede especificarse de una de las siguientes maneras:

Hostname
: Esto es el nombre de dominio completamente cualificado de una máquina. Si coincide literalmente con el nombre canónico de la máquina del cliente, la entrada se acepta ignorándose todas las entradas posteriores.

IP address
: Esto es una dirección IP en notación de cuatro cifras con sus puntos correspondientes. Si la dirección IP del cliente concuerda con ésta, se acepta la entrada ignorándose todas las entradas posteriores.

Domain name
: Esto es el nombre de dominio especificado como `*.domain`. Si el nombre de la máquina del cliente concuerda con el nombre de dominio, se acepta la entrada.

Network name
: Esto es el nombre de una red tal y como se especifica en `/etc/networks`. Si el número de red de la dirección IP del cliente coincide con con el número de red asociado al nombre de la red, se entiende que la entrada coincide.

Default
: La cadena `default` concuerda con cualquier cliente.

Las entradas con una especificación del sitio más general deberían especificarse al principio, ya que se encuentran supeditadas a cualquier concordancia posterior más exacta.

Los campos segundo y tercero describen los derechos de acceso que se garantizan al cliente. El segundo campo detalla los permisos para descargar noticias impulsadas (read), y transmitirlas impulsando (xfer). Un valor de both activa ambas; no deniega el acceso en conjunto. El tercer campo garantiza al cliente el derecho a publicar artículos, p.ej., enviar artículos con la información de la cabecera incompleta y que el software de noticias se encarga de completarla. Si el segundo campo contiene no, entonces se ignora el tercer campo.

El cuarto campo es opcional y contiene una lista de grupos separados por comas el acceso a los cuales se deniega al cliente.

Esto es un fichero `nntp_access` de ejemplo:

```
#
# de manera predeterminada, cualquiera puede transferir noticias, pero no leer
# ni publicar
default                 xfer            no
#
```

```
# public.vbrew.com ofrece el acceso público mediante módem. Les permitimos
# leer y publicar en cualquier grupo excepto en los local.*
public.vbrew.com        read            post    !local
#
# el resto de las máquinas de la cervecera pueden leer y publicar
*.vbrew.com             read            post
```

22.4. Autorización NNTP

El demonio **nntpd** proporciona un sencillo esquema de autorización. Si pone en mayúsculas cualquier elemento de acceso en el fichero `nntp_access`, **nntpd** requerirá una autorización por parte del cliente para la operación respectiva. Por ejemplo, si se especifica un permiso de Xfer o XFER, (como opuesto a xfer), **nntpd** no permitirá al cliente transferir artículos a su sitio a menos que pase la autorización.

El proceso de autorización se implementa mediante una nueva orden NNTP conocida como **AUTHINFO**. Con esta orden el cliente transmite un nombre de usuario y una contraseña al servidor NNTP. **nntpd** las valida cotejándolas con la base de datos de `/etc/passwd` y verifica que el usuario pertenezca al grupo nntp.

La implementación actual de la autorización en NNTP sólo es experimental, por lo que aún no resulta muy portable. Como resultado de esto, sólo funciona con bases de datos en las que las contraseñas se encuentren en texto plano; las contraseñas ensombrecidas (shadow passwords) aún no se reconocen. Si está compilando desde las fuentes y tiene instalado el paquete PAM, podrá cambiar de una manera bastante sencilla la comprobación de la contraseña.

22.5. Interacción de nntpd con C News

Cuando **nntpd** recibe un artículo, tiene que enviárselo a un nuevo subsistema. Dependiendo de si se recibió como resultado de una orden **IHAVE** o **POST**, el artículo pasa a manejarlo **rnews** o **inews** respectivamente. En vez de invocar a **rnews**, también puede configurarlo (durante la compilación) para que procese por lotes los artículos entrantes y mueva los lotes resultantes a `/var/spool/news/in.coming`, donde se les deja para que **relaynews** los recoja la próxima vez que se ejecute la cola.

nntpd tiene que poder acceder al fichero `history` para poder ejercer de manera adecuada el protocolo ihave/sendme. Tiene que asegurarse, durante la compilación, de que la ruta a ese fichero es la correcta. Si usa C News, asegúrese de que C News y **nntpd** están de acuerdo en el formato a usar en el fichero history. C News usa funciones de marcado `dbm` para acceder a él; de todas maneras, existen implementaciones diferentes y ligeramente incompatibles de la biblioteca `dbm`. Si C News se ha enlazado con una biblioteca `dbm` diferente a la que tenga en su `libc` estándar, tendrá que enlazar **nntpd** con esa misma biblioteca.

El desacuerdo entre**nntpd** y C news produce a menudo mensajes de error en el fichero de registro del sistema que **nntpd** no puede abrir adecuadamente, o quizá vea artículos duplicados recibiéndose por NNTP. Una buena prueba para corregir errores en la transferencia de noticias es tomar un artículo de la cola, conectar por telnet al puerto **nntp** y ofrecérselo a **nntpd** como se muestra en el ejemplo siguiente. Evidentemente, tendrá que sustituir *`msg@id`*con el ID de mensaje de un artículo con el que quiera suministrar a **nntpd**:

```
$ telnet localhost nntp
Trying 127.0.0.1...
Connected to localhost
Escape characters is '^ ]'.
201 vstout NNTP[auth] server version 1.5.11t (16 November 1991) ready at
Sun Feb 6 16:02:32 1194 (no posting)
IHAVE msg@id
435 Got it.
QUIT
```

Esta conversación muestra la reacción adecuada de **nntpd**; el mensaje `Got it` le dice que ya tiene el artículo. Si en vez de eso obtiene el mensaje **335 Ok**, la búsqueda en el fichero history falló por alguna razón.

Termine la conversación con Ctrl−D. Puede mirar qué ha ido mal comprobando el fichero de registro del sistema; **nntpd** anota todo tipo de mensajes gracias a la propiedad daemon de **syslog**. Una biblioteca`dbm` incompatible se manifiesta normalmente en un mensaje quejándose de que `dbminit` falló.

Capítulo 23. Noticias de Internet

El demonio de noticias de Internet (INN por sus siglas en inglés) es discutiblemente el más popular servidor de noticias usado actualmente. INN es extremadamente flexible y es adecuado para sitios pequeños.[211] Sin embargo, se adapta muy bien a configuraciones de servidores de noticias más grandes.

El servidor INN esta compuesto por una serie de componentes, cada uno con su fichero de configuración, que serán explicados a su tiempo. La configuración de INN puede ser un poco complicada, pero se describirán cada una de las etapas en este capítulo y se proveerá de suficiente información para darle sentido a los manuales y páginas man de INN, además de ejemplos para todas sus herramientas.

23.1. Algunos aspectos internos de INN

El núcleo de INN es el demonio **innd**. La tarea de **innd** es manejar todos los artículos entrantes, los almacena y se los pasa a cualquier proveedor de noticias que los requiera. innd se activa cuando se carga el núcleo del sistema y queda trabajando de forma continua en segundo plano. Ejecutándose como demonio se incrementa el rendimiento ya que sólamente cuando se inicia se leerán los ficheros de estado. Dependiendo del volumen del proveedor de noticias, algunos ficheros como por ejemplo `history` (que contiene una lista de todos los artículos procesados recientemente) puede estar en el rango que va desde unos pocos megabytes hasta varias decenas.

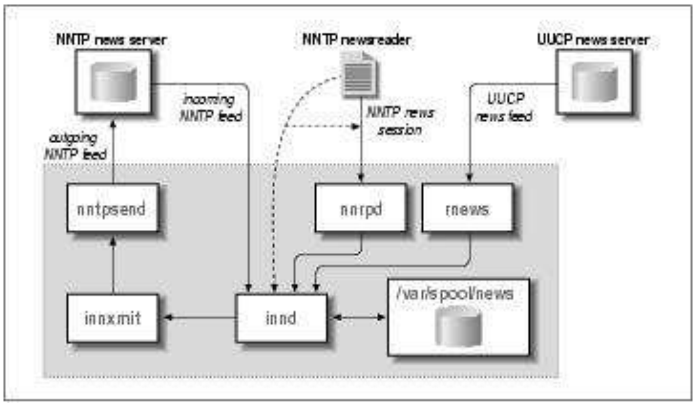
Otra característica importante de INN es que sólamente hay una copia de **innd** ejecutándose todo el tiempo. Esto es muy beneficioso a nivel de rendimiento, ya que el demonio puede procesar todos los artículos sin tener que preocuparse por el sincronismo de sus estados internos con otras copias de **innd** que se encuentran revolviendo la cola del servidor al mismo tiempo. Sin embargo, esta opción afecta el diseño global del sistema de noticias. Debido a ésto, es importante que las noticias entrantes sean procesadas lo más rápidamente, es inaceptable que el servidor esté ocupado en tareas mundanas tales como darle acceso a un cliente de noticias vía NNTP o descomprimir paquetes que lleguen por UUCP. En consecuencia, este tipo de tareas deben ser separadas del servidor principal e implementadas por otros programas. Figura 23−1 intenta ilustrar las relaciones entre **innd**, las otras tareas locales, los servidores y clientes de noticias remotos.

Hoy día, NNTP es el medio de transporte más común en cuanto a noticias se refiere, y es el único que **innd** soporta directamente. Esto significa que **innd** continuamente está escuchando el puerto 119 (TCP) y acepta las conexiones que utilizan el protocolo ihave.

Los artículos que llegan por otro tipo de transporte que no sea NNTP son soportados de forma indirecta haciendo que otros procesos acepten los artículos y se los reenvíen a **innd** por NNTP. Los paquetes que provienen de un enlace UUCP, por ejemplo, son tradicionalmente manipulados por el programa **rnews**. Este programa descomprime los paquetes si es necesario, y separa cada uno de los artículos; hecho esto, se los ofrece a **innd** uno por uno.

Los clientes de noticias, pueden entregar un artículo escrito por un usuario. Como el manejo de estos clientes merece especial atención, volveremos a este tema un poco más tarde.

Figura 23−1. Arquitectura de INN (simplificada)



Cuando recibe un artículo, **innd** primero mira el identificador el mensaje (message ID) en el fichero `history`. Los artículos y ocurrencias duplicados, son descartados y opcionalmente registrados en algún fichero de registro. Lo mismo sucede con artículos muy viejos o por ausencia de algún campo requerido, por ejemplo `Subject:`.[212]Si **innd** encuentra que el artículo está en orden, busca en el campo `Newsgroups:` para saber a qué grupos de noticias fue remitido. Si alguno o todos estos grupos se encuentran en el fichero `active`, el artículo se archiva en el disco. En caso contrario, se archiva en un grupo especial llamado junk (Basura).

Los artículos individuales se guardan en `/var/spool/news`, también llamado cola de noticias *(news spool)*. Cada grupo de noticias tiene su propio directorio, en el cuál cada artículo se guarda por separado en un fichero. Los nombres de estos ficheros son números consecutivos, por ejemplo, un artículo publicado en comp.risks se guardará como `comp/risks/217`. En el momento de guardarlo, **innd** busca el directorio donde debería ubicarse, si no se encuentra, lo crea automáticamente.

Aparte de guardar los artículos localmente, puede reenviarlos a otros servidores. Esto se controla por el fichero `newsfeeds` donde están listados todos los servidores de menor jerarquía a los cuáles se les deben pasar los artículos.

De la misma forma que **innd** gestiona el proceso de entrada de los mensajes, gestiona en una sola interfaz, los que salen. Él mismo puede gestionar todo el transporte saliente. Sin embargo, necesita varios motores que envíen los artículos a los demás servidores. Todos los recursos para el envío se llaman en forma colectiva canales[213]. Dependiendo de su propósito, un canal puede tener diferentes atributos que determinen exactamente qué información debe pasarle **innd**.

Para un suministro NNTP saliente, por ejemplo, **innd** podría bifurcar el suministro hacia el programa **innxmit** al comienzo, y por cada artículo pasarle el identificador, el tamaño, y el nombre del fichero hacia su entrada estándar, por otra parte, si se usa UUCP como suministro,**innd** puede escribir el tamaño del artículo y su nombre en un registro especial, el cuál es la cabecera de un proceso diferente a intervalos regulares en orden de crear los ficheros por lotes y hacer la cola para el subsistema UUCP.

Además de estos dos ejemplos, existen otros tipos de canales que no son estrictamente para suministros de salida. Éstos son usados, por ejemplo, cuando se desea archivar ciertos grupos de noticias, o cuando se quiere generar información general. Esta información general se crea con la intención de ayudar a los lectores de noticias a seguir el hilo de un tema de manera más eficaz. Los antiguos lectores de noticias tienen que buscar en todos los artículos de forma separada para obtener la información contenida en las cabeceras utilizada para seguir el hilo de los mensajes. Esto impone una pesada carga al servidor, especialmente cuando se usa NNTP; adicionalmente, es muy lento. [214] El mecanismo de información general alivia este problema pregrabando las cabeceras que son relevantes en un fichero separado (llamado `.overview`) por cada grupo de noticias. Esta información puede ser recogida por los lectores de noticias leyendo directamente desde el directorio donde se encuentra la cola de los mensajes, o usando la instrucción **XOVER** estando conectado vía NNTP. INN tiene al demonio **innd** para suministrar todos los mensajes usando la instrucción **overchan** la cuál se adosa al demonio a través del canal. Luego veremos este método cuando se discutan las configuraciones de los suministros de noticias.

23.2. INN y los lectores de noticias

Los lectores de noticias que se ejecutan en la misma máquina junto con el servidor (o que tienen montado el directorio donde se encuentra la cola del servidor de noticias vía NFS) pueden leer los artículos de forma directa. Para remitir un artículo compuesto por un usuario, el lector, invoca al programa **inews** el cuál agrega cualquier cabecera que esté faltando y se lo envía al demonio vía NNTP.

Alternativamente, los lectores pueden acceder al servidor remoto vía NNTP. Este tipo de conexión se gestiona de forma diferente por los proveedores de noticias basados en NNTP, para no dejar colgado el demonio. En el momento que un lector de noticias se conecta al servidor NNTP, **innd** bifurca la conexión hacia un programa llamado **nnrpd**[215], el cual maneja la sesión y libera a **innd** para que realice tareas más importantes (recibir noticias nuevas, por ejemplo). Usted estará admirado al ver cómo el proceso **innd** puede distinguir entre un proveedor de noticias entrante y la conexión de un cliente. La respuesta es simple: el protocolo NNTP requiere que el cliente emita la orden **mode reader**[216]después de conectarse al servidor; cuando esta orden se recibe, el servidor activa el proceso **nnrpd** y le entrega el control de la conexión, luego retorna a su habitual estado de escucha, esperando conexiones de otros servidores de noticias. Solía haber un lector de noticias basado en DOS que no estaba configurado para hacer esto, y fallaba de forma miserable cuando intentaba comunicarse con INN, porque **innd** no reconocía ninguno de las órdenes usadas para leer noticias si no sabía que la conexión era desde un lector de noticias

Volveremos al tema de cómo los lectores de noticias pueden acceder a INN en el capítulo «Controlando el acceso de los lectores».

23.3. Instalación de INN

Antes de sumergirse en la configuración de INN, déjeme hablarle acerca de la instalación. Lea esta sección incluso si ya tiene instalado INN de alguno de los distribuidores de GNU/Linux, pues contiene algunas sugerencias sobre seguridad y compatibilidad.

Algunos distribuidores de GNU/Linux, incluyeron la versión 1.4 de INN por algún tiempo. Desafortunadamente, esta versión contiene dos sutiles errores que comprometen la seguridad del sistema. Las versiones más modernas no tienen estos problemas y la mayoría de los distribuidores incluyen versiones precompiladas de INN versión 2 o superiores para GNU/Linux.

Si Ud. lo desea, puede compilar INN. Puede obtener los ficheros fuentes desde ftp.isc.org en el directorio `/isc/inn/`. Compilar INN requiere que escriba un fichero de configuración con algunos parámetros y datos sobre su sistema operativo que INN necesita para poder compilarse satisfactoriamente.

Compilar los paquetes por su cuenta es bastante simple; existe un guión llamado **BUILD** que lo guiará a través del proceso. Además los paquetes con el código fuente contienen mucha información de cómo instalar y configurar INN.

Después de instalar los binarios, se requieren algunos retoques manuales para que INN trabaje en armonía con otras aplicaciones que acceden a **rnews** o a **inews**. Por ejemplo, UUCP espera encontrar a **rnews** en el directorio `/usr/bin` o en `/bin`, cuando INN se instala, lo hace por omisión en `/usr/lib/bin` Asegúrese que el directorio `/usr/lib/bin/` esté incluido en ruta de acceso, o que existe un enlace simbólico que apunte hacia las órdenes **rnews** y a **inews**.

23.4. Configuración de INN: Configuración Básica

Uno de los grandes obstáculos con los que se topan los usuarios novatos es que INN requiere que algunas configuraciones de red funcionen de forma correcta, incluso en una máquina sin conexión de red. Por tal motivo, es esencial que el núcleo soporte algunos servicios de TCP/IP cuando inicie INN, además se debe tener configurada la interfaz de bucle local explicada en el Capítulo 5.

Luego, asegúrese que **innd** se activa en la secuencia de inicialización del sistema. La instalación predeterminada de INN, provee un guión llamado `boot` que se encuentra en `/etc/news/`. Si su distribución de Linux utiliza el estilo de System V (iniciando a través de **init**) sólamente se necesita crear un enlace simbólico desde `/etc/init.d/inn` que apunte a `/etc/news/boot`. Si **init** utiliza otros parámetros, asegúrese que `/etc/news/boot` se ejecuta por alguno de los guiones `rc`. Ya que INN necesita que la red esté funcionando, el guión de inicio se debe encontrar o activar *después* de que la interfaz de red se haya configurado.

23.5. INN: Ficheros de Configuración

Habiendo concluido con estas tareas generales, puede dirigirse hacia la parte más interesante de INN: Sus ficheros de configuración. Todos ellos residen en `/etc/news`. Algunos cambios se han introducido a partir de la versión 2, la cual es descrita aquí. Si tiene nstalada una versión anterior, este capítulo le puede ser útil en el momento de mejorar su configuración actual. Durante las próximas secciones, se discutirá cada fichero por separado, construyendo la configuración de la Cervecera Virtual como ejemplo.

Si necesita más información de alguna característica o de algún fichero en especial, puede consultar las páginas del manual; la distribución de INN contiene información de cada fichero por separado.

23.5.1. Parámetros Globales

INN posee un número de parámetros que son de naturaleza global; estos afectan a todos los grupos de noticias que gestiona.

23.5.1.1. El fichero inn.conf

El fichero principal de configuración de INN es `inn.conf`. Entre otras cosas, éste determina cómo se conoce su computadora en Usenet. La versión 2 de INN posee un número desconcertante de parámetros. Afortunadamente, la mayoría de éstos tienen valores predeterminados, que son razonablemente compatibles para diferentes situaciones. El fichero de ayuda `inn.conf(5)` detalla todos los parámetros, y debería leerlo con cuidado si experimenta algún problema.

Un ejemplo simple de `inn.conf`:

```
# Ejemplo de inn.conf para la Cervecera Virtual
server:          vlager.vbrew.com
domain:          vbrew.com
fromhost:        vbrew.com
pathhost:        news.vbrew.com
organization:    La cervecería vertual
mta:             /usr/sbin/sendmail -oi %s
moderatormailer: %s@uunet.uu.net
#
# Rutas de acceso y ficheros de INN.
#
pathnews:               /usr/lib/news
pathbin:                /usr/lib/news/bin
pathfilter:             /usr/lib/news/bin/filter
```

```
    pathcontrol:            /usr/lib/news/bin/control
    pathdb:                 /var/lib/news
    pathetc:                /etc/news
    pathrun:                /var/run/news
    pathlog:                /var/log/news
    pathhttp:               /var/log/news
    pathtmp:                /var/tmp
    pathspool:              /var/spool/news
    patharticles:           /var/spool/news/articles
    pathoverview:           /var/spool/news/overview
    pathoutgoing:           /var/spool/news/outgoing
    pathincoming:           /var/spool/news/incoming
    patharchive:            /var/spool/news/archive
    pathuniover:            /var/spool/news/uniover
    overviewname:           .overview
```

La primera línea le dice a **rnews** y a **inews** cuál es el servidor al que deben contactar para entregar los artículos. Esta entrada es absolutamente crucial; para pasarle artículos a **innd**, se debe establecer una conexión NNTP con el servidor.

El campo domain (dominio) debe contener la porción de la dirección del servidor que se encuentra completamente calificada. Un par de programas necesitan esta porción del nombre de dominio; si la biblioteca que resuelve los nombres, sólamente retorna nombres no calificados, el nombre dado en el campo domain se deriva hacia ella. No es un problema configurar este modo, pero es mejor definir un dominio en domain.

La siguiente línea define el nombre del servidor que **inews** va a utilizar cuando agregue la línea *`From:`* a los artículos publicados por los usuarios locales. Muchos lectores de noticias utilizan el campo *`From:`* cuando se compone un mensaje de respuesta para el autor de un artículo. Si omite este campo, por domisión será completado con el nombre de dominio entero. Ésta es siempre la mejor opción. Puede, por ejemplo, tener noticias y mensajes gestionados por diferentes servidores. En este caso, deberá dar el nombre del servidor de correo después de la declaración fromhost.

pathhost, define el nombre del servidor que INN agregará a la cabecera `Path:` cuando quiera recibir un artículo. En la mayoría de los casos, querrá utilizar el nombre del dominio de su servidor de noticias; si éste es el caso, puede omitir esta línea ya que por omisión se utiliza este nombre. Ocasionalmente, puede utilizar el nombre genérico, como por ejemplo **news.vbrew.com**,para dar servicio a un dominio grande. Haciendo esto, se puede mover el sistema de noticias fácilmente hacia un servidor diferente, cuando se requiera.

La siguiente línea contiene la clave organization. Esta declaración le permite saber a **inews** qué texto debe introducir en el campo `Organization:` de los artículos publicados por los usuarios locales. Formalmente, éste es el lugar donde debe ir una descripción de su organización, o el nombre extendido de la misma. Si no desea ser tan formal, está muy de moda, que las organizaciones con un poco de humor lo expresen aquí.

El campo mta es obligatorio y especifica la ruta de acceso y el nombre del agente de transporte de correo, usado para enviarle mensajes al moderador. `%s` se reemplaza por la dirección de correo del moderador.

La línea que contiene la entrada moderatormailer define la dirección predeterminada que se utiliza cuando un usuario intenta dejar un mensaje en un grupo de noticias que se encuentra moderado. Las direcciones de los moderadores de cada grupo usualmente son guardadas en un fichero separado, pero toma mucho tiempo seguirle los pasos a todos ellos. La entrada moderatormailer es, por consiguiente, consultada como último recurso. Si ésta está definida, **inews** reemplazará la cadena `%s` con el (sigilosamente transformado) nombre del grupo de noticias y enviará el artículo entero a esa dirección. Por ejemplo, si se desea publicar en soc.feminism, el artículo será enviado a soc−feminism@uunet.uu.net, pasándole la configuración citada. En UUNET, se debe encontrar el alias del correo, instalado para cada una de las direcciones dependientes, que serán automáticamente utilizadas para reenviar todos los mensajes al moderador apropiado.

Finalmente, cada una de las entradas restantes, especifica la ubicación de algún componente o fichero perteneciente a INN. Si instaló INN desde los paquetes, estas ubicaciones han sido creadas por usted. Por el contrario, si se decidió a compilar el sistema, debe asegurarse que estas entradas reflejen las ubicaciones donde se encuentra INN.

23.5.2. Configuración de los Grupos de Noticias

El administrador del sistema de noticias es capaz de controlar qué usuarios tienen acceso a los grupos. INN provee dos ficheros de configuración, los cuales dejan al administrador decidir cuáles son los grupos de noticias a los cuáles se les da soporte, y además proveen una descripción de cada uno de ellos.

23.5.2.1. Los ficheros active y newsgroups

Los ficheros `active` y `newsgroups` se usan para guardar y describir los grupos de noticias alojados en el servidor. En ellos se encuentran los grupos de noticias en los que se tiene interés en publicar y recibir artículos, y además, algo de información administrativa. Estos ficheros se pueden encontrar en el directorio `/var/lib/news/`.

El fichero `active` determina a qué grupos de noticias se le da soporte. Su sintaxis es lineal. Cada línea del fichero `active` contiene cuatro campos delimitados por un espacio en blanco:

```
    name himark lomark flags
```

El campo *`name`* es el nombre del grupo. El campo *`himark`* es el mayor número que se ha usado para un artículo en ese grupo. *`lomark`* es usado para guardar el número más bajo de un mensaje activo. Para ilustrar como trabaja esto, considere el siguiente escenario como ejemplo. Imagine que ha creado un grupo; *`himark`* y *`lowmark`* se encuentran en 0 por que no hay artículos. Luego se publican 5 artículos, que se numeran del 1 al 5. *`himark`* ahora estará en 5, el número del artículo más alto, y *`lowmark`* será puesto en 1, el número mas bajo. Si el artículo 5 es cancelado, no habrá ningún cambio; *`himark`* permanecerá sin cambios para asegurarse de que ese numero de artículo no sea usado nuevamente y *`lowmark`* seguirá en 1, el número de artículo más bajo. Ahora, si se cancela el artículo 1, *`himark`* permanecerá sin cambios, pero *`lowmark`* será igual a 2, ya que 1 no está más en uso. Si luego se publica un artículo nuevo, se le asignará el número 6, lo que pondrá a *`himark`* en 6. El artículo 5 ha estado en uso, así que no se usará su número por más que haya sido borrado. *`lowmark`* permanece en 2. Este mecanismo le permite encontrar un artículo fácilmente, ya que poseen números únicos y calcular de forma aproximada cuántos artículos activos hay en el grupo haciendo: *`himarklowmark`*.

El campo flag, debe contener alguno de estos parámetros:

y
: Permite la publicación de forma directa en el servidor.

n
: Publicar directamente en el servidor no está permitido. Esto previene que los lectores de noticias publiquen de forma directa los artículos en el servidor. Los artículos nuevos, deben venir de otros servidores de noticias.

m
: El grupo está moderado. Cualquier artículo publicado en este grupo es desviado hacia la dirección del moderador, para su aprobación antes de ser publicado. La mayoría de los grupos, no están moderados.

j
: Los artículos en estos grupos no son almacenados, sólamente son pasados a otro servidor. Esto causa que el servidor de noticias acepte los artículos, pero todo lo que hace es reenviarlos al siguiente servidor que se encuentra más alto en la cadena de flujo. Esto no permite que los artículos estén disponibles para lectura por parte de los usuarios de ese servidor.

x
: Este grupo de noticias no acepta artículos. La única forma de que los artículos sean recibidos por este servidor, es que provengan de otro servidor de noticias. Los lectores de noticias, no podrán acceder para publicar artículos.

`=foo.bar`
: Los artículos son guardados en el servidor local con el nombre de grupo «foo.bar».

En nuestro ejemplo de configuración, tenemos una pequeña cantidad de grupos de noticias, así que el fichero `/var/lib/news/active` se verá de este modo:

```
control 0000000000 0000000001 y
junk 0000000000 0000000001 y
rec.crafts.brewing 0000000000 0000000001 y
rec.crafts.brewing.ales 0000000000 0000000001 y
rec.crafts.brewing.badtaste 0000000000 0000000001 y
rec.crafts.brewing.brandy 0000000000 0000000001 y
rec.crafts.brewing.champagne 0000000000 0000000001 y
rec.crafts.brewing.private 0000000000 0000000001 y
```

Los números asignados a *`himark`* y a *`lomark`* son aquellos que va a usar cuando cree sus propios grupos de noticias. Estos dos números se ven un poco diferentes en grupos de noticias que han estado activos durante algún tiempo.

El fichero `newsgroups` no es muy sofisticado. Sólamente provee una breve descripción (de una sola línea) de los grupos de noticias. Algunos lectores son capaces de leer este fichero y presentarle la información al usuario para ayudarlo a decidir si quiere suscribirse al grupo descrito.

El formato del fichero `newsgroups` es el siguiente:

```
name description
```

El campo *`name`* es el nombre del grupo de noticias, y el campo *`description`* la descripción del mismo.

Para el ejemplo de la Cervecera Virtual, si deseamos poner una descripción a los grupos, cree un fichero `newsgroups` con el siguiente formato:

```
rec.crafts.brewing.ales         Elaboración casera de cerveza negra y rubia
rec.crafts.brewing.badtaste     Elaboración casera de cerveza adulterada
rec.crafts.brewing.brandy       Elaboración casera de brandy
rec.crafts.brewing.champagne    Elaboración casera de Champagne
rec.crafts.brewing.private      Grupo local de la Cervecera Virtual
```

23.5.3. Configurando los Proveedores de Noticias

INN le da al administrador la habilidad de controlar qué grupos son reenviados y de qué forma, a otros servidores de noticias. El método mas usado, utiliza NNTP (visto anteriormente) como transporte, pero pueden utilizarse otros como UUCP.

23.5.3.1. El fichero newsfeeds

En el fichero `newsfeeds` se encuentran determinados los artículos que serán enviados. Normalmente se encuentra en el directorio `/etc/news/`.

El formato de `newsfeeds` puede parecer un poco complicado al principio. Aquí será descrito de forma esquemática. Las páginas man de `newsfeeds(5)` explican cómo implementarlo. El formato es el siguiente:

```
# formato del fichero newsfeed
```

```
site:pattern:flags:param
site2:pattern2\
    :flags2:param2
```

Cada proveedor de noticias se describe en una sola línea, puede utilizarse más de una usando el carácter \ para denotar continuidad. El carácter : delimita los campos en cada línea y el carácter # es utilizado para poner comentarios.

El campo `site` nombra el sitio con el cual ese alimentador se relaciona. El nombre del sitio puede ser codificado de la forma que uno quiera y no tiene que ser el nombre del dominio del sitio. Este nombre se usará posteriormente y se referirá a una entrada en una tabla que proporciona el nombre del servidor al programa **innxmit** que transmite los artículos a través de NNTP hacia el servidor remoto. Puede tener múltiples entradas para cada sitio; cada entrada será tratada individualmente.

El campo `pattern` especifica qué grupos son enviados a ese servidor. Por omisión, son enviados todos los grupos. Si es lo que usted desea, sólamente deje este campo en blanco. Este campo es usualmente una lista de expresiones que corresponden a un patrón de búsqueda, delimitado por comas. El carácter * equivale a cualquier carácter, incluyendo al cero. El carácter . (punto) no tiene ningún significado especial, el carácter ! (usado al comienzo de una expresión) realiza la operación lógica NOT, y el carácter @ al comienzo del nombre de un grupo significa que no se envíen o reenvíe ningún articulo publicado en el grupo. Esta lista, es leída y analizada gramaticalmente de izquierda a derecha, así que asegúrese de introducir las reglas específicas al principio. Un ejemplo del patrón:

```
rec.crafts.brewing*,!rec.crafts.brewing.poison,@rec.crafts.brewing.private
```

En el ejemplo, se desea enviar todos los grupos pertenecientes a la jerarquía *rec.crafts.brewing excepto* el grupo *rec.crafts.brewing.poison*. Tampoco se enviarán o recibirán artículos para el grupo *rec.crafts.brewing.private* el cual, sólamente está disponible para los usuarios que utilizan el servidor local. Si, en el ejemplo, invierte los dos primeros patrones de búsqueda, el primero de ellos será ignorado por el segundo, y los mensajes del grupo *rec.crafts.brewing.poison* serán enviados. Lo mismo pasa con el primer y último de ellos; Siempre se deben introducir primero los parámetros más específicos, para que los menos específicos introducidos después de éestos, sean efectivos.

El campo `flags` controla y restringe los artículos que van al proveedor de noticias. Este campo (`flags`) se encuentra delimitado por comas y contiene una lista de cualquiera de las órdenes que se encuentran en la siguiente lista:

`<size`
: El tamaño del artículo debe ser menor que lo expresado, en bytes.

`Aitems`
: Los artículos serán verificados. *`items`* puede ser uno o más de `d` (deberá contener cabecera de distribución) o `p` (no se verificará el destino en la cabecera path).

`Bhigh/low`
: Define el tamaño del búffer antes de escribirlo en la salida.

`H[count]`
: El artículo deberá tener por lo menos *`count`* saltos; por omisión es 1.

`Isize`
: Tamaño del búffer interno (para el fichero de salida).

`Mpattern`
: Sólo los grupos moderados pueden hacer uso del patrón.

`Npattern`

Sólo los grupos sin moderar pueden hacer uso del patrón.

Ssize
: Iniciar la cola de mensajes si el tamaño especificado en bytes es alcanzado.

Ttype
: Tipo de alimentación con el proveedor: `f` (fichero), `m` (canalizar; el campo *`param`* contiene el nombre al cual serán suministrados los artículos), `p` (tubería (pipe) que apunta a un programa), `c` (envía al canal de stdin los parámetros en *`param`*), y `x` (parecido al parámetro c pero manejando los comandos de stdin).

Witems
: Que se escribirá: `b` (el tamaño del artículo en bytes), `f` (la ruta de acceso completa), `g` (el primer grupo de noticias), `m` (el identificador de artículo), `n` (la ruta de acceso relativa), `s` (origen del artículo), `t` (antigüedad), `*` (nombre del canal alimentador o todos los lugares donde llegará el artículo), `N` (cabecera del grupo de noticias), `D` (cabecera de distribución), `H` (todas las cabeceras), `O` (datos de información general), y `R` (datos de réplica).

El campo `param` tiene una codificación especial que es dependiente del tipo de suministro. En las configuraciones más comunes es donde se especificará el nombre del fichero de salida donde escribirá el suministro de salida. En otras configuraciones, puede dejarlo fuera. También, dependiendo de la configuración, puede tener otro significado. Si desea que realice algo inusual, el las páginas man de `newsfeeds(5)` le explicarán el uso de `param` con algunos detalles.

Existe un nombre especial que debe ser codificado como `ME` y debe ser la primera línea en el fichero. Esta entrada sirve para controlar las configuraciones preeterminadas para los suministros de noticias. Si la entrada `ME` tiene una lista de distribución asociada con ella, esta lista será anexada con cada una de las otras entradas antes de que se envíen. Ésto le permite a Ud., por ejemplo, declarar cuales grupos de noticias serán automáticamente suministrados, o bloqueados de suministro, sin tener que repetir el patrón para cada entrada.

Se mencionó anteriormente que es posible el uso de suministros especiales para generar hilos de mensajes, haciendo más fácil el trabajo de los lectores de noticias. Esto es posible mediante la instrucción **overchan** que es parte del paquete INN. Para hacer esto, se debe crear un suministro especial llamado `overview` que será el que pase los artículos al programa **overchan** para luego ser procesados como información general.

El servidor de noticias mostrado como ejemplo, provee un solo suministro, que se dirige hacia la Universidad Groucho Marx y recibe los artículos de todos los grupos excepto de *control* y de *junk*, el grupo *rec.crafts.brewing.private* queda para uso local sólamente, y el grupo *rec.crafts.brewing.poison* que no queremos que la gente de la Cervecera pueda publicar.

Se utiliza el comando **nntpsend** para el transporte de noticias vía NNTP hacia news.groucho.edu. **nntpsend** requiere el uso del método del fichero para la entrega, además de la ruta de acceso completa y el identificador de cada artículo. Cabe destacar que el campo *`param`* ha sido configurado con el nombre del fichero de salida. Volveremos al tema del comando **nntpsend** en un momento. El resultado del suministro de noticias es el siguiente:

```
# fichero /etc/news/newsfeeds para la Cervecería Virtual
#
# Envía todos los grupos por defecto, excepto control y junk
ME:!control,!junk::
#
# Genera ingormacion general para cualquier lector que se utilice.
overview::Tc,WO:/usr/lib/news/bin/overchan
#
# Alimenta a Groucho Marx University con todo, excepto el grupo privado
# y cualquier artículo publicado en rec.crafts.brewing.poison.
```

```
gmarxu:!rec.crafts.brewing.poison,@rec.crafts.brewing.private:\
    Tf,Wnm:news.groucho.edu
#
```

23.5.3.2. El fichero nntpsend.ctl

El programa **nntpsend** maneja la transmisión de los artículos usando NNTP como protocolo invocando al comando **innxmit**. Hemos hecho un vistazo de **nntpsend** anteriormente, pero él también dispone de un fichero de configuración para flexibilizar a nuestros suministros de noticias.

nntpsend espera encontrar ficheros de guiones para los grupos que suministra. La ruta que el comando necesita para encontrar los guiones, sigue el siguiente patrón `/var/spool/news/out.going/`*`sitename`*. **innd** crea esos guiones cuando actúa como entrada en `newsfeeds`, como ya hemos visto anteriormente. Se especifica el nombre del sitio como nombre de fichero en el campo *`param`* y eso satisface los requerimientos de la entrada al comando **nntpsend**.

El programa **nntpsend** tiene un fichero de configuración llamado `nntpsend.ctl` que generalmente es guardado en el directorio `/etc/news/`.

El fichero `nntpsend.ctl` le permite asociar un nombre de dominio completo, algunas restricciones acerca del tamaño de los suministros, y un número de parámetros acerca de las transmisiones de un sitio en particular. El nombre del sitio significa excepcionalmente un suministro lógico de los artículos. El formato general es el siguiente:

```
sitename:fqdn:max_size:[args]
```

La siguiente lista describe los elementos de este formato:

`sitename`
: El nombre del sitio escrito en el fichero `newsfeeds`.

`fqdn`
: El nombre de dominio completo del servidor de noticias que será suministrado con los artículos.

`max_size`
: El máximo volumen de artículos a suplir en una sola transferencia.

`args`
: Argumentos adicionales que serán pasados el comando **innxmit**.

Nuestro ejemplo de configuración requiere un fichero `nntpsend.ctl` muy sencillo. Sólo existe un suministro de noticias. Se restringe el tamaño máximo de tráfico a 2 MB y se pasa como argumento a **innxmit** un tiempo de espera de 3 minutos (180 segundos). Si se posee un sitio más grande, simplemente se pueden crear nuevas entradas por cada suministro, que en tal caso se verán muy parecidas a ésta:

```
# /etc/news/nntpsend.ctl
#
gmarxu:news.groucho.edu:2m:-t 180
#
```

23.5.4. Controlando el acceso de los Lectores de Noticias

No hace mucho tiempo, era muy común que las organizaciones dieran acceso público a sus servidores de noticias. Hoy día es muy difícil encontrar acceso público a algún servidor; la mayoría de las organizaciones, controlan cuidadosamente quién tiene acceso a sus servidores, típicamente conceden acceso sólamente a los usuarios de su red. INN provee ficheros de configuración para controlar esos accesos.

23.5.4.1. El fichero incoming.conf

Se mencionó en la introducción a INN que su eficiencia consiste en separar el mecanismo de suministro del mecanismo de lectura de noticias. El fichero `/etc/news/incoming.conf` es donde se especifica qué servidores van a proveerle de noticias usando el protocolo NNTP, además de algunos parámetros de control y cómo serán suministrados los artículos. Cualquier servidor que no se encuentre en la lista, no será gestionado por el demonio **innd** en cambio, podrá gestionarse con el demonio **nnrpd**.

La sintaxis del fichero `/etc/news/incoming.conf` es bastante simple, pero toma algún tiempo acostumbrarse a ella. Tres tipos de entradas están disponibles. Pares de clave/valor, para claves específicas con su valor respectivo; compañeros (peers), que especifican el nombre de los servidores que tienen permitido enviar artículos usando NNTP; y los grupos (groups) , que es la manera de aplicar los pares (pairs) de clave/valor a los grupos (groups) de compañeros . Los pares clave/valor tienen tres tipos diferentes de ámbitos. Global, el cual abarca a cualquier par definido en el fichero. Grupos de pares, que son aplicadas sólamente a un grupo determinado. Pares que son aplicadas a un solo compañero. Las definiciones específicas anulan a las definiciones más amplias: por consiguiente, las definiciones de compañeros (peers) anulan a las de grupos (groups), que a su vez anulan a las globales.

Las llaves (`{}`) se usan para delimitar el inicio y fin de un grupo (`group`) y las especificaciones de los compañeros (`peer`). El carácter # se usa como comentario. Los pares (`pairs`) clave/valor se separan por dos puntos (:) y aparecen de una a una en diferentes líneas.

Existe un número de claves diferentes. Las más comunes y útiles son:

hostname
: Esta clave, separada por comas, especifica una lista de los nombres o direcciones IP de los servidores compañeros (peers) que están autorizados a enviar artículos. Si esta clave no se pone, se usará como nombre del servidor de la estiqueta del compañero (peer).

streaming
: Esta clave determina si el servidor puede enviar flujos de órdenes. El valor es booleano, que por omisión está establecido a verdadero (`true`).

max−connections
: Aquí se especifica el número máximo de conexiones que puede aceptar el grupo de compañeros (peers). Si el valor es cero, son ilimitadas (también se puede especificar usando `none`).

password
: Puede especificar una contraseña que será usada por el compañero (peer) si éste está autorizado a transferir noticias. Por omisión no se requiere el uso de contraseñas.

patterns
: Esta clave especifica qué grupos de noticias serán aceptados del compañero (peer) asociado. Este campo es codificado precisamente con las mismas reglas que se usan en el fichero `newsfeeds`.

En el ejemplo de la Cervecera existe un solo servidor que espera suplirnos de noticias: el servidor de la Universidad Groucho Marx. No se requiere una contraseña, pero nos aseguraremos de que no introduzca ningún artículo a nuestro grupo privado desde el exterior. El fichero `hosts.nntp` se muestra así:

```
# Cervecera virtual, fichero incoming.conf

# Parámetros globales
streaming:       true
max-connections: 5

# Permitir la publicación de artículos por NNTP de un cliente local.
peer ME {
    hostname: "localhost, 127.0.0.1"
}

# Permitirle a Groucho el acceso a todos los grupos excepto al privado.
peer groucho {
    hostname: news.groucho.edu
    patterns: !rec.crafts.brewing.private
}
```

23.5.4.2. El fichero nnrp.access

Se mencionó anteriormente, que los lectores de noticias, y los servidores que no estén en la lista del fichero `hosts.nntp`, para conectarse al servidor de INN se gestionan por el programa **nnrpd**. Este programa utiliza el fichero `/etc/news/nnrp.access` para determinar quién está autorizado a acceder al servidor de noticias, y qué tipo de permisos tiene.

El fichero `nnrp.access` contiene una estructura similar a la vista en la configuración anterior. Está compuesto por un conjunto de patrones usados para encontrar equivalencias con nombres o direcciones IP de los servidores, y algunos campos que determinan qué tipos de permisos se les conceden. Cada entrada debe aparecer en una sola línea; y los campos, separados por dos puntos. La última entrada de este fichero, debe coincidir con el nombre del servidor que va a ser usado; así que nuevamente, deben introducirse los patrones generales primero, seguidos de los más específicos. Los cinco campos deben ser introducidos en el orden en que aparecen en la siguiente lista:

Hostname o dirección IP
: Este campo, está sujeto a las reglas sobre identidades que establece `wildmat(3)`, y describe los nombres o direcciones IP de los servidores.

Permissions
: Aquí se determinan los permisos que tendrá el servidor. Existen dos permisos: `R` le otorga permisos de sólo lectura y `P` permisos de publicación.

Username
: Este campo es opcional y le permite a especificar el nombre de usuario del cliente NNTP que deberá autenticarse en el servidor antes de publicar algún artículo. Puede dejarse en blanco. No se pedirá autenticación alguna para leer artículos.

Password
: También es opcional, y es la contraseña que acompaña al campo anterior (*`username`*). Dejándolo en blanco, no se pedirá ninguna contraseña para publicar artículos

Newsgroups
: Este campo especifica un patrón de cuáles son los grupos a los que el cliente tiene permitido el acceso. Este patrón sigue las mismas reglas usadas en el fichero `newsfeeds` La opción

predetermnada, es no acceder a ninguno, así que normalmente debería existir algún patrón configurado aquí.

En el ejemplo de la Cervecera Virtual, dejamos que cualquier cliente vía NNTP en el dominio de la cervecera, pueda leer y publicar en cualquier grupo de noticias. Además se da acceso a cualquier cliente por NNTP (fuera del dominio) sólamente para leer cualquier grupo de noticias excepto el grupo privado. Nuestro ejemplo del fichero `nnrp.access` se ve de esta forma:

```
    # Cervecera Virtual - fichero nnrp.access
    # Se permite la lectura pública de todos los grupos excepto el privado.
    *:R:::*,!rec.crafts.brewing.private

    # Cualquier servidor que pertenezca al dominio de la Cervecera
    # Virtual, puede publicar y leer los artículos de cualquier grupo.
    *.vbrew.com:RP::*
```

23.5.5. Caducidad de Artículos

Cuando los artículos se reciben en el servidor, se guardan en el disco. Estos artículos deben estar disponibles un cierto período de tiempo para que su uso sea eficaz, de modo que los grandes servidores de noticias, consumen mucho espacio en disco manteniéndolos. Para asegurarse de que espacio en disco se usa de forma efectiva, puede optar por eliminar automáticamente algunos artículos después de un período de tiempo. Este proceso se llama **expiración de artículos**. Naturalmente, INN proporciona una manera automática para caducar los artículos.

23.5.5.1. El fichero expire.ctl

El servidor INN utiliza un programa llamado **expire** para eliminar los artículos caducos. Este programa, utiliza un fichero llamado `/etc/news/expire.ctl` donde se encuentran las reglas que van a dirigir la eliminación de los artículos.

La sintaxis del fichero `/etc/news/expire.ctl` es medianamente simple. Como la mayoría de los ficheros de configuración, las líneas en blanco y las que comienzan con el símbolo #, son ignoradas. La idea general, es que especifique una regla por línea. Cada una de estas reglas definen como se ejecutarán la tareas de expiración en los grupos que concuerden con el patrón suministrado. La sintaxis de una regla se ve de esta forma:

```
patrón:opciones_moderados:mantener:omisión:purgar
```

La siguiente lista describe cada campo:

patrón
: Este campo está delimitado por comas, y contiene una lista de los nombres o patrones de búsqueda para los grupos de noticias. La rutina `wildmat`(3) se usa para buscar con los patrones dados. La última regla que coincida con el nombre de un grupo es la única que va a ser aplicada, o sea que si desea aplicar patrones con comodines (*), se deben encontrar al principio del fichero.

opciones_moderados
: Este parámetro describe cómo se aplica una regla a un grupo moderado. Puede usarse una `M` que asigne esa regla sólamente a los grupos moderados, una `U` para los grupos que no están moderados, o una `A` la cuál significa que se ignore el estado de moderado y se aplique la regla a todos los grupos de noticias.

mantener

El siguiente campo, le permite especificar el tiempo mínimo que debe mantenerse guardado un artículo que contenga la cabecera de expiración. La unidad de tiempo es días, y este valor se guarda como una variable de punto flotante, así que puede especificar valores como `7.5` para siete días y medio. También puede especificar `never` si desea que los artículos se queden en el servidor para siempre.

omisión
: El campo más importante, el cual le permite especificar el tiempo que un artículo sin cabecera de expiración permanece en el grupo. La mayoría de los artículos no tienen cabecera de expiración (`expires`). Este campo se codificado de la misma forma que el campo `mantener`, donde `never` significa que los artículos sin la cabecera no expiran nunca

purgar
: Este campo le permite especificar el tiempo máximo que un artículo con cabecera de expiración será guardado después de que expire. La codificación de este campo es la misma que para el campo `mantener`.

Para la cervecera, nuestros requerimientos son simples. Serán guardados todos los artículos en todos los grupos 14 días por omisión, y entre 7 y 21 días los artículos que tengan cabecera de expiración (`Expires`). El grupo *rec.crafts.brewing.private* es interno, así que se prestará atención para que ningún artículo del mismo expire:

```
# fichero expire.ctl file de la Cervecera Virtual

# Los artículos expiran en 14 días por omisión,
# entre 7 y 21 días los que contengan cabecera Expires:
*:A:7:14:21

# Este es un grupo especial donde los artículos nunca expiran.
rec.crafts.brewing.private:A:never:never:never
```

Existe una entrada especial que debe estar en el fichero `/etc/news/expires.ctl`. Debe tener una línea en el fichero exactamente como se muestra a continuación:

```
/remember/:días
```

Esta entrada le permite especificar el número mínimo de días que un artículo será recordado en el fichero de historial, sin tomar en cuenta de que el mismo haya expirado o no. Esto puede ser útil si uno de los sitios que le proveen de noticias es de uso poco frecuente o si tiene el hábito de enviarle artículos viejos o los mismos cada vez que accede a él. Introduciendo el campo */remember/* ayudará a prevenir que el servidor del cual se provee, vuelva a enviarle los mismo artículos incluso cuando éstos ya hayan caducado en su servidor local. Su servidor podrá recordar si un artículo ya ha sido recibido, y rechazará cualquier intento remoto de reenvío. Es importante recordar que esta configuración no surte efecto en todos los artículos; sólamente afecta a aquellos que se encuentran guardados en el historial. .

23.5.6. Gestión de los Mensajes de Control

Igualmente que con C News, INN puede procesar mensajes de control de forma automática. INN proporciona un mecanismo de configuración muy potente para controlar qué acción se toma para uno de los mensajes de control y un mecanismo de control de acceso puede iniciar acciones contra algún grupo de noticias.

23.5.6.1. El fichero control.ctl

La estructura del fichero `control.ctl` es bastante simple. Su sintaxis es muy parecida a otros ficheros de configuración que posee INN. Las líneas que comienzan con # (comentarios) son ignoradas, para continuar con una línea, se usa /, y los campos se delimitan con dos puntos `:`.

Cuando se recibe un mensaje de control, se verifica con cada regla hasta el final. La última regla en el fichero que coincida con un mensaje es la que será usada, de modo que se deben poner primero las reglas genéricas y luego las más especificas al final del fichero. La sintaxis general es la siguiente:

```
mensaje:desde:grupo_noticias:acción
```

El significado de cada uno de los campos es el siguiente:

mensaje
: Este es el nombre del mensaje de control. Los mensajes de control típicos se describen después.

desde
: Éste es un patrón de búsqueda al estilo del intérprete de órdenes para buscar a la persona que envía el mensaje. La dirección de correo se convierte a minúsculas antes de la comparación.

grupos_noticias
: Si el mensaje de control es `newgroup` o `rmgroup`, este campo es un patrón de búsqueda al estilo línea de órdenes para crear o eliminar grupos.

acción
: Este campo especifica qué acción se debe tomar para cualquier mensaje que coincida con la regla. Existen muchas acciones que se pueden tomar; éstas se describen en la siguiente lista.

El campo *`mensaje`* de cada línea puede contener uno de estos valores:

checkgroups
: Este mensaje envía una petición al administrador de noticias para que re−sincronice la base de datos de los grupos de noticias con la lista adjuntada en el mensaje.

newgroup
: Este mensaje envía una petición para la creación de un nuevo grupo. El cuerpo del mensaje de control debe contener una descripción corta del propósito de la creación de un nuevo grupo.

rmgroup
: Petición de eliminar a un grupo.

sendsys
: Este mensaje envía una petición para que el fichero `sys` del servidor de noticias sea transmitido por correo al creador del mensaje de control. En el documento RFC−1036 se describe que éste es un requisito de los miembros de Usenet que este fichero esté disponible públicamente ya que se usado para que el mapa de Usenet esté actualizado.

version
: Este mensaje envía una petición para que el nombre y la versión del software utilizado por el servidor de noticias le retornen al creador del mensaje de control.

all
: Este es un código especial que compara cualquier mensaje de control.

El campo *`mensaje`* debe contener alguna de las siguientes acciones:

doit
: El comando requerido es ejecutado. En muchos casos, se envía un mensaje al administrador comunicándole qué acción ha temido lugar.

doit=`fichero`
: Esta acción es similar a `doit` excepto que crea un mensaje en el (*fichero*) de registro. Si el nombre del fichero especificado es *mail*, la entrada de registro se envía por correo. Si la cadena especificada es nula, el mensaje de registro se escribe en `/dev/null` lo que equivale a una acción descalificada (`doit`). Si el nombre del (*fichero*) comienza con el carácter / , el nombre se toma como un nombre de fichero absoluto; por el contrario, si no se especifica, será trasladado a `/var/log/news/file.log`.

doifarg
: La orden requerida se ejecuta si contiene algún argumento. Si no contiene ningún argumento, el mensaje de control se ignora.

drop
: El mensaje solicitado se ignora.

log
: Se envía a la salida un mensaje de registr `stderr` del proceso **innd**. Ésto generalmente está dirigido al fichero `/var/log/news/errlog`.

log=`file`
: Es igual a la acción `log` excepto que el fichero de registro está especificado como un par de reglas dadas de la acción `doit=`*`fichero`*.

mail
: Se envía un correo al administrador de noticias conteniendo un pedido de detalle de un comando. Ninguna otra acción toma lugar.

*verify−**
: Si una acción comienza con la cadena `verify−`, el mensaje de control es autenticado usando PGP (o GPG). [217]

A continuación se muestra el fichero `control.ctl` en la práctica. Esto es un ejemplo ilustrativo del fichero, bastante limitado:

```
## Ejemplo de /etc/news/control.ctl
##
## Cuidado: No se debe utilizar este fichero, es suministrado
## sólamente con propósitos ilustrativos.

##  Manejo de mensajes de control
all:*:*:mail
checkgroups:*:*:mail
ihave:*:*:drop
sendme:*:*:drop
sendsys:*:*:log=sendsys
senduuname:*:*:log=senduuname
version:*:*:log=version
newgroup:*:*:mail
rmgroup:*:*:mail

##  Manejo de mensajes de control para las ocho jerarquías más importantes
##  COMP, HUMANITIES, MISC, NEWS, REC, SCI, SOC, TALK
checkgroups:*:comp.*|humanities.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:drop
newgroup:*:comp.*|humanities.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:drop
```

```
rmgroup:*:comp.*|humanities.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:drop
checkgroups:group-admin@isc.org:*:verify-news.announce.newgroups
newgroup:group-admin@isc.org:comp.*|misc.*|news.*:verify-news.announce.newgroups
newgroup:group-admin@isc.org:rec.*|sci.*|soc.*:verify-news.announce.newgroups
newgroup:group-admin@isc.org:talk.*|humanities.*:verify-news.announce.newgroups
rmgroup:group-admin@isc.org:comp.*|misc.*|news.*:verify-news.announce.newgroups
rmgroup:group-admin@isc.org:rec.*|sci.*|soc.*:verify-news.announce.newgroups
rmgroup:group-admin@isc.org:talk.*|humanities.*:verify-news.announce.newgroups

## GNU ( Free Software Foundation )
newgroup:gnu@prep.ai.mit.edu:gnu.*:doit
newgroup:news@*ai.mit.edu:gnu.*:doit
rmgroup:gnu@prep.ai.mit.edu:gnu.*:doit
rmgroup:news@*ai.mit.edu:gnu.*:doit

## LINUX (Suministro para news.lameter.com)
checkgroups:christoph@lameter.com:linux.*:doit
newgroup:christoph@lameter.com:linux.*:doit
rmgroup:christoph@lameter.com:linux.*:doit
```

23.6. Activación de INN

Los paquetes con los fuentes de **inn** proporcionan un guión adecuado para que **inn** se active cuando se inicia el sistema. Este guión se llama comúnmente `/usr/lib/news/bin/rc.news`. El guión lee los argumentos que provienen de otro guión usualmente llamado `/usr/lib/news/innshellvars`, el cuál contiene las definiciones de los nombres de ficheros y rutas de búsqueda que **inn** usará para localizar los componentes que necesita. Generalmente, es buena idea ejecutar **inn** con permisos que no sean del root, como por ejemplo bajo el usuario `news`.

Para asegurarse de que **inn** se activa al inicio del sistema, se debe virificar que `/usr/lib/news/innshellvars` esté configurado correctamente y que llame al guión `/usr/lib/news/bin/rc.news` para que se ejecute al inicio.

Adicionalmente, existen algunas tareas administrativas que deben realizarse de forma periódica. Estas tareas son usualmente ejecutadas por el programa **cron**. La mejor forma de hacer esto, es añadir las instrucciones apropiados al fichero `/etc/crontab` o incluso mejor, crear un fichero adecuado en el directorio `/etc/cron.d`, si su distribución proporciona uno. Un ejemplo de esto, se muestra de la siguiente forma:

```
# Ejemplo de /etc/cron.d/inn , utilizado en Debian.
#
SHELL=/bin/sh
PATH=/usr/lib/news/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Expire noticias antiguas y le de un vistazo a las entradas
# en la noche, generar informes.

15 0 * * *      news    news.daily expireover lowmark delayrm

# cada hora, ejecutar rnews -U. Esto no es sólamente para sitios
# UUCP, también se procesarán los artículos en espera puestos
# aquí por in.nnrpd en caso de que innd no pueda aceptar
# ningún artículo.

10 * * * *      news    rnews -U
```

Estas órdenes aseguran que las noticias antiguas expiren cada día, y que los artículos en espera se procesen cada hora. Fíjese que estas instrucciones son ejecutadas con los permisos del usuario `news`.

23.7. Uso de INN: El programa ctlinnd

El servidor de noticias INN acude a una programa para manejar las tareas cotidianas. El programa **ctlinnd** puede usarse para manipular los grupos de noticias, y los suministros de estos; además, puede cambiar a diferentes estados el servidor de noticias.

Puede obtener un sumario del la sintaxis del programa **ctlinnd** usando:

```
# ctlinnd -h
```

Se cubrirán algunos de los usos más importantes de **ctlinnd** aquí; puede consultar las páginas man para más detalles.

23.7.1. Añadir un Grupo Nuevo

Utilice la siguiente sintaxis para agregar un grupo nuevo:

```
    ctlinnd newgroup grupo resto creador
```

Los argumentos son:

grupo
: El nombre del grupo que desea crear.

resto
: Este argumento debe ser codificado en la misma forma que el campo *flags* en el fichero `active`. Por omisión `y` no es proporcionado.

creador
: El nombre de la persona que crea el grupo. Enciérrelo entre comillas si existe algún espacio en el nombre.

23.7.2. Cambiar un Grupo

Utilice la siguiente sintaxis para cambiar las propiedades de un grupo:

```
    ctlinnd changegroup grupo resto
```

Los argumentos son:

grupo
: El nombre del grupo que desea cambiar.

resto
: Este argumento debe ser codificado de la misma forma que el campo *opciones* en el fichero `active`.

Este comando es útil a la hora de cambiar los atributos de moderación de un grupo.

23.7.3. Eliminar un Groupo

Utilice la siguiente sintaxis para eliminar un grupo:

```
    ctlinnd rmgroup grupo
```

El argumento es:

grupo
: El nombre del grupo a eliminar.

Este comando elimina al grupo especificado del fichero `active`. No tiene ningún efecto sobre la cola de mensajes de noticias. Todos los artículos que se encuentren en la cola, expirarán de la manera acostumbrada, pero no se aceptará ningún artículo nuevo.

23.7.4. Renumerar un Grupo

Utilice la siguiente sintaxis para renumerar un grupo:

```
    ctlinnd renumber grupo
```

El argumento es:

grupo
: El nombre del grupo a renumerar. Si el argumento **grupo** es una cadena vacía (sin argumento), todos los grupos volverán a numerarse.

Esta instrucción actualiza la marca de agua en los grupos especificados.

23.7.5. Permitir / Denegar el acceso de los Lectores de Noticias

Utilice la siguiente sintaxis para permitir o denegar el acceso de los lectores de noticias:

```
    ctlinnd readers flag texto
```

Los argumentos son:

flag
: Especificando `n` causa que cualquier intento de acceder al servidor sea denegado, por el contrario, si se especifica `y` se concederá acceso al mismo.

texto
: El texto introducido en *texto*, les da a los lectores de noticias que intentan acceder, una explicación de por qué el servidor rechaza las conexiones. Cuando el servidor vuelve a conceder el acceso, este campo debe estar en blanco o con el mismo texto que se ingresó al desactivarlo.

  Este comando no controla a los proveedores de noticias, sólamente controla las conexiones de los lectores.

23.7.6. Rechazar las conexiones de los proveedores

Utilice la siguiente sintaxis para rechazar las conexiones de los proveedores:

```
    ctlinnd reject razón
```

El argumento es:

razón
: El texto introducido debe explicar por qué **innd** rechaza las conexiones de los proveedores de noticias.

Este comando no afecta las conexiones manejadas por **nnrpd** ;sólamente afecta aquellas conexiones que **innd** maneja directamente, como por ejemplo, los proveedores de noticias remotos.

23.7.7. Permitir el acceso a los proveedores

Utilice la siguiente sintaxis para permitir el acceso de los proveedores de noticias:

```
ctlinnd allow razón
```

El argumento es:

razón
: El texto ingresado debe ser el mismo que se ingreso para el comando **reject** , o una cadena en blanco.

Este comando revierte la situación creada por **reject**.

23.7.8. Desactivar el servidor de noticias

La siguiente sintaxis desactiva el servidor de noticias:

```
ctlinnd throttle reason
```

El argumento es:

reason
: La razón por la cual se desactiva el servidor.

Este comando es equivalente a `newsreaders no` y a `reject`, es útil cuando se deben realizar tareas de emergencia en la base de datos de noticias. Esto le asegura que nada intente actualizarse mientras se encuentra trabajando en el servidor.

23.7.9. Reinicio del servidor

La siguiente sintaxis reinicia el servidor de noticias:

```
ctlinnd go razón
```

El argumento es:

razón
: La misma razón que se dio para desactivarlo. Si este campo está vacío, el servidor se activará incondicionalmente. Si una razón es dada, sólamente las funciones donde concuerde la razón dada, se activarán en el servidor.

Este comando es utilizado para iniciar el servidor después de que un comando **throttle**, **pause**, o **reject** es ejecutado.

23.7.10. Mostrar el estado de un proveedor de noticias

Utilice la siguiente sintaxis para mostrar el estado de un proveedor:

```
ctlinnd feedinfo site
```

El argumento es:

site
: El nombre del sitio (tomado del fichero `newsfeeds`) por cada uno de los proveedores que se desea ver el estado en que se encuentran.

23.7.11. Baja de un proveedor

La siguiente sintaxis es utilizada para dar de baja a un proveedor:

```
ctlinnd drop sitio
```

El argumento es:

sitio
: El nombre del sitio (tomado del fichero `newsfeeds`) que desea darse de baja. Si el campo está en blanco, todos los proveedores activos serán dados de baja.

Darle de baja a un proveedor, detiene cualquier suministro activo, pero éste no es un cambio permanente. Este comando es útil si se desean modificar algunos valores para el proveedor y éste se encuentra activo en el momento.

23.7.12. Activar un proveedor

Utilice la siguiente sintaxis para activar un proveedor:

```
ctlinnd begin site
```

El argumento es:

site
: El nombre del sitio que se encuentra en el fichero `newsfeeds` el cuál será activado. Si el proveedor se encuentra activo, el comando **drop** es ejecutado primero de forma automática.

Este comando causa que el servidor vuelva a leer el fichero `newsfeeds`, localizando la entrada ingresada y comenzar el suministro de noticias usando los detalles encontrados. Puede utilizarse este comando para probar el funcionamiento de un proveedor nuevo o si realizó alguna modificación en alguna entrada del fichero `newsfeeds`.

23.7.13. Cancelar un artículo

La siguiente sintaxis es utilizada para cancelar un artículo:

```
ctlinnd cancel Message-Id
```

El argumento es:

Message-ID
: El identificador del artículo.

El comando elimina al mensaje especificado del servidor. Esto no genera un mensaje que advierta la operación.

Capítulo 24. Configuración del lector de noticias

Un **lector de noticias** es un programa que los usuarios invocan para ver, almacenar, y crear artículos de noticias. Varios lectores de noticias han sido portados a GNU/Linux. Se describirá la configuración básica para los tres lectores de noticias más populares: **tin**, **trn** y **nn**.

Uno de los lectores más efectivos es:

```
    $ find /var/spool/news -name '[0-9]*' -exec cat {} \; |
more
```

Así es como los unixeros a ultranza leen sus noticias.

La mayoría de los lectores de noticias, sin embargo, son mucho más sofisticados. Generalmente ofrecen una interfaz a pantalla completa con niveles separados para mostrar todos los grupos a los que el usuario está suscrito, una descripción general de todos los artículos de cada grupo, y artículos individuales. Muchos navegadores web hacen las funciones de lectores de noticias, pero si usted quiere utilizar sólo un lector de noticias, este capítulo explica como configurar dos de los clásicos: **trn** y **nn**.

A nivel de grupo, la mayoría de los lectores presentan una lista de artículos en la que aparece el tema de los mismos y el autor. En los grupos grandes es difícil que el usuario caiga en la cuenta de los artículos relacionados entre sí, aunque es posible identificar las respuestas a un artículo anterior.

Una respuesta normalmente repite el título del artículo original precedido por `Re:`. Adicionalmente, el identificativo del mensaje al que se responde debería indicarse en la línea de cabecera `References:`. Ordenar los artículos utilizando esos dos criterios genera pequeños grupos (de hecho, son árboles) del artículo, los cuáles son llamados *hilos*. Una de las tareas al escribir un lector de noticias es diseñar un algoritmo eficiente para ordenar los artículos, ya que el tiempo requerido para ello es proporcional al cuadrado del número de artículos.

No discutiremos aquí cómo se construyen las interfaces de usuario. Actualmente todos los lectores disponibles para GNU/Linux tienen una buena función de ayuda; por favor, refiérase a ella para más detalle.

En las siguientes secciones, sólo trataremos tareas administrativas. La mayoría de ellas están relacionadas con la creación de bases de datos de hilos y contabilidad.

24.1. Configuración de tin

El lector más versatil en lo que al tratamiento de hilos se refiere es **tin**. Fue escrito por Iain Lea siguiendo el modelo de un lector anterior llamado **tass** (escrito por Rich Skrenta). Ordena los temas en el momento en que el usuario accede al grupo y es muy rápido haciéndolo excepto que se haga por NNTP.

En un 486DX50 se tarda unos 30 segundos en ordenar 1000 artículos, leyéndolos directamente del disco. Mediante NNTP con un servidor ocupado, rondaría los cinco minutos. [218] Se puede mejorar este tiempo actualizando regularmente los ficheros índice con la opción `u`, para que cuando usted vuelva a ejecutar **tin** para leer noticias los hilos ya existan. Como alternativa, para leer las noticias puede invocar **tin** con la opción `U`. Cuando lo invoque de esta manera, **tin** crea un nuevo proceso en segundo plano con el fin de construir los ficheros índice mientras usted está leyendo las noticias.

Normalmente **tin** guarda la información sobre las hilos en el directorio del usuario, bajo `.tin/index`. Sin embargo, esto puede ser costoso en términos de espacio en disco, así que usted debería mantener una copia única de ellas en un sitio centralizado. Esto se puede lograr haciendo que **tin** posea la propiedad setuid como news, por ejemplo. **tin** guardará la base de datos de los hilos en `/var/spool/news/.index`. Para cualquier acceso a fichero o secuencia de escape del intérprete de órdenes, reestablecerá su uid efectivo al uid real del usuario que lo invocó. [219]

La versión de **tin** incluida en algunas distribuciones de GNU/Linux no tiene soporte NNTP, pero la mayoría sí lo incorporan. Cuando se invoca como **rtin** o con la opción `r`, **tin** trata de conectar con el servidor NNTP especificado en el fichero `/etc/nntpserver` o en la variable de entorno NNTPSERVER. El fichero `nntpserver` simplemente contiene en una única línea el nombre del servidor.

24.2. Configuración de trn

trn es también el sucesor de un programa anterior, llamado **rn** (siglas de *read news*[220]) La t en su nombre significa threaded. [221]. Fue escrito por Wayne Davidson.

Al contrario que **tin**, **trn** no provee la generación de su base de datos de hilos en tiempo de ejecución. En cambio, usa las bases de datos creadas por un programa llamado **mthreads**, el cual debe ser ejecutado regularmente desde el **CRON** para actualizar los ficheros índice.

Aun así, se puede acceder a nuevos artículos aunque no esté ejecutándose **mthreads**, pero tendrá todos esos artículos sobre UNA OPORTUNIDAD DE INVERSIÓN GENUINA esparcidos por el menú de selección de artículos en vez de un solo hilo, el cuál puede saltarse fácilmente.

Para activar la ordenación en hilos de un grupo en particular, **mthreads** se invoca con la lista de grupos desde la línea de órdenes. El formato de la lista es el mismo que el del fichero `sys` de las C NEWS:

```
$ mthreads comp,rec,!rec.games.go
```

Esta instrucción permite ordenar en hilos todos los grupos comp y rec, excepto rec.games.go (la gente que juegue al Go no necesita hilos bonitos). Después de esto, simplemente se le invoca sin ninguna opción para que ordene todos los artículos que vayan llegando. El ordenamiento de todos los grupos del fichero `active` puede ser activado llamando al programa **mthreads** con una lista de grupos de all [222].

Si recibe las noticias durante la noche, bastaría con ejecutar **mthreads** una vez por la mañana, pero también puede más frecuentemente si es necesario. En sistemas con un tráfico muy denso, puede ser deseable ejecutar **mthreads** como tarea de fondo (modo demonio). Si se le llama al arrancar con la opción `d`, se pone como demonio, comprobando cada diez minutos si han llegado nuevos artículos, y ordenándolos si éste es el caso. Para ejecutar **mthreads** como tarea de fondo (modo demonio), ponga la siguiente línea en la macro `rc.news`:

```
/usr/local/bin/rn/mthreads -deav
```

La opción `a` hace que **mthreads** ordene automáticamente los nuevos grupos según se vayan creando. La opción `v` habilita los mensajes largos en el fichero de registro, llamado `mt.log` y situado en el directorio donde esté instalado **trn**.

Los ficheros antiguos que no estén disponibles en el sistema deben ser eliminados de los ficheros índice regularmente. Por omisión, sólo los artículos cuyo número esté por debajo de la marca de agua menor serán eliminados. [223] Los artículos que a pesar de estar por encima de este número hayan caducado (porque tengan el campo *`Expires:`* en la cabecera) pueden purgarse usando la opción `e` del programa **mthreads**. Cuando **mthreads** está ejecutándose en modo demonio, esta opción hace que use un modo de purga mejorado una vez al día, poco después de la media noche.

24.3. Configuración de nn

nn, escrito por Kim F. Storm, proclama ser un lector cuya última finalidad es no leer noticias. Su nombre significa No News,[224], y su lema es no news is good news. **nn** is better.[225]

Para alcanzar su ambiciosa meta, **nn** viene equipado con una gran cantidad de herramientas de mantenimiento que no sólo permiten la creación de hilos, sino también comprobaciones extensivas de la consistencia de tales bases de datos, contabilidad, recopilación de estadísticas, y restricciones de acceso. Existe también un programa de administración llamado **nnadmin**, que permite llevar a cabo estas tareas interactivamente. Es muy intuitivo, por lo que no profundizaremos en sus aspectos, sino que nos limitaremos a la creación de los ficheros índice.

El programa encargado de manejar las bases de datos para **nn** se llama **nnmaster**. Generalmente trabaja en modo demonio, el cual se inicia en el fichero `rc` en el proceso de arranque. Se le invoca de la siguiente manera:

```
/usr/local/lib/nn/nnmaster -l -r -C
```

Esto habilita la indexación para todos los grupos presentes en el fichero `active`.

De manera equivalente, se puede ejecutar **nnmaster** periódicamente desde **cron**,pasándole la lista de grupos sobre la que actuar. Esta lista es muy parecida a la lista de suscripciones del fichero `sys`, salvo que usa espacios en blanco en vez de comas. En vez del nombre **all**, se debe usar un argumento vacío de **""** para referirse a todos los grupos. Un ejemplo es:

```
# /usr/local/lib/nn/nnmaster !rec.games.go rec comp
```

Tenga en cuenta que el orden es significativo: la especificación de grupo que concuerde y esté más a la izquierda es la que vale. Por tanto, si se pone !rec.games.go después de rec, los artículos de este grupo se indexarían de todos modos.

nn ofrece varios métodos para borrar los artículos caducados de su base de datos. El primero es actualizar la base comprobando los directorios de los grupos, y desechando la entradas cuyo artículo correspondiente ya no esté disponible. Éste es el método predeterminado obtenido al invocar **nnmaster** con la opción `E`. Es razonablemente rápido, a menos que se haga vía NNTP.

El segundo método actúa exactamente como la opción predeterminada de **mthreads**; sólo elimina las entradas referidas a artículos cuyo número está por debajo de la línea de flotación en el fichero `active`. Puede ser habilitado con la opción `e`.

Finalmente, el tercer método consiste en desechar toda la base de datos y catalogar todos los artículos de nuevo. Esto puede hacerse pasándole la opción `E3` a **nnmaster**.

La lista de grupos sobre los que actuar se especifica mediante la opción `F`, del mismo modo que se describió anteriormente. Sin embargo, si **nnmaster** está ejecutándose como demonio, hay que matarlo (con la opción `k`) antes de proceder a purgar, y reiniciarlo después con las opciones originales. Por lo tanto, los comandos apropiados para purgar los índices de todos los grupos usando el primer método es:

```
# nnmaster -kF ""
# nnmaster -lrC
```

Hay muchas más opciones que pueden ser utilizadas para ajustar el comportamiento de **nn**. Si le interesa saber cómo eliminar artículos erróneos o agrupar los artículos en un compendio, lea la página de manual de **nnmaster**.

nnmaster se guía usando un fichero llamado `GROUPS`, situado en `/var/lib/nn`. Si no existe inicialmente, se crea. Para cada grupo, contiene una línea que comienza con el nombre del mismo, opcionalmente seguido de una anotación de tiempo y diversos indicadores. Es posible editar dichos indicadores para habilitar un determinado comportamiento para el grupo en cuestión, pero no se debe cambiar el orden en que aparecen los grupos [226] Los indicadores permitidos y sus efectos también vienen detallados en la página de manual de **nnmaster**.

Apéndice A. Red de ejemplo:La Cervecera Virtual

A lo largo de este libro hemos usado el siguiente ejemplo que es un poco menos complejo que la Universidad de Groucho Marx y puede ser más cercano a las tareas a las que realmente se enfrente.

La Cervecera Virtual es una pequeña compañía que elabora cerveza; como su propio nombre sugiere, cerveza Virtual. Para gestionar su negocio mas eficientemente, los cerveceros Virtuales quieren unir en red sus máquinas, que son PCs ejecutando el brillante y sobresaliente núcleo de producción de GNU/Linux. Figura A−1 muestra la configuración de red.

En el mismo piso, atravesando el salón principal, está la Vinatera Virtual, que opera íntimamente con la cervecera. Los viticultores tienen una red Ethernet de su propiedad. Naturalmente, las dos empresas quieren unir sus redes una vez que funcionen. Como primer paso, ellos quieren poner una máquina como puerta de enlace que reenvíe tráfico de datos entre las dos subredes. Luego quieren tener un enlace UUCP al mundo exterior, a través del cual poder intercambiar correo y noticias. Más adelante, quieren poner conexiones punto−a−punto (PPP) para conectar a sitios externos y a Internet.

La Cervecera Virtual y la Vinatera Virtual poseen una subred de clase C cada una, de la subred de clase B de la cervecera, y la puerta de enlace de cada una, a través de la máquina vlager, que también soporta la conexión UUCP. Figura A−2 muestra la configuración.

Figura A−1. Las subredes de la Cervecera Virtual y la Vinatera Virtual

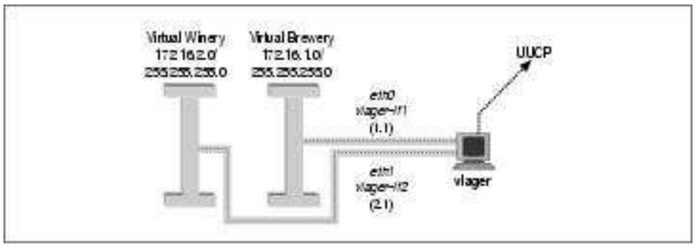
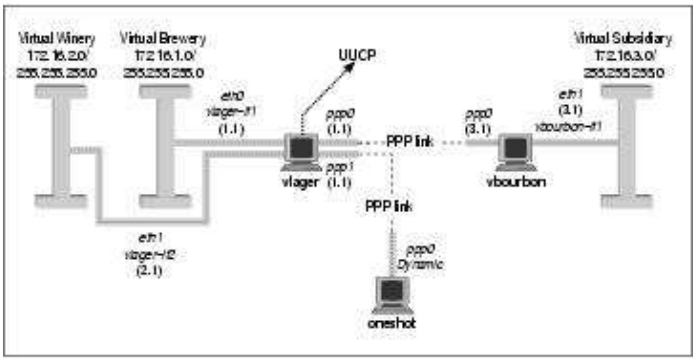


Figura A−2. La red de la cervecera



A.1. Conexión de la red Virtual subsidiaria

La Cervecera Virtual crece y abre otra tienda en otra ciudad. La franquicia dispone una ethernet propia usando la dirección IP 172.16.3.0, que es parte de la subred 3 de la red de clase B de la cervecera. La máquina vlager trabaja como la puerta de enlace para la red de la cervecera y soporta el enlace punto−a−punto (PPP); su igual en la nueva tienda se llama vbourbon y tiene la dirección IP 172.16.3.1. Esta red esta ilustrada en Figura A−2.

Apéndice B. Configuraciones de cableado útiles

Si desea conectar dos máquinas y no tiene una red ethernet, necesitará un cable de módem nulo o bien un cable paralelo PLIP.

Estos cables pueden comprarse, pero sale mucho más barato si lo construye usted mismo.

B.1. Un cable paralelo PLIP

En la construcción de un cable de impresora tipo nulo para usar en una conexión PLIP, se necesitarán dos conectores de 25 patillas (de los llamados DB−25) y un cable de 11 hilos. El cable no puede tener más de 15 metros de largo. El cable puede estar o no blindado, pero si estamos construyendo un cable largo, es recomendable que lo sea.

Si mira el conector, podrá ver pequeños números en la base de cada patilla, que van desde el 1 en la patilla superior izquierda (si coloca el lado más ancho arriba) hasta el 25 para la patilla de abajo a la derecha. Para tener un cable de impresora tipo nulo, se deberán conectar las siguientes patillas entre ambos conectores, como se muestra en Figura B−1.

Todas las patillas restantes quedarán desconectadas. Si el cable posee una malla externa, la misma se conectará a la carcasa metálica del conector DB−25 en *uno sólo* de los extremos.

B.2. Cable de Módem nulo de puerto serie

Un cable de módem nulo de puerto serie, funcionará tanto con SLIP como con PPP. De nuevo, necesitará dos conectores DB−25. Esta vez su cable necesita sólo ocho conductores.

Puede haber visto otros diseños de cables de módem nulo, pero éste le permite usar control de flujo por hardware, que es mucho mejor que el control de flujo XON/XOFF o a no tener control de flujo en absoluto. La configuración del conductor se muestra en Figura B−2:

De nuevo, si tienes un cable blindado, debería conectarlo a la primera patilla.

Figura B−1. Cable paralelo PLIP

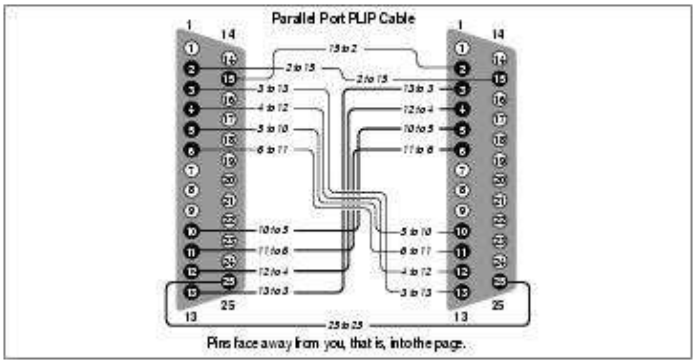
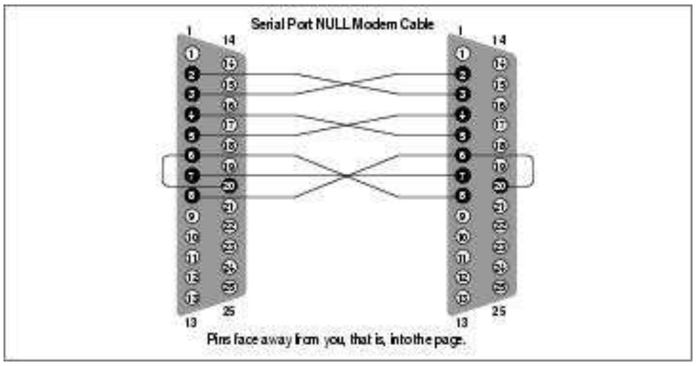


Figura B−2. Cable de módem nulo de puerto serie


Serial Port NULL Modem Cable
Pins face away from you, that is, into the page.

Apéndice C. Linux Network Administrator's Guide, Second Edition Copyright Information

C.1. 0. Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft," which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

C.2. 1. Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document," below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you."

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front−matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front−Cover Texts or Back−Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine−readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque."

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard−conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine−generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats that do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

C.3. 2. Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

C.4. 3. Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front−Cover Texts on the front cover, and Back−Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine−readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly−accessible computer−network location containing a complete Transparent copy of the Document, free of added material, which the general network−using public has access to download anonymously at no charge using public−standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

C.5. 4. Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A.
   Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B.
   List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C.
   State on the Title page the name of the publisher of the Modified Version, as the publisher.

D.
   Preserve all the copyright notices of the Document.

E.
   Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F.

Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G.
Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H.
Include an unaltered copy of this License.

I.
Preserve the section entitled "History," and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J.
Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K.
In any section entitled "Acknowledgements" or "Dedications," preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L.
Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M.
Delete any section entitled "Endorsements." Such a section may not be included in the Modified Version.

N.
Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front−matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements," provided it contains nothing but endorsements of your Modified Version by various partiesfor example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front−Cover Text, and a passage of up to 25 words as a Back−Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front−Cover Text and one of Back−Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

C.6. 5. Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements," and any sections entitled "Dedications." You must delete all sections entitled "Endorsements."

C.7. 6. Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

C.8. 7. Aggregation with Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate," and this License does not apply to the other self–contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

C.9. 8. Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

C.10. 9. Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

C.11. 10. Future Revisions of this License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Apéndice D. Guía de Administración de Redes con GNU/Linux, Segunda Edición Información de Copyright

Versión 1.1, Marzo de 2000

Esta es la GNU Free Document License (GFDL), versión 1.1 (de marzo de 2.000), que cubre manuales y documentación para el software de la Free Software Foundation, con posibilidades en otros campos. La traducción[227] no tiene ningún valor legal, ni ha sido comprobada de acuerdo a la legislación de ningún país en particular. Vea el original

Los autores de esta traducción son:

- Igor Támara <`ikks@bigfoot.com`>
- Pablo Reyes <`reyes_pablo@hotmail.com`>
- Revisión : Vladimir Támara P. <`vtamara@gnu.org`>

Boston, MA 02111−1307 USA

Se permite la copia y distribución de copias literales de este documento de licencia, pero no se permiten cambios.

0. Preámbulo

El propósito de esta licencia es permitir que un manual, libro de texto, u otro documento escrito sea "libre" en el sentido de libertad: asegurar a todo el mundo la libertad efectiva de copiarlo y redistribuirlo, con o sin modificaciones, de manera comercial o no. En segundo término, esta licencia preserva para el autor o para quien publica una manera de obtener reconocimiento por su trabajo, al tiempo que no se consideran responsables de las modificaciones realizadas por terceros.

Esta licencia es una especie de "copyleft" que significa que los trabajos derivados del documento deben a su vez ser libres en el mismo sentido. Esto complementa la Licencia Pública General GNU, que es una licencia de copyleft diseñada para el software libre.

Hemos diseñado esta Licencia para usarla en manuales de software libre, ya que el software libre necesita documentación libre: Un programa libre debe venir con los manuales que ofrezcan la mismas libertades que da el software. Pero esta licencia no se limita a manuales de software; puede ser usada para cualquier trabajo textual, sin tener en cuenta su temática o si se publica como libro impreso. Recomendamos esta licencia principalmente para trabajos cuyo fin sea instructivo o de referencia.

1. Aplicabilidad y definiciones

Esta Licencia se aplica a cualquier manual u otro documento que contenga una nota del propietario de los derechos que indique que puede ser distribuido bajo los términos de la Licencia. El "Documento", en adelante, se refiere a cualquiera de dichos manuales o trabajos. Cualquier miembro del público es un licenciatario, y será denominado como "Usted".

Una "Versión Modificada" del Documento significa cualquier trabajo que contenga el Documento o una porción del mismo, ya sea una copia literal o con modificaciones y/o traducciones a otro idioma.

Una "Sección Secundaria" es un apéndice titulado o una sección preliminar al prólogo del Documento que tiene que ver exclusivamente con la relación de quien publica o, los autores del Documento o, el tema general del Documento(o asuntos relacionados) y cuyo contenido no entra directamente en este tema general. (Por ejemplo, si el Documento es en parte un texto de matemáticas, una Sección Secundaria puede no explicar matemáticas.) La relación puede ser un asunto de conexión histórica, o de posición legal, comercial, filosófica, ética o política con el tema o la materia del texto.

Las "Secciones Invariantes" son ciertas Secciones Secundarias cuyos títulos son denominados como Secciones Invariantes, en la nota que indica que el documento es liberado bajo esta licencia.

Los "Textos de Cubierta" son ciertos pasajes cortos de texto que se listan, como Textos de Portada o Textos de Contra Portada, en la nota que indica que el documento es liberado bajo esta Licencia.

Una copia "Transparente" del Documento, significa una copia para lectura en máquina, representada en un formato cuya especificación está disponible al público general, cuyos contenidos pueden ser vistos y editados directamente con editores de texto genéricos o (para imágenees compuestas por pixeles) de programas genéricos de dibujo o (para dibujos) algún editor gráfico ampliamente disponible, y que sea adecuado para exportar a formateadores de texto o para traducción automática a una variedad de formatos adecuados para ingresar a formateadores de texto. Una copia hecha en un formato de un archivo que no sea Transparente, cuyo formato ha sido diseñado para impedir o dificultar subsecuentes modificaciones posteriores por parte de los lectores no es Transparente. Una copia que no es "Transparente" es llamada "Opaca".

Como ejemplos de formatos adecuados para copias Transparentes están el ASCII plano sin formato, formato de Texinfo, formato de LaTeX, SGML o XML usando un DTD disponible ampliamente, y HTML simple que sigue los estándares, diseñado para modificaciones humanas. Los formatos Opacos incluyen PostScript, PDF, formatos propietarios que pueden ser leídos y editados unicamente en procesadores de palabras propietarios, SGML o XML para los cuáles los DTD y/o herramientas de procesamiento no están disponibles generalmente, y el HTML generado por máquinas producto de algún procesador de palabras solo para propósitos de salida.

La "Portada" en un libro impreso significa, la portada misma, más las páginas siguientes necesarias para mantener la legibilidad del material, que esta Licencia requiere que aparezca en la portada. Para trabajos en formatos que no tienen Portada como tal, "Portada" significa el texto cerca a la aparición más prominente del título del trabajo, precediendo el comienzo del cuerpo del trabajo.

2. Copia literal

Puede copiar y distribuir el Documento en cualquier medio, sea en forma comercial o no, siempre y cuando esta Licencia, las notas de derecho de autor, y la nota de licencia que indica que esta Licencia se aplica al Documento se reproduzca en todas las copias, y que usted no adicione ninguna otra condición a las expuestas en en esta Licencia. No puede usar medidas técnicas para obstruir o controlar la lectura o copia posterior de las copias que usted haga o distribuya. Sin embargo, usted puede aceptar compensación a cambio de las copias. Si distribuye un número suficientemente grande de copias también deberá seguir las condiciones de la sección 3.

También puede prestar copias, bajo las mismas condiciones establecidas anteriormente, y puede exhibir copias publicamente.

3. Copiado en cantidades

Si publica copias impresas del Documento que sobrepasen las 100, y la nota de Licencia del Documento exige Textos de Cubierta, debe incluir las copias con cubiertas que lleven en forma clara y legible, todos esos textos de Cubierta: Textos Frontales en la cubierta frontal, y Textos Posteriores de Cubierta en la Cubierta Posterior. Ambas cubiertas deben identificarlo a Usted clara y legiblemente como quien publica tales copias. La Cubierta Frontal debe mostrar el título completo con todas las palabras igualmente prominentes y visibles. Además puede adicionar otro material en la cubierta. Las copias con cambios limitados en las cubiertas, siempre que preserven el título del Documento y satisfagan estas condiciones, puede considerarse como copia literal.

Si los textos requeridos para la cubierta son muy voluminosos para que ajusten legiblemente, debe colocar los primeros (tantos como sea razonable colocar) en la cubierta real, y continuar el resto en páginas adyacentes.

Si publica o distribuye copias Opacas del Documento cuya cantidad exceda las 100, debe incluir una copia Transparente que pueda ser leída por una máquina con cada copia Opaca, o entregar en o con cada copia Opaca una dirección en red de computador publicamente−accesible conteniendo una copia completa Transparente del Documento, sin material adicional, a la cual el público en general de la red pueda acceder a bajar anónimamente sin cargo usando protocolos de standard público. Si usted hace uso de la última opción, deberá tomar medidas necesarias, cuando comience la distribución de las copias Opacas en cantidad, para asegurar que esta copia Transparente permanecerá accesible en el sitio por lo menos un año después de su última distribución de copias Opacas (directamente o a través de sus agentes o distribuidores) de esa edición al público.

Se solicita, aunque no es requisito, que contacte a los autores del Documento antes de redistribuir cualquier gran número de copias, para permitirle la oportunidad de que le provean una versión del Documento.

4. Moodificaciones

Puede copiar y distribuir una Versión Modificada del Documento bajo las condiciones de las seccions 2 y 3 anteriores, siempre que usted libere la Versión Modificada bajo esta misma Licencia, con la Versión Modificada haciendo el rol del Documento, por lo tanto licenciando la distribución y modificación de la Versión Modificada a quienquiera que posea una copia de este. En adición, debe hacer lo siguiente en la Versión Modificada:

- **A.** Uso en la Portada (y en las cubiertas, si hay alguna) de un título distinto al del Documento, y de versiones anteriores (que deberían, si hay alguna, estar listados en la sección de Historia del Documento). Puede usar el mismo título que versiones anteriores al original siempre que quién publicó la primera versión lo permita.
- **B.** Listar en la Portada, como autores, una o más personas o entidades responsables por la autoría o las modificaciones en la Versión Modificada, junto con por lo menos cinco de los autores principales del Documento (Todos sus autores principales, si hay menos de cinco).
- **C.** Estado en la Portada del nombre de quién publica la Versión Modificada, como quien publica.
- **D.** Preservar todas las notas de derechos de autor del Documento.
- **E.** Adicionar una nota de derecho de autor apropiada a sus modificaciones adyacentes a las otras notas de derecho de autor.
- **F.** Incluir, immediatamente después de la nota de derecho de autor, una nota de licencia dando el permiso público para usar la Versión Modificada bajo los términos de esta Licencia, de la forma mostrada en la Adición (LEGAL)abajo.
- **G.** Preservar en esa nota de licencia el listado completo de Secciones Invariantes y en los Textos de las Cubiertas que sean requeridos como se especifique en la nota de Licencia del Documento
- **H.** Incluir una copia sin modificación de esta Licencia.
- **I.** Preservar la sección llamada "Historia", y su título, y adicionar a esta una sección estableciendo al menos el título, el año,los nuevos autores, y quién publicó la Versión Modificada como reza en la Portada. Si no hay una sección titulada "Historia" en el Documento, crear una estableciendo el título, el año, los autores y quien publicó el Documento como reza en la Portada, añadiendo además un artículo describiendo la Versión Modificada como se estableció en el punto anterior.
- **J.** Preservar la localización en red, si hay , dada en la Documentación para acceder públicamente a una copia Transparente del Documento, tanto como las otras direcciones de red dadas en el Documento para versiones anteriores en las cuáles estuviese basado. Estas pueden ubicarse en la sección "Historia". Se puede omitir la ubicación en red para un trabajo que sea publicado por lo menos 4 años antes que el mismo Documento, o si quien publica originalmente la versión da permiso explícitamente.
- **K.** En cualquier sección titulada "Agradecimientos" o "Dedicatorias", preservar el título de la sección, y preservar en la sección toda la sustancia y el tono de los agradeimientos y/o dedicatorias de cada contribuyente que estén incluídas.
- **L.** Preservar todas las Secciones Invariantes del Documento, sin alterar su texto ni sus títulos. Números de sección o el equivalente no son considerados parte de los títulos de la sección. M. Borrar cualquier sección titulada "Aprobaciones". Tales secciones no pueden estar incluidas en las Versiones Modificadas.
- **M.** Borrar cualquier sección titulada "Aprobaciones". Tales secciones no pueden estar incluidas en las Versiones Modificadas.
- **N.** No retitular ninguna sección existente como "Aprobaciones" o conflictuar con título con alguna Sección Invariante.

Si la Versión Modificada incluye secciones o apendices nuevos o preliminares al prólogo que califican como Secciones Secundarias y contienen material no copiado del Documento, puede opcionalmente designar algunas o todas esas secciones como invariantes. Para hacerlo, adicione sus títulos a la lista de Secciones Invariantes en la nota de licencia de la Versión Modificada. Tales títulos deben ser distintos de cualquier otro título de sección.

Puede adicionar una sección titulada "Aprobaciones", siempre que contenga unicamente aprobaciones de su Versión Modificada por varias fuentes−−por ejemplo, observaciones de peritos o que el texto ha sido aprobado por una organización como un standard.

Puede adicionar un pasaje de hasta cinco palabras como un Texto de Cubierta Frontal, y un pasaje de hasta 25 palabras como un texto de Cubierta Posterior, al final de la lista de Textos de Cubierta en la Versión Modificada. Solamente un pasaje de Texto de Cubierta Frontal y un Texto de Cubierta Posterior puede ser adicionado por (o a manera de arreglos hechos por) una entidad. Si el Documento ya incluye un texto de cubierta para la misma cubierta, previamente adicionado por usted o por arreglo hecho por la misma entidad, a nombre de la cual está actuando, no puede adicionar otra; pero puede reemplazar la anterior, con permiso explícito de quien publicó anteriormente tal cubierta.

El(los) autor(es) y quien(es) publica(n) el Documento no dan con esta Licencia permiso para usar sus nombres para publicidad o para asegurar o implicar aprobación de cualquier Versión Modificada.

5. Combinando documentos

Puede combinar el Documento con otros documentos liberados bajo esta Licencia, bajo los términos definidos en la sección 4 anterior para versiones modificadas, siempre que incluya en la combinación todas las Secciones Invariantes de todos los documentos originales, sin modificar, y listadas todas como Secciones Invariantes del trabajo combinado en su nota de licencia.

El trabajo combinado necesita contener solamente una copia de esta Licencia, y múltiples Seccions Invariantes Idénticas pueden ser reemplazadas por una sola copia. Si hay múltiples Secciones Invariantes con el mismo nombre pero con contenidos diferentes, haga el título de cada una de estas secciones único adicionándole al final de este, en paréntesis, el nombre del autor o de quien publicó originalmente esa sección, si es conocido, o si no, un número único. Haga el mismo ajuste a los títulos de sección en la lista de Secciones Invariantes en la nota de licencia del trabajo combinado.

En la combinación, debe combinar cualquier sección titulada "Historia" de los varios documentos originales, formando una sección titulada "Historia"; de la misma forma combine cualquier seción titulada "Agradecimientos", y cualquier sección titulada "Dedicatorias". Debe borrar todas las secciones tituladas "Aprobaciones."

6. Colecciones de documentos

Puede hacer una colección consistente del Documento y otros documentos liberados bajo esta Licencia, y reemplazar las copias individuales de esta Licencia en los varios documentos con una sola copia que esté incluida en la colección, siempre que siga las reglas de esta Licencia para una copia literal de cada uno de los documentos en cualquiera de todos los aspectos.

Puede extraer un solo documento de una de tales colecciones, y distribuirlo individualmente bajo esta Licencia, siempre que inserte una copia de esta Licencia en el documento extraido, y siga esta Licencia en todos los otros aspectos concernientes a la copia literal de tal documento.

7. Agregación con trabajos independientes

Una recopilación del Documento o de sus derivados con otros documentos o trabajos separados o independientes, en cualquier tipo de distribución o medio de almacenamiento, no como un todo, cuenta como una Versión Modificada del Documento, teniendo en cuenta que ninguna compilación de derechos de autor sea clamada por la recopilación. Tal recopilación es llamada un "agregado", y esta Licencia no aplica a los otros trabajos auto−contenidos y por lo tanto compilados con el Documento, o a cuenta de haber sido compilados, si no son ellos mismos trabajos derivados del Documento.

Si el requerimiento de la sección 3 del Texto de la Cubierta es aplicable a estas copias del Documento, entonces si el Documento es menor que un cuarto del agregado entero, Los Textos de la Cubierta del Documento pueden ser colocados en cubiertas que enmarquen solamente el Documento entre el agregado. De otra forma deben aparecer en cubiertas enmarcando todo el agregado.

8. Traducción

La Traducción es considerada como una clase de modificación, Así que puede distribuir traducciones del Documento bajo los términos de la sección 4. Reemplazar las Secciones Invariantes con traducciones requiere permiso especial de los dueños de derecho de autor, pero puede incluir traducciones de algunas o todas las Secciones Invariantes adicionalmente a las versiones originales de las Secciones Invariantes. Puede incluir una traducción de esta Licencia siempre que incluya también la versión Inglesa de esta Licencia. En caso de un desacuerdo entre la traducción y la versión original en Inglés de esta Licencia, la versión original en Inglés prevalecerá.

9. Terminación

No se puede copiar, modificar, sublicenciar, o distribuir el Documento excepto por lo permitido expresamente bajo esta Licencia. Cualquier otro intento de copia, modificación, sublicenciamiento o distribución del Documento es nulo, y serán automáticamente terminados sus derechos bajo esa licencia. De todas maneras, los terceros que hayan recibido copias, o derechos, de su parte bajo esta Licencia no tendrán por terminadas sus licencias siempre que tales personas o entidades se encuentren en total conformidad con la licencia original.

10. Futuras revisiones de esta licencia

La Free Software Foundation puede publicar nuevas, revisadas versiones de la Licencia de Documentación Libre GNU de tiempo en tiempo. Tales nuevas versiones serán similares en espíritu a la presente versión, pero pueden diferir en detalles para solucionar problemas o intereses. Vea http://www.gnu.org/copyleft/.

Cada versión de la Licencia tiene un número de versión que la distingue. Si el Documento especifica que una versión numerada particularmente de esta licencia o "cualquier versión posterior" se aplica a esta, tiene la opción de seguir los términos y condiciones de la versión especificada o cualquiera posterior que ha sido publicada(no como un borrador)por la Free Software Foundation. Si el Documento no especifica un número de versión de esta Licencia, puede escoger cualquier versión que haya sido publicada(no como un borrador) por la Free Software Foundation.

Addendum

Para usar esta licencia en un documento que usted haya escrito, incluya una copia de la Licencia en el documento y ponga el siguiente derecho de autor y nota de licencia justo después del título de la página:

Derecho de Autor © Año Su Nombre.

Permiso para copiar, distribuir y/o modificar este documento bajo los términos de la Licencia de Documentación Libre GNU, Versión 1.1 o cualquier otra versión posterior publicada por la Free Software Foundation; con las Secciones Invariantes siendo LISTE SUS TÍTULOS, con los siendo LISTELO el texto de la Cubierta Frontal, y siendo LISTELO el texto de la Cubierta Posterior. Una copia de la licencia es incluida en la sección titulada "Licencia de Documentación Libre GNU".

Si no tiene Secciones Invariantes, escriba "Sin Secciones Invariantes" en vez de decir cuáles son invariantes. Si no tiene Texto de Cubierta Frontal, escriba "Sin Texto de Cubierta Frontal" en vez de"siendo LISTELO el texto de la Cubierta Frontal"; Así como para la Cubierta Posterior.

Si su documento contiene ejemplos de código de programa no triviales, recomendamos liberar estos ejemplos en paralelo bajo su elección de licencia de software libre, tal como la Licencia de Público General GNU, para permitir su uso en software libre.

Apéndice E. SAGE: El Gremio del Administrador

Si nuestras necesidades no son cubiertas mediante los grupos de noticias comp.os.linux.* así como la documentación, posiblemente sea el momento de pensar en unirse al SAGE, el Gremio de los Administradores de Sistemas (System Administrators Guild). El objetivo del SAGE es avanzar en la administración de sistemas como una profesión. El SAGE intenta acercar a los administradores de sistemas y redes a otros profesionales, para compartir problemas y soluciones, comunicarse con los usuarios, gestores y comerciales en lo referente a los sistemas.

Actualmente, las iniciativas de SAGE son:

- Co−patrocinar las exitosas Conferencias Anuales de Administración de Sistemas (LISA) con USENIX.
- Publicar la revista **Job Descriptions for System Administrators**, editada por Tina Darmohray, la primera de una serie de prácticas guías que cubren asuntos diversos de la administración de sistemas.
- Creación de un servidor de ficheros, ftp.sage.usenix.org, para los artículos de las Conferencias de Administración de Sistemas y documentación relacionada con este mundo.
- Establecer grupos de trabajo en áreas de importancia para los administradores de sistemas, como trabajos, publicaciones, políticas, distribución electrónica de información, educación, comerciales y estándares.

Para saber más acerca de la Asociación USENIX y su Grupo Especial Técnico, SAGE, contacte con la oficina USENIX en el teléfono (510) 528−8649 de los Estados Unidos, o por correo electrónico a office@usenix.org. Para recibir información por vía electrónica, contacte con info@usenix.org. La cuota anual para pertenecer al SAGE es de 25 dólares americanos. Además hay que ser miembro de USENIX. Los socios reciben gratuitamente las revistas *login:* y **Computing Systems**; descuentos en conferencias y simposios; y ahorros importantes en compras en SAGE y otros servicios.

Índice

Notas

[1] Disponible en http://lucas.hispalinux.es/

[2] http://www.insflug.org/

[3] N. del T.: Sistema de Charlas en Internet

[4] http://www.hispalinux.es/

[5] A veces nos costará esperar menos bajándonos una distribución a lo largo de 24 horas que esperar las 72 horas que suelen tardar en enviarnos un CD−ROM por correo.

[6] Terry Dawson está localizable en terry@linux.org.au.

[7] Philip Hazel puede ser localizado en la dirección ph10@cus.cam.ac.uk.

[8] Observe que mientras usted puede imprimir la versión en línea, Ud. *no* puede fotocopiar el libro de O'Reilly, y mucho menos vender ninguna de esas (hipotéticas) copias.

[9] http://lucas.hispalinux.es/

[10] Costumbre que todavía se usa algunas ocasiones en Europa (ver arriba).

[11] Una shell es un intérprete de órdenes, que actúa como interfaz con el sistema operativo Unix. Se le puede comparar al COMMAND.COM de DOS en un entorno de Microsoft Windows, aunque mucho más potente.

[12] Respectivas traducciones de repeaters, bridges y routers. Nota del T.

[13] El FAQ de Ethernet que se encuentra en http://www.faqs.org/faqs/LANs/ethernet−faq/ habla sobre este tema. También se puede encontrar abundante información histórica y técnica, muy detallada en el web de Charles Spurgeon's dedicado a Ethernet, http://wwwhost.ots.utexas.edu/ethernet/.

[14] En el original: Frame Relay Access Device. Nota del T.

[15] En el original: Asynchronous Transfer Mode. Nota del T.

[16] Terminal Node Controller, en el original. Nota del T.

[17] Definiciones que en el inglés serían dotted decimal notation y dotted quad notation, respectivamente. Nota del T.

[18] Berkeley Socket Library. Nota del T.

[19] En donde las traducciones al español de los términos bind, connect, listen y accept son enlazar, conectar, escuchar y aceptar respectivamente

[20] Amateur Radio protocol. Nota del T.

[21] Se puede contactar a Alan, en alan@lxorguk.ukuu.org.uk

[22] Cortafuegos IP, Auditoría IP, Enmascaramiento IP y Encapsulación IP, son las respectivas traducciones de los términos IP Firewalling, IP accounting, IP Masquerade, y IP Tunnelling

[23] NCP es el protocolo en el cuál se basa el servicio de ficheros e impresión de Novell.

[24] Y hasta en distintos idiomas. Nota del T.

[25] Se volverá a retomar este tema en Capítulo 12.

[26] Ha habido sistemas UNIX comerciales, (por los que hay que pagar un montón de dinero), que venían con un script de shell setuid−root que permitía a los usuarios obtener privilegios de root utilizando un simple y conocido truco.

[27] En 1988, el gusano RTM llevó a gran parte de Internet a un colapso, en parte por explotar un agujero que había en algunos programas, incluyendo a **sendmail**. Este agujero ya ha sido reparado con creces.

[28] del que ya hay disponible una traducción en tldp−es

[29] La versión del Protocolo de Internet usada más frecuentemente en Internet es la Versión 4. Se ha hecho un gran esfuerzo para diseñar una versión de reemplazo llamada IP Versión 6. IPv6 tiene un esquema de direcciones distinto y direcciones más largas. GNU/Linux tiene un implemento de IPv6, pero no está preparado para ser documentado en este libro aún. El soporte del núcleo de GNU/Linux para IPv6 es bueno, pero un gran número de aplicaciones de red necesitan ser modificadas para soportarlo también. Manténgase informado.

[30] Frecuentemente, las direcciones IP le son asignadas por el proveedor al que compró su conexión IP. De todos modos, también puede solicitar una dirección IP directamente al NIC enviando un email a hostmaster@internic.net, o usando el formulario en http://www.internic.net/.

[31] Los sistemas autónomos son ligeramente más generales. En ellos se puede comprender más de una red IP.

[32] El valor de una ruta puede imaginarse, en casos simples, como el número de saltos requeridos para alcanzar el destino. Aunque calcular apropiadamente los valores de las rutas en diseños de redes complejas puede ser una gran dificultad.

[33] N. del T.: Puerto Inalcanzable.

[34] N. del T.: Redirección.

[35] N. del T.: Del inglés Interrupt ReQuest

[36] Las IRQs 2 y 9 son las mismas porque el diseño del IBM PC tiene 2 procesadores de interrupciones en cascada con 8 IRQs cada uno, el procesador secundario es conectado a la IRQ 2 del primario.

[37] La gente suele usar núcleos en desarrollo e informar de los fallos que encuentra, hacer esto es algo muy útil si tiene una máquina que pueda usar como máquina de pruebas. Las instrucciones de cómo informar de los fallos están detalladas en el fichero `/usr/src/linux/REPORTING−BUGS` del código fuente del núcleo Linux.

[38] Recuerde, el protocolo IP puede ser transportado sobre multitud de tipos diferentes de redes, y no todas las redes podrán soportar tamaños de paquetes tan largos como los de las Ethernet.

[39] en el Ethernet HOWTO, Paul puede ser localizado en gpg109@rsphy1.anu.edu.au

[40] ¡Luche para limpiar el nombre del haking! Use siempre cracker cuando se refiera a gente que intenta conscientemente saltarse la seguridad de un sistema, y hacker cuando se refiera a gente que encuentra una forma inteligente de solucionar un problema. Los hackers pueden ser crackers, pero nunca se debe confundirlos. Consulte el nuevo diccionario de hackers (popularmente llamado the Jargon file) para

entender mejor el significado de los términos.

[41] El parche del adaptador de puerto paralelo mejorado para el núcleo 2.0 está disponible en http://www.cyberelk.demon.co.uk/parport.html.

[42] El NCSA **telnet** es un programa popular para DOS que corre TCP/IP sobre Ethernet o PLIP, y soporta **telnet** y FTP.

[43] Niibe puede ser localizado en gniibe@mri.co.jp.

[44] Puede contactar con Philip en *Philip.Blundell@pobox.com*.

[45] Es posible contactar con David en la dirección de correo electrónico bf347@lafn.org.

[46] ¡Nótese que no estamos hablando sobre WinModem aquí! Los WinModems tienen una circuitería muy simple y dependen por completo de la CPU principal de su ordenador, en lugar de hardware específico que haga todo el trabajo duro. Si ha decidido comprar un módem, le recomendamos encarecidamente que *no* compre un módem de ese tipo; consiga un módem de verdad. Quizá encuentre soporte para WinModems en GNU/Linux, pero eso sólo los convierte en una solución marginalmente más atractiva.

[47] El primer número de cada subred es la dirección de la subred, y el último se reserva para las direcciones de difusión (broadcast), luego tenemos un total de 62 máquinas por subred.

[48] Sólo necesita la dirección del servidor NIS si usa el NYS de Peter Eriksson. Otras implementaciones de NIS sólo encuentran sus servidores en ejecución usando **ypbind**.

[49] Dése cuenta de que los nombres en `networks` no pueden coincidir con nombres de máquinas del fichero `hosts` o algunos programas pueden producir extraños resultados.

[50] ¿Alguien recuerdaEchoes de Pink Floyd?

[51] Por ejemplo, todas las aplicaciones basadas en RPC utilizan la interfaz de bucle local para registrarse en el demonio **portmapper**(mapa de puertos) durante el arranque. Entre estas aplicaciones están NIS y NFS.

[52] Dése cuenta de que **`pointopoint`** no es una errata, es así como se escribe.

[53] Simplemente por precaución, debería configurar de todos modos sus enlaces PLIP o SLIP una vez que ha completado la configuración de la tabla de encaminamiento de las Ethernets. Con algunos núcleos más antiguos, la tabla de encaminamiento para la red puede acabar apuntando a su enlace punto–a–punto.

[54] La interfaz comodín se llama `dummy0` si usted lo tiene cargado como un módulo en lugar de una opción integrada en el núcleo. Esto se debe a que es posible cargar varios módulos y tener más de una interfaz comodín.

[55] Más correctamente, el uso de alias de IP se conoce como hosting virtual de capa de red. En los mundos del WWW y STMP es más común usar hosting virtual de capa de aplicación, en el que la misma dirección IP es usada por cada máquina virtual, pero en cada petición de la capa de aplicación se pasa un nombre de máquina diferente. Servicios como el FTP no son capaces de operar de esta forma, y necesitan hosting virtual de capa de red.

[56] RIP elige la ruta óptima a una determinada máquina basándose en la longitud del camino. Esto se calcula sumando los valores de las métricas individuales de cada enlace máquina–a–máquina. por omisión, un salto tiene longitud 1, pero puede ser cualquier entero positivo menor de 16 (Una longitud de ruta igual a 16 equivale a infinito. Estas rutas se consideran inusables). El parámetro `metric` establece este coste de los saltos, que es difundido entonces por el demonio de encaminamiento.

[57] ssh puede obtenerse en ftp.cs.hut.fi en `/pub/ssh`.

[58] Para saber si una conexión es saliente por los números de puerto. El número de puerto mostrado por una máquina *que llama* siempre será un entero simple. En el caso de llamar a una máquina, usaremos un puerto correspondiente a un servicio conocido, por lo que **netstat** usará el nombre simbólico, como `smtp`, que encuentre en `/etc/services`.

[59] N. del T.: literalmente, burla

[60] N. del T.: Ya han sido aprobados algunos, cuya elección no ha estado exenta de polémica.

[61] Si la información no se almacenara en caché, el sistema sería realmente ineficiente.

[62] En todo caso debe servir el dominio localhost y resolución inversa de 127.0.0.1.

[63] BIND 4.9 fue desarrollado por Paul Vixie, paul@vix.com, aunque actualmente lo mantiene el Consorcio de Software para Internet, en bind−bugs@isc.org.

[64] N. del T.: Time to Live

[65] Nótese que no podemos preguntar esto a nuestro servidor de nombres si no tenemos entradas de servidores raíz instaladas. Para evitar esto, podemos usar **nslookup** para interrogar a un servidor de nombres externo, o usar como fichero de caché el mostrado en Ejemplo 6−10 y de ahí obtener una lista completa.

[66] **diplogin** debe activarse con el bit setuid como root. Examine la sección al final de este capítulo.

[67] Los RFCs más relevantes están indicados en la bibliografia al final del libro.

[68] En realidad, HDLC es un protocolo mucho más general publicado por la Organización Internacional de Estándares (ISO).

[69] Si usted tiene alguna duda genérica sobre PPP, pregunte a gente de la lista de correo de Linux−net en vger.rutgers.edu.

[70] Se puede contactar con Karl en karl@morningstar.com.

[71] El encaminamineto predeterminado es instalado solamente si no hay ninguno establecido previamente.

[72] Si edita el `syslog.conf` para redirigir estos mensajes a un fichero, asegúrese de que este fichero no pueda ser leído por cualquiera, pues **chat** también captura todo el script de entrada por omisión − incluyendo las contraseñas.

[73] Más información sobre mecanismos de asignación dinámica a nodos puede encontrarse aquí: http://www.dynip.com/ y http://www.justlinux.com/dynamic_dns.html.

[74] Usar nombres de dominio en esta opción tiene consecuencias en la autentificación CHAP. Por favor, consulte la sección Sección 8.8 más adelante en este mismo capítulo.

[75] Las opciones `ipcp-accept-local` y `ipcp-accept-remote` indican a **pppd** aceptar la dirección local y remota ofrecidas por el PPP remoto, incluso si usted a provisto de alguna en su configuración. Si estas opciones no son configuradas, su **pppd** rechazará cualquier intento de negociación de las direcciones IP usadas.

[76] Si quisieramos tener rutas creadas para otros sitios cuando se conecten, tendriamos que añadir entradas apropiadas para permitir a aquellos `...` que aparecieran en el ejemplo.

[77] secrets es sólo el nombre que da PPP a las contraseñas. Las contraseñas de PPP no tienen las mismas limitaciones de tamaño que las contraseñas de registro de Linux.

[78] Las comillas no son parte de la contraseña, simplemente sirven para proteger el espacio en blanco del interior de la contraseña.

[79] Este nombre de ordenador se toma del reto de CHAP.

[80] La utilidades **useradd** o **adduser**, si las tiene, simplificaran la tarea.

[81] N. del T.: 'firewalls' en el original en inglés

[82] N. del T.: 'spoofing' en el original en inglés

[83] N. del T.: "método de aprovechamiento de una debilidad o vulnerabilidad"

[84] N. del T.: "fisgoneo"

[85] El término *cortafuegos* (N. del T.: 'firewall' en inglés) proviene de un aparato utilizado para proteger a las personas del fuego. El cortafuegos es un escudo de material resistente al fuego que se coloca entre un fuego potencial y las personas que protege.

[86] N. del T.: 'perimeter network' en el original en inglés

[87] N. del T.: 'chokes' en el original en inglés

[88] N. del T.: 'proxy' en el original en inglés

[89] N. del T.: 'socket' en el original en inglés

[90] N. del T.: existe una traducción oficial al castellano con el nombre de "Cortafuegos−Como".

[91] 'firewall packet logging' (N. del T.: "registro de paquetes del cortafuegos" en español) es una característica especial que permite escribir una línea de información en un dispositivo especial y visible

para usted por cada datagrama que coincida con un regla del cortafuegos.

[92] N. del T.: 'script' en el original en inglés

[93] N. del T.: "entrada"

[94] N. del T.: "reenvío"

[95] N. del T.: "salida"

[96] N. del T.: "denegación"

[97] El modo activo de FTP se habilita, de forma poco intuitiva, con la orden **PORT**. El modo pasivo de FTP se habilita con la orden **PASV**.

[98] El demonio ProFTPd constituye un buen ejemplo de un servidor de FTP que no procede así, al menos,en sus versiones antiguas.

[99] N.del T.: se han utilizado las descripciones de la traducción al español por P.J. Ponce de León, dentro del proyecto RFC−ES, del RFC0792 "Protocolo de mensajes de control de internet"

[100] N. del T.: "cadenas de IP"

[101] Puede contactar con Paul en Paul.Russell@rustcorp.com.au.

[102] N. del T.: 'Network Address Translation' en el original en inglés

[103] Véase el fichero `/etc/protocols` para buscar los nombres y números de los protocolos.

[104] N. del T.: 'Type Of Service' en el original en inglés

[105] N. del T.: 'datalink' en el original en inglés

[106] N. del T.: "aceptado"

[107] N. del T.: "rechazado"

[108] Traducción de IP Chains Firewall. N. del T.

[109] Traducción de *ping flooding* N. del T.

[110] Traducción de ICMP Echo Request. N. del T.

[111] No es bueno hacer esto, si su máquina Linux se emplea como encaminador. Si desactiva reenvío IP, ¡ dejará de encaminar !. Sólo haga esto en una máquina con una sola interfaz física de red.

[112] Real Audio es una marca de Progressive Networks Corporation.

[113] Ambrose es accesible en la dirección de correo ambrose@writeme.com.

[114] daemons

[115] Internet Daemon en inglés, N. del T.

[116] espera

[117] no espera

[118] single−threaded

[119] multi−threaded

[120] login ID

[121] TCP wrapper

[122] wrapper en inglés

[123] escrito por Wietse Venema, wietse@wzv.win.tue.nl.

[124] Habitualmente sólo los nombres de hosts locales obtenidos a través de búsquedas en `/etc/hosts` no contienen puntos.

[125] Mientras este nombre sugiere una medida extrema, la palabra clave PARANOID es buena por omisión, y le protege contra hosts maliciosos que pretenden hacerse pasar por lo que no son. No todos los **tcpd** se proporcionan con la palabra clave PARANOID compilada; si el suyo no la tiene, necesitará recompilar **tcpd** para usarla.

[126] shell

[127] Network Information System

[128] Network File System

[129] Representación de Datos Externa

[130] mapeador de puertos

[131] OpenSSH se desarrolló por el proyecto OpenBSD y representa un ejemplo de los beneficios del software libre.

[132] fingerprint

[133] Se puede contactar con Swen en swen@uni−paderborn.de. Los clientes NIS están disponibles como `yp−linux.tar.gz` en metalab.unc.edu dentro de `system/Network`.

[134] Se puede contactar con Peter en pen@lysator.liu.se. La versión actual de NYS es la 1.2.8.

[135] Se puede contactar con Thorsten en kukuk@uni−paderborn.de.

[136] DBM es una sencilla biblioteca para manejo de bases de datos que usa técnicas de dispersión (hashing) para aumentar la velocidad de las operaciones de búsqueda. Existe una implementación libre de DBM del proyecto GNU llamada `gdbm`, que es parte de la mayoría de las distribuciones de Linux.

[137] Para habilitar el uso del método `/etc/hosts.allow`, puede que tenga que recompilar el servidor. Por favor, lea las instrucciones del fichero `README` incluido en la distribución.

[138] El mapeador de puertos seguro está disponible vía FTP anónimo en ftp.win.tue.nl en el directorio `/pub/security/`.

[139] Puede contactar a Rick en jrs@world.std.com.

[140] Actualmente, usted puede omitir el argumento `t nfs` porque la orden **mount** interpreta por los dos puntos que esto especifica un volumen NFS.

[141] No se dice sistema de ficheros porque no son propiamente sistemas de ficheros.

[142] minor timeout

[143] major timeout

[144] write−behind

[145] Novell y NetWare son marcas registradas de Novell Corporation.

[146] Se puede contactar con Alan en alan@lxorguk.ukuu.org.uk.

[147] Se puede contactar con Greg en gpage@sovereign.org.

[148] Se puede contactar con Volker en lendecke@namu01.gwdg.de.

[149] Se puede contactar con Ales en A.Dryak@sh.cvut.cz. Se puede contactar con Martin en mstover@freeway.de.

[150] Se puede encontrar información sobre Caldera en *http://www.caldera.com/*.

[151] Parece que los administradores de sistemas han estado probando algunos de los artículos de la Cervecera Virtual antes de elegir alguno de esos nombres de cola de impresión. ¡Esperemos que los nombres que usted elija tengan más sentido!

[152] Autoría y copyright de Ian Taylor, 1995.

[153] También se incluye en el *Manual del Administrador de Sistemas* 4.4BSD.

[154] Si sólo quiere probar UUCP, obtenga el número de un sistema cercano a usted. Apunte el nombre de usuario y la clave son públicos para permitir posibles transferencias anónimas. En la mayoría de los casos, son algo como uucp/uucp o nuucp/uucp.

[155] La única limitación es que no puede ser más largo que siete caracteres, para no confundir a algunos nodos con sistemas de ficheros que imponen un estrecho límite en los nombres de ficheros.

[156] El Proyecto de Mapeado UUCP registra los nombres de nodos UUCP en todo el mundo y asegurándose de que sean únicos.

[157] Los UUCPs Versión 2 antiguos no hacen saber su nombre cuando se les llama; de todos modos, sí lo hacen las implementaciones más recientes, y así lo hace Taylor UUCP.

[158] Por ejemplo, muchas instalaciones de compañías privadas requieren que marque un 0 o un 9 para obtener línea hacia el exterior.

[159] La velocidad en baudios del terminal *tty* debe configurarse al menos como la máxima velocidad de transferencia.

[160] Si el sistema remoto usa también Taylor UUCP es seguro que obedecerá.

[161] A algunos módems parece no gustarles esto y se cuelgan ocasionalmente.

[162] **bsmtp** se usa para enviar correo con SMTP por lotes.

[163] Puede usar una tilde (~) para referirse al directorio público de UUCP, pero sólo en los ficheros de configuración de UUCP; fuera de ellos suele traducirse en el directorio principal del usuario.

[164] Tenga en cuenta que **tcpd** normalmente tiene el modo 700, por lo que debe invocarlo como usuario root, no uucp. **tcpd** se discute con mayor detalle en Capítulo 12.

[165] El **mgetty** de Gert Doering es una bestia de esa calaña. Corre sobre varias plataformas, incluyendo SCO Unix, AIX, SunOS, HP−UX y GNU/Linux.

[166] Esta opción no se encuentra presente en la versión 1.04.

[167] Es decir, con nombres que empiezan con un punto. Esos ficheros no suele mostrarlos la orden **ls**.

[168] ¡Lea el RFC−1437 si no lo cree!

[169] Puede localizar a Guylhem en guylhem@danmark.linux.eu.org.

[170] Se suele añadir una *firma* (signature) o `.sig` a un mensaje, que normalmente contiene información sobre el autor, junto con un chiste o cita célebre. Se separa del resto del mensaje con una línea que contiene `−−` seguido por un espacio.

[171] Esto es así porque el espacio de disco se asigna usualmente en bloques de 1024 Bytes. Incluso un mensaje de unas docenas de Bytes ocupara 1 Kb completo.

[172] Si desea acceder a una dirección DEC desde el entorno RFC−822, puede hacerlo de la siguiente manera anfitrión::usuario@relay, siendo relay el nombre de un repetidor DEC conocido.

[173] Los mapas para los sitios registrados en el proyecto de mapeado UUCP se distribuyen a través del grupo de noticias comp.mail.maps; otras organizaciones pueden publicar distintos mapas para sus redes.

[174] Suelen ser publicados en news.lists.ps−maps. Ojo! Son INMENSOS.

[175] N. del T: exploits en el original.

[176] starting at the beginning of the next line, delete all characters up to and including the next newline.

[177] N. del T. mailers en el original

[178] N. del T: relay anfitrións en inglés

[179] N. del T. hubs en inglés

[180] El diccionario gratuito de la computación en línea puede ser encontrado empaquetado en muchas distribuciones de GNU/Linux, o en línea en su página web en http://wombat.doc.ic.ac.uk/foldoc/.

[181] N. del T:literalmente "scumbags" o sacos de mierda

[182] N. del T: Del inglés "black hole", agujero negro

[183] mail spool

[184] Otros lugares posibles son `/etc/rc.d/init.d` y `rc.inet2`. El último es común en sistemas que usen una estructura al estilo BSD para los ficheros de administración del sistema en el directorio `/etc`.

[185] Éste es el nuevo lugar estándar de **sendmail** de acuerdo con el Estándar para el Sistema de Ficheros en GNU/Linux. Otro lugar común es `/usr/lib/sendmail`, que está más orientado a que lo usen programas de correo que no estén especialmente configurados para GNU/Linux. Puede definir estos dos nombres de ficheros como enlaces simbólicos hacia Exim para que los programas y guiones que invoquen a *sendmail* invoquen a Exim en su lugar para que haga las mismas cosas.

[186] De todos modos, algunos clientes usan el protocolo SMTP para pasarle los mensajes al agente de transporte, llamándole con la opción `bs`.

[187] Use `kill HUP` *`pid`*, siendo *`pid`* el ID del proceso que **inetd** obtiene a partir de un listado de **ps**.

[188] La carga del sistema es una medida estándar en Unix de la cantidad media de procesos que están en la cola esperando ejecutarse. El **uptime** muestra la carga media del sistema tomada en los 1, 5 y 15 minutos previos.

[189]

Esto es una simplificación. Los direccionadores son capaces de pasar direcciones a los transportes para que las envíen a anfitriones remotos, y de manera similar, los encaminadores son capaces de pasar direcciones a los transportes locales para que escriban el mensaje en un fichero o en un "pipe". Los encaminadores también pueden pasar direcciones a los direccionadores en algunas circunstancias.

[190] Un direccionador se salta si la dirección que se va a procesar es una que ya haya procesado previamente durante la generación de la dirección actual.

[191] Por favor, si elige usar un programa para sus ausencias, ¡asegúrese de que no responde a los mensaje enviados desde listas de correo! Resulta muy molesto descubrir que alguien se ha ido de vacaciones y encontrar un mensaje de ausencia por cada mensaje que hayan recibido. Para los administradores de listas de correo: esto es un buen ejemplo de por qué es una mala práctica forzar el campo `Responder a:` de los mensajes de una lista de correo para que contenga la dirección de envío a esa misma lista de correo.

[192] El formato de los mensajes de Usenet News está especificado en la RFC−1036, Standard for interchange of USENET messages.

[193] Espera un minuto: 60 MB a 9,600 bps, son 60 millones mutiplicados por 1024, eso es ¡ mutter, mutter Eh! eso es 34 horas!

[194] N. del T.: inundación.

[195] N. del T.: notación bang−path

[196] batching

[197] empujar

[198] tirar

[199] Algunas persontas dicen que Usenet es una conspiración entre vendedores de modems y discos duros.

[200] Debe haber una diferencia entre los grupos que existen en su servidor y aquellos que su servidor está preparado para recibir. Por ejemplo, la lista de subscripción puede especificar comp.all, que debe enviar todos los grupos por debajo de comp, pero en nuestro sitio podemos no tener listados todos los grupos de esa jerarquía en el fichero `active`. Los artículos enviados a esos grupos serán movidos a junk.

[201] Recuerde que debe ser el **crontab** de noticias; los permisos de ficheros no serán cambiados.

[202] Puede obtener el código de C−News en su servidor principal que se encuentra en ftp.cs.toronto.edu `/pub/c−news/c−news.tar.Z`

[203] Esto sucede frecuentemente, por ejemplo, digamos que un artículo enviado desde Hamburgo, para ir a Frankfurt, vaya vía reston.ans.net en Holanda , o incluso vía algún servidor en EE.UU.

[204] Tal como se distribuye con C−News, compcun usa compress con la opción 12−bit , ya que éste es el mínimo común denominador de la mayoría de los servidores. Puede hacer una copia del guión, digamos compcun16, y usar el método de compresión 16−bit. De todas formas, la mejora en rendimiento no es muy impresionante.

[205] La fecha de llegada del artículo se almacena en el campo de en medio de la línea de historia, dado en segundos desde el 1 de Enero de 1970

[206] No se *por qué* ocurre esto, a mí me sucede de vez en cuando.

[207] Yo tampoco intentaría esto en Internet.

[208] Cuando se publica un artículo por NNTP, el servidor siempre añade al menos un campo a la cabecera, `NNTP−Posting−Host:`. El campo contiene el nombre de la máquina del cliente.

[209] Existe el mismo problema con el Protocolo para la Transferencia Sencilla de Correo (SMTP), aunque muchos agentes transportadores de correo ya ofrecen mecanismos para prevenir el "spoofing".

[210] *leafnode* se encuentra disponible por FTP anónimo en **wpxx02.toxi.uni−wuerzburg.de** en el directorio `/pub/`.

[211] Los sitios muy pequeños deberían considerar el uso de un programa como **leafnode**, el cuál es un servidor NNTP con caché. leafnode, está disponible en la dirección http://wpxx02.toxi.uni−wuerzburg.de/~krasel/leafnode.html.

[212] Esto es indicado por la cabecera *Date:*; y el límite es usualmente dos semanas.

[213] channels

[214] Hilar 1.000 artículos cuando se conversa con un servidor activo puede tomar fácilmente alrededor de cinco minutos, que sólamente el más dedicado adicto a las noticias de Internet encontraría aceptable.

[215] El nombre aparentemente es una sigla de NetNews Read & Post Daemon

[216] modo lectura

[217] PGP y GPG son herramientas diseñadas para autenticar o encriptar mensajes utilizando técnicas de clave pública. GPG es la versión GNU de PGP. GPG puede encontrarse en http://www.gnupg.org/, y PGP en http://www.pgp.com/.

[218] El tiempo se reduce drásticamente si el servidor NNTP crea los hilos por sí mismo y permite al cliente recibir esos datos; por ejemplo, un servidor que permite hacer ésto es INN.

[219] Esta es la razón por la cual se obtendrán mensajes de error feos al invocar a **tin** como usuario. De todas formas, no se deberían realizar trabajos de rutina como root.

[220] N. del T.: Leer Noticias

[221] N. del T.: Ordenado en hilos, o temas

[222] N. del T.: Lista de grupos todos

[223] Tenga en cuenta que C News (descrito en Capítulo 21) no actualiza su línea de flotación automáticamente; hay que ejecutar **updatemin** para ello.

[224] N. del T.: Sin Noticias

[225] N. del T.: Algo así como si no hay noticias es buena noticia, **nn** es mejor

[226] Esto se debe a que el orden debe coincidir con el del fichero binario `MASTER`

[227] N. del T. Derechos Reservados en el sentido de GNU http://www.gnu.org/copyleft/copyleft.es.html