Guía de Administración de Redes con Linux

Versión final 1.0, 1 de Junio de 1999.
Fecha de montaje: 30 de mayo de 1999

Para Britta

Nota Legal

UNIX es una marca registrada de Univel.
Linux no es una marca registrada, y no tiene relación con UNIX™ o Univel.

Kattreinstr. 38, 64295 Darmstadt, Alemania
okir@monad.swb.de

Índice General

Índice de Figuras

Prólogo a la Edición en Castellano

Los últimos años de la década de los 90 son, claramente, los de la explosión definitiva de la Informática Personal. Es la época en la que ordenadores de mediana potencia pueblan las casas particulares. También es el momento en el que Internet se hace popular. Y como ya no es difícil que en un hogar haya más de un PC, tal vez interesaría tenerlos conectados en red local: al fin y al cabo una tarjeta Ethernet puede costar menos de 5000 pesetas.

Este libro trata sobre Linux y su capacidad de red, capacidad que no tiene nada que envidiar a la que pueda tener cualquier otro sistema UN⋆X. Otros sistemas operativos quieren tener ahora, a toda prisa, capacidades de acceso a la Red, pero recordemos que la tecnología de Internet nació en Unix. Por esto, esos sistemas seguirán estando, durante mucho tiempo, a años luz de lo que Unix puede ofrecer.

Por lo tanto, le doy mi enhorabuena si ha elegido Linux u otro Unix para acceder a la Red y para trabajar. Lo que aquí explicamos intentará echarle una mano en las tareas de administración de red. Le será especialmente útil si su sistema Linux o Unix es servidor de una pequeña red local o de Internet. Nuestro intento como traductores ha sido acercar las ventajas de Unix en red al usuario hispanohablante, ya que creemos que es un asunto aun no cubierto por la bibliografía existente en nuestro idioma.

Hemos intentado hacer una traducción homogénea, pese a haber colaborado en ella gran cantidad de gente. Gracias a la propia Internet, hemos podido tener un canal de comunicación fluido para discutir cualquier duda que pudiera surgir entre los traductores, adoptando finalmente un glosario común de términos informáticos. Esperamos que el resultado merezca la pena.

Ahora es el momento de dar las gracias a todos aquellos que han colaborado en la traducción o en las posteriores revisiones. Estas personas son 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.

Por último, debo agradecer el nacimiento y desarrollo de la idea de documentar Linux en nuestro idioma, a Ramón Gutiérrez y de nuevo a Alfonso Belloso y Francisco J. Montilla.

Juan José Amor
Mayo de 1997
revisado en Abril de 1999

Nota Acerca de Esta Traducción

Esta guía es traducción de la versión 1.0 del libro “Linux Network Administration Guide”, escrito por Olaf Kirch entre 1992 y 1994. Es un trabajo más del proyecto LuCAS, el LDP español; y ha sido coordinado por Juan José Amor, quien desea recibir cualquier tipo de comentario acerca de erratas o sugerencias en su dirección de correo electrónico: `jjamor@ls.fi.upm.es`.

Prefacio

Las redes de ordenadores parecen estar acercándose a la situación de los televisores y los hornos microondas, debido a la publicidad que Internet está teniendo, y el hecho de que la gente de opinión más respetada en el mundo de la Informática, esté todo el tiempo hablando de las virtudes de la *red*. Es decir, Internet está teniendo una cobertura informativa inusual, y las autoridades en ciencia social están integrándose en los grupos de noticias de Usenet para dirigir las investigaciones sobre la "Cultura Internet". Las compañías suministradoras trabajan en la introducción de nuevas técnicas de transmisión como ATM, que ofrecen un ancho de banda varias veces superior al que ofrecen las redes utilizadas actualmente.

En realidad, las redes han estado presentes durante bastante tiempo. La conexión de ordenadores para formar redes de area local ha sido práctica común incluso en instalaciones pequeñas, así como lo han sido los enlaces a larga distancia usando líneas telefónicas públicas. Un rápido crecimiento del conglomerado de redes a nivel mundial ha hecho, sin embargo, que integrarse en la aldea global sea una opción viable incluso para pequeñas organizaciones sin beneficio de usuarios de ordenadores privados. Instalar un nodo de Internet con capacidad de correo y noticias ofreciendo acceso telefónico ha pasado a ser algo accesible, y con la RDSI acelerará sin duda esta introducción.

Hablar de redes de ordenadores a menudo significa hablar de UNIX. Por supuesto, UNIX no es el único sistema operativo con capacidades de red, ni seguirá siendo la puerta de entrada, pero sí ha estado en el negocio de las redes por mucho tiempo y seguramente continuará haciendolo al menos durante bastante tiempo.

Lo que lo hace particularmente interesante para los usuarios privados es que ha habido gran actividad para conseguir sistemas operativos tipo UNIX gratuitos para PC, como 386BSD, FreeBSD — y Linux. Sin embargo, Linux *no* es UNIX. Eso es una marca comercial registrada de quien actualmente tenga los derechos sobre ella (Univel, en el momento de escribir este libro). Linux, en cambio, es un sistema operativo que lucha por ofrecer toda la funcionalidad que requieren los estándares POSIX para sistemas operativos tipo UNIX, aunque es una reimplementación completa, desde cero.

El núcleo de Linux fue escrito enteramente por Linus Torvalds, quien lo comenzó como un proyecto para conseguir conocer el Intel i386, y para "hacer un MINIX mejor". MINIX era entonces otro sistema operativo popular para PC que ofrecía los ingredientes vitales de la funcionalidad UN⋆X, y fue escrito por el Profesor Andrew S. Tanenbaum.

Linux está cubierto por la Licencia Pública General (GPL) GNU, que permite la libre distribución del código (léase, por favor, la GPL en el apéndice C para tener una definición de lo que es "software libre" ("free software")). Superando sus males de joven, y atraido por una siempre creciente base de programas de aplicación gratuitos, se está convirtiendo rápidamente en el sistema operativo de elección de muchos usuarios de PC. Tanto el núcleo como la librería C han llegado a ser tan buenos que la mayoría del software estándar se puede compilar sin esfuerzo mayor que el que se requiere en cualquier otro sistema tipo UN⋆X, y una amplia variedad de distribuciones empaquetadas de Linux le permiten prácticamente volcarlo en su disco duro y comenzar a manejarlo.

Documentación de Linux

Una de las quejas que se recogen frecuentemente en torno a Linux (y al software gratuito en general) es sobre el lamentable estado de la documentación o su completa carencia. En los primeros días lo usual era que un paquete viniese con unas útiles notas de instalación y *README*s (*LEAME*s). Éstos daban a los magos de UN⋆X, moderadamente experimentados, suficiente información para instalarlo y ejecutarlo con éxito, pero dejaban al típico novato fuera de juego.

Allá por finales de 1992, Lars Wirzenius y Michael K. Johnson sugirieron la formación del Linux Documentation Project (Proyecto de Documentación de Linux), o LDP, con el fin de proporcionar un conjunto coherente de manuales. Lejos de contestar preguntas tipo "¿Cómo?", o "¿Por qué?", o "¿Cual es el significado de la vida, el universo y todo lo demás?", estos manuales intentan cubrir la mayoría de aspectos del uso y funcionamiento de un sistema Linux sin necesidad de graduarse previamente en UN⋆X.

Entre los logros del LDP están el *Installation and Getting Started Guide*[1], escrita por Matt Welsh, el *Kernel Hacker's Guide*[2] de Michael K. Johnson, y el proyecto de páginas de manual coordinado por Rik Faith, que hasta ahora ha producido un conjunto de unas 450 páginas de manual para la mayoría de las llamadas al sistema y librerías de C. La *System*

[1]N. del T.: Y traducida, como este manual, por el Proyecto LuCAS, y cuyo título en castellano es "Linux: Instalación y Primeros Pasos"

[2]N. del T.: Guía sobre el Núcleo

Administrators' Guide[3], escrita por Lars Wirzenius, está aun en estado *alpha*. También se está preparando una Guía de Usuario.

Este libro, la *Guía de Administración de Redes con Linux*, es también parte de la serie LDP. Como tal, puede ser copiado y distribuido libremente bajo la licencia de copia del LDP que se reproduce en la segunda página.

Sin embargo, los libros del LDP no son la única fuente de información en Linux. En este momento hay más de una docena de HOWTOs (COMOs) que se publican regularmente en **comp.os.linux.announce** y es posible encontrarlos en múltiples servidores de FTP. Los HOWTOs son documentos cortos de unas pocas páginas que dan una breve introducción a materias como el soporte de Ethernet bajo Linux, o la configuración del software de noticias Usenet, y responde a las preguntas más frecuentes. Generalmente proporcionan la información más precisa y actualizada disponible sobre la materia. Podrá encontrar una lista de los HOWTOs disponibles en la "Bibliografía" hacia el final de este libro.

Sobre Este Libro

Cuando me uní al Linux Documentation Project en 1992, escribí dos pequeños capítulos sobre UUCP y *smail*, con los que quise contribuir a la *System Administrator's Guide*. El desarrollo del TCP/IP acababa de comenzar, y cuando esos "pequeños capítulos" comenzaron a crecer, me pregunté si no estaría bien tener una Guía de Red. "Fantástico", dijo todo el mundo, "Yo diría, ¡a por ello!". De modo que fui, y escribí una primera versión de la Guía de Red, que di a conocer en Septiembre de 1993.

La nueva Guía de Red que está Ud. leyendo ahora es una reescritura completa que contiene varias aplicaciones nuevas que ya están disponibles para los usuarios de Linux desde la primera versión.

El libro está organizado estrictamente en la secuencia de pasos que debe de seguir para configurar su sistema para redes. Comienza discutiendo los conceptos básicos de redes, y, en particular, las redes basadas en TCP/IP. Entonces, poco a poco caminamos desde la configuración de TCP/IP a nivel de dispositivos hasta la puesta en marcha de aplicaciones comunes como *rlogin* y parecidas, el Sistema de Ficheros de Red (NFS), y el Sistema de Información de Red (NIS). A esto le sigue un capítulo sobre cómo hacer de su máquina un nodo UUCP. El resto del libro está dedicado a dos grandes aplicaciones que corren por encima de TCP/IP y UUCP: el correo elecrónico y las noticias.

La parte de correo electrónico (e-mail) conforma una introducción a las partes más íntimas del transporte y encaminado del correo, y los miles de esquemas de direccionamiento con los que se puede encontrar. Describe la configuración y gestión de *smail*, un agente de transporte de correo muy usado en pequeños nodos de correo, y *sendmail*, que es para gente

[3] N. del T.: Guía del Administrador de Sistemas

que tiene que hacer encaminamientos más complicados, o tiene que manejar gran volumen de correo. El capítulo *sendmail* ha sido escrito y aportado por Vince Skahan.

La parte de noticias intenta ofrecerle una visión de cómo funcionan las noticias de Usenet, abarca a C-News, el software de transporte de noticias más usado de momento, y el uso de NNTP para proporcionar acceso a la lectura de noticias a una red local. El libro cierra con un capítulo corto sobre cómo usar los lectores de noticias más populares de Linux.

La Versión Impresa Oficial

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.[4] Esto significa que Ud. puede elegir libremente: puede conseguir las fuentes LaTeX 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 querría Ud. 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?[5] ¿O hay alguna diferencia entre estas versiones?

Las respuestas son “depende”, “no, definitivamente no” y “sí y no”. O'Reilly and 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.

¿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

[4] El aviso de copyright se reproduce en la página inmediantemente posterior a la del título.

[5] 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.

de lo que es ahora sin su contribución. Todas estas ediciones han recalado en la versión en línea, como lo será cualquier cambio que se haga a la Guía de Red durante la fase de edición en O'Reilly. De modo que no habrá diferencia en contenido. Aun así, la versión de O'Reilly *será* diferente: Por un lado, la gente de O'Reilly ha trabajado mucho en el aspecto y forma, dando lugar a un diseño mucho más agradable que jamás se lograría con el LaTeX estándar. Por otro lado, contará con un par de avances como un índice mejorado, y más y mejores figuras.

Más Información

Si sigue las instrucciones de este libro y algo no funciona, por favor, sea paciente. Algunos de sus problemas pueden deberse a fallos estúpidos por mi parte, pero también pueden ser causados por cambios en el software de red. Además, probablemente debería preguntar primero en **comp.os.linux.help**. Hay muchas posibilidades de que usted no sea el único con esos problemas, y probablemente ya se conozca una solución o al menos se esté trabajando en ello. Si tiene oportunidad, debería intentar conseguir el último núcleo y versión de red de alguno de los servidores de FTP de Linux, o de una BBS próxima a usted. Muchos problemas vienen de usar software en distintas fases de desarrollo, que fallan al intentar funcionar juntos. Después de todo Linux es un "trabajo en progreso".

Otro buen lugar donde informarse del estado actual de desarrollo es el Networking HOWTO[6]. Lo mantiene Terry Dawson[7]. Se incluye en **comp.os.linux.announce** una vez al mes, y contiene la información más actualizada. La versión actual puede obtenerse de (entre otros) **tsx-11.mit.edu**, en */pub/linux/doc*. Para problemas que no puede resolver de ninguna forma, también puede contactar con el autor de este libro en la dirección que se facilita en el prefacio. Sin embargo, por favor, absténgase de pedir ayuda a los desarrolladores. Ellos ya están dedicando la mayor parte de su tiempo libre a Linux, y a veces incluso viven más alla de la red `:-)`.

Sobre los autores

Olaf ha sido un usuario de UNIX y administrador a tiempo parcial por un par de años mientras estudiaba matemáticas. Actualmente, trabaja como programador de UNIX y está escribiendo un libro. Uno de sus deportes favoritos es hacer cosas con *sed* que otra gente hace con su intérprete *perl*. Él se divierte tanto con esto como con ir al monte con una mochila y una tienda de campaña.

Vince Skahan ha estado administrando gran número de sistemas UNIX desde 1987 y actualmente ejecuta sendmail+IDA en aproximadamente 300 estaciones UNIX para unos

[6] N. del T.: El HOWTO sobre redes

[7] A Terry Dawson se le puede localizar en **terryd@extro.ucc.su.oz.au**.

2000 usuarios. Admite haber perdido un considerable número de horas de sueño editando unos pocos ficheros *sendmail.cf* "a pelo" antes de descubrir sendmail+IDA en 1990. También admite haber esperado ansiosamente el envío de la primera versión de sendmail basada en perl incluso para sus diversiones más oscuras[8]...

A Olaf se le puede encontrar en la siguiente dirección:

Olaf Kirch
Kattreinstr. 38
64295 Darmstadt
Germany

okir@monad.swb.de

A Vince se le puede encontrar en:

Vince Skahan
vince@victrola.wa.com

Estamos abiertos a sus preguntas, comentarios, postales, etc. Sin embargo, le pediríamos que *no* nos telefonee a no ser que sea realmente importante.

Agradecimientos

Olaf dice: Este libro debe gran parte de su contenido a gente que dedicó tiempo a revisarlo y ayudó a depurar muchos fallos, tanto técnicos como gramáticos (jamás supe que existiera algo como un participio oscilante[9]). El más riguroso de ellos fue Andy Oram de O'Reilly and Associates.

Estoy muy en deuda con Andres Sepúlveda, Wolfgang Michaelis, Michael K. Johnson, y todos los desarrolladores que dedicaron su tiempo a comprobar la información de la Guía de Red. También quiero agradecer a todos los que leyeron la primera versión de la Guía de Red y me enviaron sus correcciones y sugerencias. Espero que puedan encontrar una lista completa de los que han contribuido en el fichero *Thanks* de la distribución en línea. Por fin, este libro no hubiese sido posible sin el apoyo de Holger Grothe, quien me facilitó lo crucial, la conexión a Internet.

También quisiera agradecer a los siguientes grupos y compañías que imprimieron la primera versión de la Guía de Red y han donado dinero incluso para mí, o para el Proyecto de Documentación de Linux en su conjunto.

[8] ¿No crees que lo podríamos hacer con *sed*, Vince?

[9] N. del T.: Realmente no sé si en inglés existe algo así, o es simplemente una broma del autor

- Linux Support Team, Erlangen, Germany
- S.u.S.E. GmbH, Fuerth, Germany
- Linux System Labs, Inc., United States

Vince dice: Mis agradecimientos van para Neil Rickert y Paul Pomes por su gran ayuda estos años concerniente al cuidado y mantenimiento de sendmail+IDA y a Rich Braun por portar inicialmente el sendmail+IDA a Linux. El mayor agradecimiento va para mi esposa Susan por todo su apoyo en este y otros proyectos.

Convenciones Tipográficas

Al escribir este libro se han usado ciertas convenciones tipográficas para marcar comandos de shell, argumentos variables, etc. Se detallan a continuación.

Fuente Negrita

Se usa para marcar nombres de máquinas y direcciones de correo, así como nuevos conceptos y avisos.

Fuente Itálica

Se usa para marcar nombres de fichero, comandos UNIX, y palabras clave en ficheros de configuración. También se usa para *enfatizar* texto.

`Fuente de máquina de escribir`

Se usa para representar interacción con la pantalla, como la interacción del usuario ejecutando un programa.

También se usa para ejemplos de código, ya sea un fichero de configuración, un script de shell, o algo así.

`Fuente de máquina de escribir inclinada`

Se usa para marcar meta-variables en el texto, especialmente en representaciones de la línea de comandos. Por ejemplo,

```
$ ls -l ejemplo
```

donde *`ejemplo`* "representaría" un nombre de fichero, como, por ejemplo, */tmp*.

Tecla — Representa una tecla a pulsar. Lo verá a menudo de esta forma:

Pulse return para continuar.

◇ — Un rombo en el margen, como un rombo negro en una pista de esquí, marca "peligro" o "precaución". Lea los parrafos marcados así cuidadosamente.

`$` y `#` — Cuando preceden a un comando de shell a escribir, esto denota el prompt del shell. El símbolo '`$`' se usa cuando el comando puede ser ejecutado como un usuario normal; '`#`' significa que el comando requiere privilegios de superusuario.

El Proyecto de Documentación de Linux

El Linux Documentation Project (Proyecto de Documentación de Linux), o LDP, es un amplio equipo de escritores, revisores, y editores que están trabajando juntos para facilitar documentación completa para el sistema operativo Linux. El coordinador general de este proyecto es Matt Welsh, quien está ayudado en gran medida por Lars Wirzenius y Michael K. Johnson.

Este manual está dentro de un grupo de varios que están siendo distribuidos por el LDP, incluyendo una Guía de Usuarios de Linux, otra de Administradores de Sistemas, otra para Administradores de Red, y la Guía de Desarrolladores del Núcleo. Estos manuales están disponibles en formato fuente LaTeX, formato `.dvi`, y salida postscript por FTP anónimo desde `nic.funet.fi`, en el directorio `/pub/OS/Linux/doc/doc-project`, y desde `tsx-11.mit.edu`, en el directorio `/pub/linux/docs/guides`.

Animamos a todo aquél con afición a la escritura o a la edición que se una a nosotros para mejorar la documentación de Linux. Si tiene acceso a correo Internet, puede unirse al canal `DOC` de la lista de correo `Linux-Activists` enviando un correo a

```
linux-activists-request@niksula.hut.fi
```

con la línea

```
X-Mn-Admin: join DOC
```

en la cabecera o como primera línea del cuerpo del mensaje. Un correo vacío sin la línea de cabecera adicional hará que el servidor de correo devuelva un mensaje de ayuda. Para abandonar el canal, envíe un mensaje a la misma dirección, incluyendo la línea

```
X-Mn-Admin: leave DOC
```

El Proyecto LinUx en CAStellano (LuCAS)

La versión en Castellano que tiene en sus manos es un fruto más del proyecto LuCAS (LinUx en CAStellano). Es un proyecto para documentar el Linux en el idioma de Cervantes, coordinado inicialmente por Alfonso Belloso[10] y actualmente por Juan José Amor[11] e Ismael Olea[12].

Los manuales que produce el grupo se encuentran disponibles en las páginas de WWW, `http://lucas.hispalinux.es/` así como en el servidor de FTP `slug.ctv.es`,

[10] Alfonso está localizable en **alfon@iies.es**

[11] Juan José está disponible en la dirección **jjamor@ls.fi.upm.es**

[12] cuya dirección de correo es **olea@iname.com**

directorio `pub/LuCAS`. Parte del contenido de este servidor se encuentra reflejado en `sunsite.unc.edu`. En particular, este manual se encuentra en el directorio `/pub/Linux/docs/linux-doc-project/network-guide/translations/es` de esa máquina.

El grupo LuCAS trabaja codo con codo, con el grupo INSFLUG (*Impatient & Novatous Spanish Fido Linux User's Group*), dedicado a la traducción de documentos cortos HOWTO (COMOs); y coordinados por Francisco José Montilla[13]. Sus trabajos pueden encontrarse en el servidor `www.insflug.org` así como en el propio LuCAS.

Si desea unirse al grupo, no dude en escribirnos. Mientras lo piensa, escuche un poco lo que se cuenta en la lista de correo que mantenemos: escriba a **lucas-request@hispalinux.es** un mensaje con la orden *subscribe*.

[13]La dirección electrónica de Francisco José es **pacopepe@insflug.org**

Estándares de Sistema 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 va a implementarse 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 de Sistema de Ficheros de Linux puede obtenerse de la mayoría de servidores FTP de Linux y sus imágenes; por ejemplo, puede encontrarlo en **sunsite.unc.edu** bajo */pub/linux/docs*. Daniel Quinlan, coordinador del grupo FSSTND puede ser localizado en **quinlan@bucknell.edu**.

Capítulo 1

Introducción a las Redes

1.1 Historia

La idea de red es probablemente tan vieja como la de las telecomunicaciones. Consideremos a la gente que vivía en la edad de piedra, donde los tambores se habrían utilizado para transmitir mensajes entre individuos. Suponga que el cavernícola A quiere invitar al cavernícola B a un partido de lanzamiento de rocas contra el otro, pero viven demasiado lejos como para que B oiga a A golpear su tambor. ¿Cuáles son las opciones de A? Podría 1) ir a la choza de B, 2) hacerse con un tambor más grande, o 3) pedirle a C, que vive a mitad de camino entre los dos, que retransmita el mensaje. La última opción es lo que se llama una red.

Claro, que ya ha pasado un tiempo desde los primeros intentos de nuestros antepasados. Hoy en día tenemos ordenadores que hablan entre sí a través de vastas conexiones de cables, fibras ópticas, microondas, y otros medios parecidos, para quedar para el partido del sabado. A continuación trataremos sobre las maneras en que esto se realiza, pero olvidándonos de los cables, así como de la parte del partido.

En esta Guía escribiremos sobre dos tipos de redes : las basadas en UUCP, y las basadas en TCP/IP. Éstos son conjuntos de protocolos y paquetes de software que proporcionan medios para transportar datos entre dos ordenadores. En este capítulo veremos ambos tipos y discutiremos sus principios fundamentales.

Definiremos una red como un conjunto de *nodos* que son capaces de comunicarse entre sí, a menudo contando con los servicios de varios nodos especializados que conmutan datos entre los participantes. Los nodos suelen ser ordenadores, aunque no es necesario; podemos considerar también terminales X o impresoras inteligentes como nodos. Pequeñas aglomeraciones de nodos también se llaman *instalaciones*.[1]

La comunicación sería imposible sin algún tipo de lenguaje o código. En las redes de ordenadores, estos lenguajes son llamados colectivamente *protocolos*. Sin embargo, no debería pensar en protocolos escritos, sino más bien en el código de comportamiento altamente formalizado que se observa cuando se encuentran los jefes de estado. De un modo muy similar, los protocolos usados por las redes de ordenadores no son sino normas muy estrictas para el intercambio de mensajes entre dos o mas nodos.

1.2 Redes UUCP

UUCP es una abreviatura de Unix-to-Unix Copy (Copia de Unix a Unix). Comenzó siendo un paquete de programas para transferir ficheros sobre líneas serie, programar esas transferencias, e iniciar la ejecución de programas en el lugar remoto. Ha experimentado grandes cambios desde su primera implementación a finales de los setenta, pero aun es bastante espartano en los servicios que ofrece. Su principal aplicación es todavía en redes de área metropolitana (WAN) basadas en enlaces telefónicos.

UUCP comenzó a desarrollarse por los Laboratorios Bell en 1977 para la comunicación entre sus laboratorios de desarrollo de Unix. A mediados de 1978, esta red ya conectaba a más de 80 centros. Se ejecutaban aplicaciones de correo electrónico, así como de impresión remota; sin embargo, el uso principal del sistema era distribuir software nuevo y mejoras.[2] Hoy día, UUCP ya no está confinado en el entorno UN⋆X. Hay versiones comerciales disponibles para diversas plataformas, incluyendo AmigaOS, DOS, TOS de Atari, etc.

Una de las principales desventajas de las redes UUCP es su bajo ancho de banda. Por un lado, el equipo telefónico establece un límite rígido en la tasa máxima de transferencia. Por otro lado, los enlaces UUCP raramente son conexiones permanentes; en su lugar, los nodos se llaman entre sí a intervalos regulares. Es por ello, que la mayoría del tiempo que le lleva a un mensaje viajar por una red UUCP permanece atrapado en el disco de algún nodo, esperando al establecimiento de la próxima conexión.

A pesar de estas limitaciones, aún hay muchas redes UUCP funcionando en todo el mundo, utilizado principalmente por aficionados, ya que ofrecen acceso de red a usuarios privados a precios razonables. La razón fundamental de la popularidad del UUCP es que es baratísimo comparado con tener el ordenador conectado al Gran Cable de Internet. Para hacer de su ordenador un nodo UUCP, todo lo que necesita es un módem, software UUCP, y otro nodo UUCP que desee suministrarle correo y noticias.

[1] N. del T.: Del inglés *site*

[2] No parece que con el tiempo haya cambiado mucho esto. . .

1.2.1 Cómo usar UUCP

La idea que hay detrás de UUCP es bastante simple: como su nombre indica, básicamente copia ficheros de un nodo a otro, pero también permite realizar ciertas acciones en el nodo remoto.

Suponga que le está permitido que su máquina acceda a un nodo hipotético llamado **swim**, y le va ha hacer ejecutar el comando de impresión *lpr* para Ud. Entonces, podría escribir lo siguiente en su línea de comandos para que le imprima este libro en **swim**:[3]

```
$ uux -r swim!lpr !netguide.dvi
```

Esto hace que *uux*, un comando del repertorio UUCP, planifique un *trabajo* para **swim**. Este trabajo consta del fichero de entrada, *netguide.dvi*, y la petición de enviar este fichero a *lpr*. La opción `-r` indica a *uux* que no llame al sistema remoto inmediatamente, sino que almacene el trabajo hasta que se establezca la próxima conexión. A esto se le llama *spooling*, o almacenamiento en la cola.

Otra propiedad de UUCP es que permite reenviar trabajos y ficheros a través de varios nodos, suponiendo que éstos colaboren. Asumiremos que **swim**, el nodo del ejemplo anterior, tiene un enlace UUCP con **groucho**, el cual mantiene un gran número de aplicaciones UN⋆X. Para transferir el fichero *tripwire-1.0.tar.gz* hasta su máquina debería indicarlo así:

```
$ uucp -mr swim!groucho!~/security/tripwire-1.0.tar.gz trip.tgz
```

El trabajo creado pedirá a **swim** que traiga el fichero desde **groucho**, y lo envíe hasta su máquina, donde UUCP lo almacenará en *trip.tgz* y le notificará por correo la llegada del fichero. Esto ocurrirá en tres pasos. Primero, su máquina envía el trabajo a **swim**. La siguiente vez que **swim** establezca contacto con **groucho**, se transferirá el fichero de **groucho** a **swim**. El último paso es la transferencia del mismo desde **swim** hasta su máquina.

Los servicios más importantes que proporcionan las redes UUCP hoy en día son el correo electrónico y las noticias. Lo introduciremos brevemente y después lo veremos en más detalle.

El correo electrónico – *e-mail*[4] para abreviar – le permite intercambiar mensajes con usuarios de nodos remotos sin tener realmente que saber cómo acceder a estos nodos. La

[3] Si usa *bash*, la shell GNU Bourne Again SHell, tendría que quitar los signos de exclamación, porque los usa como su carácter de histórico.

[4] En el idioma castellano comienzan a aparecer adaptaciones más o menos afortunadas, como *e-milio*

tarea de dirigir un mensaje desde su máquina destino la realiza enteramente el sistema de manejo de correo. En un entorno UUCP, el correo generalmente se transporta ejecutando el comando *rmail* en el nodo vecino, pasándole la dirección del receptor y el mensaje. *rmail* reenviará entonces el mensaje a otro nodo, y seguirá así, hasta que alcance el nodo destino. Veremos esto en detalle en el capítulo 13.

La mejor forma de definir el servicio de noticias es considerarlo como un sistema de tablón de anuncios distribuido. Muy a menudo, este término se refiere a las noticias de Usenet, que es, con mucho, la más conocida red de intercambio de noticias, con un número de nodos participantes estimado en 120.000[5]. Los orígenes de Usenet se remontan a 1979, cuando, tras la aparición del UUCP con el nuevo Unix V7, tres estudiantes graduados tuvieron la idea de un intercambio de información general entre la comunidad Unix. Éstos escribieron algunos scripts, creando el primer sistema de noticias en red. En 1980, esta red conectaba **duke**, **unc**, y **phs**, y dos Universidades de Carolina del Norte, de forma aislada. Usenet creció más todavía posteriormente. Aunque su origen fue como una red basada en UUCP, ya no está limitada a un único tipo de redes.

La unidad básica de información es el *artículo*, que puede ser enviado a una jerarquía de grupos de noticias dedicadas a temas específicos. La mayoría de los nodos reciben únicamente una selección de todos los grupos de noticias, que transportan una media de 60Mb[6] de artículos por día.

En el mundo UUCP, las noticias generalmente se envían a través de un enlace UUCP, recolectando todos los artículos de los grupos de noticias solicitados, y empaquetandolos en varios *lotes*[7]. Éstos se envían al lugar receptor, donde se pasan al comando *rnews* que los desempaqueta y procesa posteriormente.

Finalmente, UUCP es también el medio elegido por muchos servidores de ficheros que ofrecen acceso público. Generalmente podrá acceder a ellos llamando con UUCP, accediendo como usuario invitado, y transferiéndose los archivos desde un área de ficheros públicamente accesible. Estas cuentas de invitado tienen, a menudo, un nombre de acceso y password como UUCP/nuucp o algo similar.

1.3 Redes TCP/IP

Aunque UUCP puede resultar una elección razonable para enlaces de red mediante llamada de bajo coste, hay muchas situaciones en las que su técnica de almacenamiento y reenvío se muestra demasiado inflexible, por ejemplo en Redes de Area Local (LANs, o RALs). Estas

[5] Teniendo en cuenta que hace tiempo que se escribió este libro, es seguro que son muchos más.

[6] De nuevo son datos no actualizados

[7] N. del T.: Del inglés *batches*

redes están compuestas generalmente por un pequeño número de máquinas localizadas en el mismo edificio, o incluso en la misma planta, que están interconectadas para proporcionar un entorno de trabajo homogeneo. Es típico que se quiera compartir ficheros entre estos nodos, o ejecutar aplicaciones distribuidas en diferentes máquinas.

Estas tareas requieren una aproximación completamente diferente a las redes. En lugar de reenviar ficheros completos con una descripción del trabajo, todos los datos se fragmentan en pequeñas unidades (paquetes), que se envían inmediatamente al nodo destino, donde son reensamblados. Este tipo de redes son llamadas *redes de intercambio de paquetes*. Entre otras cosas, esto permite ejecutar aplicaciones interactivas a través de la red. El coste de esto supone, por supuesto, una complejidad adicional al software.

La solución que han adoptado los sistemas UN⋆X — y muchos no-UN⋆X — es conocida como TCP/IP. En esta sección echaremos un vistazo a sus conceptos básicos.

1.3.1 Introducción a las Redes TCP/IP

El TCP/IP tiene sus orígenes en un proyecto de investigación fundado en Estados Unidos por el DARPA (Defense Advanced Research Projects Agency, Agencia de Proyectos Avanzados de Investigación en Defensa) en 1969. Ésta fue una red experimental, la red ARPANET, que paso a ser operativa en 1975, después de haber demostrado ser un éxito.

En 1983, fue adoptado como estándar el nuevo conjunto de protocolos TCP/IP, y todos los nodos de la red pasaron a utilizarlo. Cuando ARPANET por fin dio paso a Internet (con la propia ARPANET integrándose en su existencia en 1990), el uso del TCP/IP se había extendido a redes más alla de la propia Internet. Las más destacables son las redes locales UN⋆X, pero con la llegada de los equipos telefónicos digitales rápidos, como la RDSI, también tiene un futuro prometedor como transporte en redes telefónicas.

Para ilustrar las explicaciones que demos en las siguientes secciones, tomaremos como ejemplo una red típica: la de una universidad, concretamente la hipotética Universidad Groucho Marx (GMU) situada, por ejemplo, en algún lugar de Libertonia. En esta universidad, la mayoría de los departamentos mantienen sus propias redes de área local, mientras que algunos comparten una, y otros poseen varias. Todos ellos están interconectados, y están enganchados a Internet a través de un solo enlace de alta velocidad.

Supongamos una máquina Linux conectada a una LAN de nodos UN⋆X en el Departamento de Matemáticas, y su nombre es **erdos**. Para acceder a un nodo del Departamento de Físicas, por ejemplo **quark**, introducirá el siguiente comando:

```
$ rlogin quark.physics
Last login: Mon Feb  2 21:06:19 on tty1
```

```
Linux 2.0.0 #1 Sun Dec 7 19:07:05 MET 1997 (POSIX)
[...]
```

En la línea de comandos, introducirá su nombre de acceso, pongamos que es **andres**, y su clave. Entonces dispondrá de una shell de **quark**, sobre la que puede escribir como si estuviera sentado en la consola del sistema. Tras salir de la shell volverá a tener la línea de comandos de su propia máquina. Acaba de utilizar una de las aplicaciones de interactividad instantánea que proporciona TCP/IP: el acceso remoto.

Mientras esté conectado a **quark**, podría también desear ejecutar una aplicación X, como un programa de dibujo de funciones, o un visor de Postscript. Para indicar a esta aplicación que desea ver las ventanas en su monitor local, debe modificar la variable de entorno DISPLAY:

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

Si pone en marcha ahora su aplicación, esta contactará con su servidor X en lugar del de **quark**, y mostrará todas las ventanas en su monitor. Por supuesto, esto requiere que esté ejecutando X11 en **erdos**. La clave está en que TCP/IP permite a **quark** y a **erdos** enviarse paquetes X11 en ambos sentidos para darle a Ud. la impresión de que está en un único sistema. La red es casi transparente en este caso.

Otra aplicación muy importante en redes TCP/IP es NFS, abreviatura de Network File System (Sistema de Ficheros de Red). Es otra forma de hacer trasparente la red, porque básicamente permite montar jerarquías de directorios de otras máquinas, de modo que aparezcan como sistemas de ficheros locales. Por ejemplo, todos los directorios "home", o personales, de los usuarios pueden estar en una máquina servidor central, desde la cual montan los directorios el resto de máquinas de la LAN. El efecto de esto es que los usuarios pueden acceder a cualquier máquina, y encontrarse a sí mismos en el mismo directorio. De forma similar, es posible instalar aplicaciones que requieren gran cantidad de espacio en disco (tales como TEX) en una única máquina, y exportar estos directorios a otras máquinas. Volveremos sobre NFS en el capítulo 11.

Por supuesto, esto son sólo ejemplos de lo que se puede hacer en un entorno de redes TCP/IP: las posibilidades son casi ilimitadas.

Ahora echaremos una mirada más de cerca al modo en que trabaja TCP/IP. Esto es necesario para comprender cómo y por qué tiene que configurar su máquina. Comenzaremos examinando el hardware, y poco a poco recorreremos todo el camino.

1.3.2 Ethernets

El tipo de hardware más utilizado en LANs es lo que comunmente conocemos como *Ethernet*. Consta de un solo cable con los nodos colgando de él a través de conectores, clavijas o transceptores. Las ethernet simples, son baratas de instalar, lo que unido a un flujo de transferencia neto de 10 Megabits por segundo avala gran parte de su popularidad.

Hay tres tipos de Ethernet, en función de su cable, llamadas *gruesas*, *finas* y *de par trenzado*. Tanto el fino como el grueso utilizan cable coaxial, diferiendo en el grosor y el modo de conectar este cable a los nodos. El 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 Ethernet grueso requiere que realice un pequeño agujero en el cable, y conecte un transceptor utilizando un "conector vampiro". Entonces se pueden 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. El par trenzado usa un cable hecho de dos hilos de cobre como las que se encuentran en las instalaciones telefónicas ordinarias, pero generalmente necesitan hardware adicional. También se conoce como 10base-T.

A pesar de que añadir un nodo a una Ethernet gruesa es un poco lioso, eso no tirará abajo la red; sin embargo, para añadir un nodo en una instalación de cable fino, se debe interrumpir el servicio de red al menos por unos minutos ya que se debe cortar el cable para insertar el conector.

La mayoría de gente prefiere el Ethernet fino porque es barato: las tarjetas de PC pueden encontrarse por unos 50 dólares americanos (unas 5000 pesetas), o incluso menos, y el cable está por unos centavos el metro. Sin embargo, para instalaciones de gran escala, es más apropiado el Ethernet grueso. Por ejemplo, la Ethernet del Departamento de Matemáticas de la GMU utiliza Ethernet gruesa, de modo que no se interrumpe el tráfico cada vez que se añade un nodo a la red.

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 Ethernet entre sí utilizando repetidores, puentes o encaminadores[8]. 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 tiempos, no puede haber más 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 solo si el nodo receptor no está en la Ethernet local.

Ethernet funciona como un sistema de bus, donde un nodo puede mandar paquetes (o

[8] N. del T.: Respectivamente, *repeaters*, *bridges* y *routers*

tramas) de hasta 1500 bytes a otro nodo de la misma Ethernet. Cada nodo se direcciona por una dirección de seis bytes grabada en el firmware de su tarjeta Ethernet. Estas direcciones se especifican generalmente como una secuencia de números hexadecimales de dos dígitos separados por dos puntos, como en **aa:bb:cc:dd:ee:ff**.

Una trama enviada por una estación la ven todas las estaciones conectadas, pero sólo el nodo destinatario la toma y la procesa. Si dos estaciones intentan emitir al mismo tiempo, se produce una *colisión*, que se resuelve por parte de las dos estaciones abortando el envío, y reintentandolo al cabo de un rato.

1.3.3 Otros tipos de Hardware

En instalaciones mayores, como la Universidad de Groucho Marx, Ethernet no es el único tipo de red que puede utilizarse. En la Universidad de Groucho Marx cada LAN de un departamento está enlazada a la troncal del campus, que es 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 *testigos*, de modo que una estación sólo pueda enviar una trama si captura un testigo. La principal ventaja de FDDI es una velocidad de hasta 100 Mbps, y una longitud de cable máxima de hasta 200 km.

Para enlaces de red de larga distancia, se utiliza frecuentemente un tipo distinto de equipos, que se basa en el estándar X.25. Muchas de las llamadas Redes Públicas de Datos, como Tymnet en Estados Unidos, Datex-P en Alemania, o Iberpac en España, ofrecen este servicio. X.25 requiere un hardware especial, llamado Ensamblador/Desensamblador de Paquetes o *PAD*. X.25 define un conjunto de protocolos de red de derecho propio, pero sin embargo se usa frecuentemente para conectar redes bajo TCP/IP y otros protocolos. Ya que los paquetes IP no se pueden convertir de forma simple en X.25 (y viceversa), éstos deben ser encapsulados en paquetes X.25 y enviados a través de la red.

Frecuentemente, los radioaficionados usan sus propios equipos de radio para conectar sus ordenadores en red; esto se llama *packet radio* o *ham radio*. El protocolo utilizado por el packet radio es el llamado AX.25, que deriva del X.25.

Otras técnicas implican el uso de las lentas pero baratas líneas serie para acceder bajo demanda. Esto requiere aun otros protocolos para la transmisión de paquetes, como SLIP o PPP, que se describen más adelante.

1.3.4 El Protocolo IP (Internet Protocol)

Por supuesto, Ud. no querrá que su red esté limitada a una Ethernet. Idealmente, Ud. desearía poder acceder a la red sin importarle ni el hardware del que dispone ni el número de subestaciones. Por ejemplo, en instalaciones grandes como la Universidad de Groucho Marx, habrá varias Ethernets separadas, que han de conectarse 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 colgadas 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 Linux, el software de red no puede mandar paquetes a **quark** directamente, porque no está 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 Físicas, usando la troncal, y por fin **niels** los entrega a la máquina destino. El flujo de datos entre **erdos** y **quark** se muestra en la figura 1.1 (con disculpas a Guy L. Steele).

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

El principal beneficio del IP es que convierte a redes físicamente distintas en una red aparentemente homogénea. A esto se le llama *internetworking* (interconexión de redes), y a la resultante "meta-red" se la denomina *internet*. Observe aquí la sutil diferencia entre *una* internet y *La* Internet. El último es el nombre oficial de una internet global 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, que define su *dirección IP*. Una dirección IP se escribe normalmente como 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**. A este formato se le llama notación de puntos.

Se dará cuenta de que ahora tenemos tres tipos distintos de direcciones: primero, tenemos el nombre del nodo, **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 escriba *rlogin quark*, se le pueda pasar la direc-

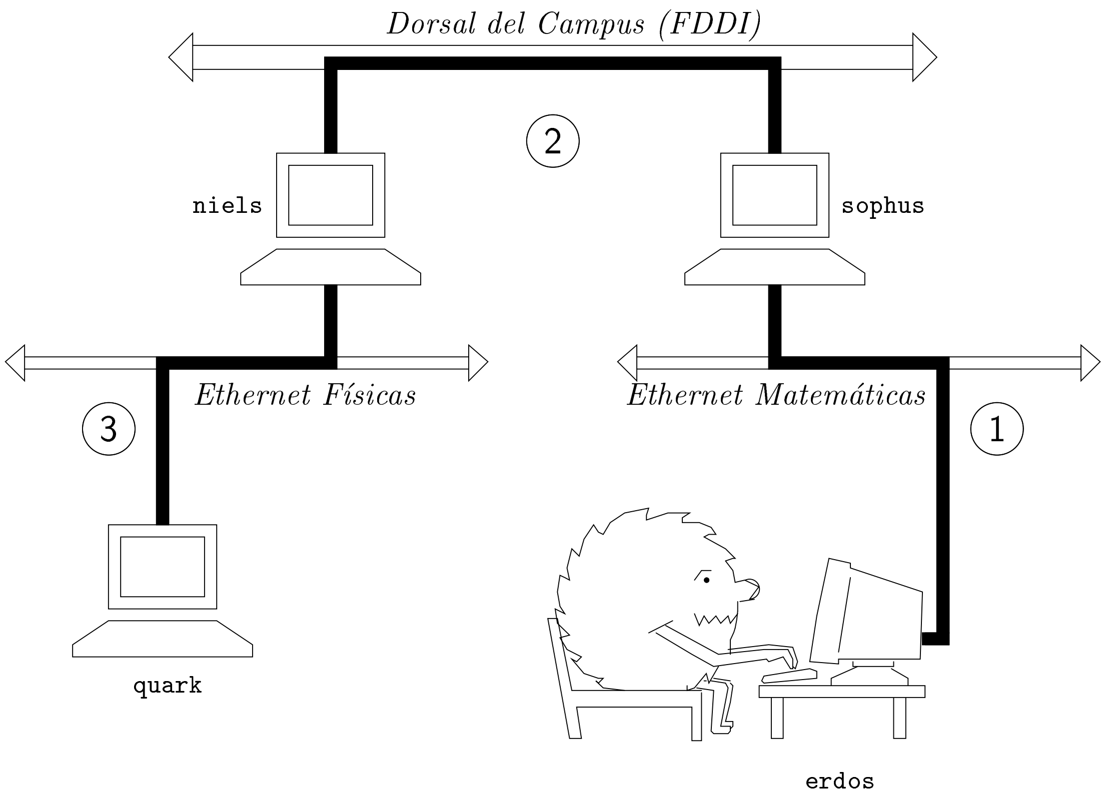

Figura 1.1: Los tres pasos para enviar un datagrama desde **erdos** a **quark**.

ción IP al software de red; y cuando el nivel IP envíe datos a la Ethernet del Departamento de Físicas, de algún modo tiene que encontrar a qué dirección Ethernet corresponde la dirección IP. Lo cual no resulta trivial.

No entraremos en esto aquí, sino que lo dejamos para el capítulo 2. De momento, es suficiente con indicar que estos pasos para encontrar las direcciones se llaman *resolución de nombres*, para mapear nombres de nodo con direcciones IP, y *resolución de direcciones*, para hacer corresponder estas últimas con direcciones hardware.

1.3.5 IP en Líneas Serie, SLIP

Para líneas serie se usa frecuentemente el estándar "de facto" conocido como SLIP o *Serial Line IP* (IP sobre línea 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.[9] Un protocolo serie distinto es el PPP, o *Point-to-Point Protocol (Protocolo Punto a Punto)*. PPP dispone de muchas más características que SLIP, incluyendo una fase de negociación del enlace. Su principal ventaja sobre SLIP es, sin embargo, que no se limita a transportar datagramas IP, sino que se diseñó para permitir la transmisión de cualquier tipo de datagramas.

1.3.6 El Protocolo de Control de Transmisión, TCP

Pero la historia no se acaba con el envío de datagramas de un nodo a otro. Si desea acceder a **quark**, necesita disponer de una conexión fiable entre su proceso *rlogin* en **erdos** y el proceso de shell en **quark**. Para ello, la información enviada en uno y otro sentido debe dividirse en 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 solo, no es fiable. Suponga que diez personas de su Ethernet comienzan a transferirse la última versión de XFree86 del servidor de FTP de GMU. La cantidad de tráfico generada por esto podría ser excesiva para que la maneje la pasarela, porque es demasiado lenta, y anda escasa de memoria. Si en ese momento Ud. envía un paquete a **quark**, **sophus** podría tener agotado el espacio del buffer durante un instante y por tanto no sería capaz de reenviarlo. IP resuelve este problema simplemente descartándolo. El paquete se pierde irrevocablemente. Lo cual traslada la responsabilidad de comprobar la integridad y exactitud de los datos a los nodos extremos, y su retransmisión en caso de error.

[9] SLIP está descrito en la norma RFC 1055. La compresión de cabeceras CSLIP, basada en él, se describe en la RFC 1144.

De esto se encarga otro protocolo, TCP, o *Transmission Control Protocol* (Protocolo de Control de la Transmisión), que construye un servicio fiable por encima de IP. La propiedad esencial de TCP es que usa IP para darle 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 qué ruta viajan realmente sus datos. Una conexión TCP funciona básicamente como una tubería de doble sentido en la que ambos procesos pueden escribir y leer; puede imaginarla como una conversación telefónica.

TCP identifica los extremos de tal conexión por las direcciones IP de los dos nodos implicados, y el número de los llamados *puertos* de cada nodo. Los puertos se pueden ver como puntos de enganche para conexiones de red. Si vamos a explotar el ejemplo del teléfono un poco más, uno 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).

En el ejemplo de *rlogin*, la aplicación cliente (*rlogin*) abre un puerto en **erdos**, y se conecta al puerto 513 de **quark**, en el que se sabe que está escuchando el servidor *rlogind*. Esto establece una conexión TCP. Usando esta conexión, *rlogind* realiza el procedimiento de autorización, y entonces muestra la shell. La entrada y salida estándar de la shell se redirigen a la conexión TCP, de modo que cualquier cosa que escriba a *rlogin* en su máquina será pasado a través del canal TCP y entregado a la shell como entrada estándar.

1.3.7 El Protocolo de Datagramas de Usuario, UDP

También es cierto que 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. Por contra, éste usa un protocolo derivado de TCP llamado UDP, o *User Datagram Protocol* (Protocolo de Datagramas de Usuario). De igual modo que TCP, UDP también 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, puede usarlo para enviar paquetes sueltos al servicio destino – de ahí su nombre.

Suponga que ha montado la jerarquía del directorio TEX del servidor de NFS central del departamento, *galois*, y desea ver un documento que describe como usar LATEX. Arranca su editor, y lee el fichero completo. Sin embargo, le llevaría demasiado tiempo establecer una conexión TCP con *galois*, enviar el fichero, y liberarla de nuevo. En cambio, se hace una petición a *galois*, que envía el fichero en un par de paquetes UDP, que es mucho más rápido. Sin embargo, UDP no se hizo para controlar la pérdida o corrupción de paquetes. Es responsabilidad de la aplicación – en este caso NFS – tener en cuenta esto.

1.3.8 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 a un puerto y espera a los clientes (a esto también se le llama *escuchar* en un puerto). Un cliente que quiera usar este servicio consigue un puerto libre en su nodo local, y se conecta al puerto del servidor en el nodo remoto.

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 al puerto servidor y esperar a más 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**, entonces el primer cliente *rlogin* usará el puerto local 1023, y el segundo usará el puerto número 1022; sin embargo, ambos se conectarán al mismo puerto 513 de **quark**.

Este ejemplo muestra el uso de puertos como puntos de encuentro, donde un cliente contacta con un puerto específico para obtener un servicio específico. Para que un cliente sepa el número de puerto adecuado, se ha tenido que llegar a un acuerdo entre los administradores de los dos sistemas para asignar estos números. Para servicios ampliamente usados, como *rlogin*, estos numeros tienen que administrarse centralmente. Esto lo realiza el IETF (o *Internet Engineering Task Force*), que regularmente publica un RFC (Request For Comment) denominado *Assigned Numbers* (Números Asignados). Describe, entre otras cosas, los números de puerto asignados a *servicios reconocidos*. Linux utiliza un fichero que mapea nombres con números, llamado */etc/services*. Se describe en la sección 9.3.

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).

1.3.9 La Librería de Sockets

En sistemas operativos UN⋆X, el software que realiza todas las tareas y protocolos descritos anteriormente es generalmente parte del kernel, y por tanto también del de Linux. El interface de programación más común en el mundo UN⋆X es la Librería de Socket de Berkeley, *Berkeley Socket Library*. Su nombre proviene de una analogía popular que ve los puertos como enchufes, y conectarse a un puerto como enchufarse. Proporciona la llamada *bind(2)* para especificar un nodo remoto, un protocolo de transporte, y un servicio al que un programa pueda conectarse o escuchar (usando *connect(2)*, *listen(2)*, y *accept(2)*). La librería de sockets, sin embargo, es algo más general, ya que proporciona no solo 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 X.25.

En Linux, la librería de sockets es parte de la librería C estándar *libc*. Actualmente sólo soporta los sockets *AF_INET* y *AF_UNIX*, pero se hacen esfuerzos para incorporar el soporte de los protocolos de red de Novell, de modo que se añadirían eventualmente una o más clases de sockets.

1.4 Redes con Linux

Siendo el resultado del esfuerzo concentrado de programadores de todo el mundo, Linux no habría sido posible sin la red global. 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 Linux; y fue en el otoño de 1992 cuando 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 1992 salió la primera versión pública de Net-2d (como parte del kernel 0.99.10), y ha sido mantenida y ampliada por varias personas, muy especialmente por Alan Cox, dando lugar al Net-2Debugged. Tras una dura corrección y numerosas mejoras en el código, cambió su nombre a Net-3 después de que saliese Linux 1.0. Esta es la versión del código de red que se incluye actualmente en las versiones oficiales del kernel.

Net-3 ofrece controladores de dispositivo para una amplia variedad de tarjetas Ethernet, así como SLIP (para enviar tráfico de red sobre líneas serie), y PLIP (para líneas paralelo). Con Net-3, Linux tiene una implementación de TCP/IP que se comporta muy bien en entornos de red de área local, mostrandose superior a algunos de los Unix comerciales para PCs. El desarrollo se mueve actualmente hacia la estabilidad necesaria para su funcionamiento fiable en nodos de Internet.

Además de estas facilidades, hay varios proyectos en marcha que mejorarán la versatilidad de Linux. Un controlador para PPP (el protocolo punto a punto, otra forma de enviar tráfico de red sobre líneas serie) está en estado Beta actualmente, y otro controlador AX.25 para *ham radio* está en estado Alfa. Alan Cox también ha implementado un controlador para el protocolo IPX de Novell, pero el esfuerzo para un paquete de red completo compatible con el de Novell se ha paralizado por el momento, debido a la negativa de Novell a facilitar la documentación necesaria. Otro proyecto muy prometedor es *samba*, un servidor de NetBIOS gratis para Unix, escrito por Andrew Tridgell.[10]

1.4.1 Diferentes Etapas de Desarrollo

Mientras tanto, Fred siguió desarrollando, continuando con el Net-2e, que dispone de un diseño más revisado de la capa de red. En el momento de escribir esto, Net-2e es aun software Beta. Lo más notable sobre Net-2e es la incorporación del DDI, el *Device Driver Interface* (Interface del controlador de dispositivo). DDI ofrece un acceso y un método de configuración uniforme a todos los dispositivos y protocolos de red.

Otra implementación más de red TCP/IP es la realizada por Matthias Urlichs, quien escribió un controlador de RDSI para Linux y FreeBSD. Para ello, integró algo del código de red de BSD en el kernel de Linux.

En un futuro previsible, sin embargo, Net-3 parece que llegará para quedarse. Alan trabaja actualmente en una implementación del protocolo AX.25 usado por radioaficionados. Sin duda, la *modularización*, aun por desarrollar para el kernel, traerá también nuevos impulsos al código de red. Los módulos le permiten añadir controladores al kernel en tiempo de ejecución.

Aunque todas estas diferentes implementaciones de red intentan dar el mismo servicio, hay grandes diferencias entre ellas a nivel de kernel y dispositivos. Además, no podrá configurar un sistema con un kernel Net-2e con utilidades de Net-2d o Net-3, y viceversa. Esto sólo se aplica a comandos que tienen mucho que ver con el funcionamiento interno del kernel; las aplicaciones y los comandos de red comunes como *rlogin* o *telnet* se ejecutan en cualquiera de ellos.

A pesar de todo, todas estas diferentes versiones de red no deben preocuparle. A no ser que esté participando en el desarrollo activo, no tendrá que preocuparse de qué versión del código TCP/IP está utilizando. Las versiones oficiales del kernel siempre estarán acompañadas de un conjunto de herramientas de red que son compatibles con el código de red presente en el propio kernel.

1.4.2 Donde Conseguir el Código

La última versión del código de red Linux se puede obtener mediante FTP anónimo de varios sitios. El servidor oficial del Net-3 es **sunacm.swan.ac.uk**, copiado en **sunsite.unc.edu** en el directorio *system/Network/sunacm*. El último parche para el Net-2e y los binarios

[10]NetBIOS es el protocolo en el que se basan las aplicaciones como *lanmanager* y Windows para Trabajo en Grupo.

se encuentran disponibles en **ftp.aris.com**. El código de red basado en BSD de Matthias Urlichs se puede conseguir en **ftp.ira.uka.de**, directorio */pub/system/linux/netbsd*.

Se pueden encontrar los últimos kernels en **nic.funet.fi**, en el directorio */pub/OS/Linux/PEOPLE/Linus*; Los nodos **sunsite** y **tsx-11.mit.edu** tienen copias de este directorio.

1.5 Mantenimiento del Sistema

En este libro, vamos a tratar principalmente los temas de instalación y configuración. Sin embargo la administración es mucho más importante — después de instalar un servicio, también hay que mantenerlo funcionando. Para la mayoría de ellos, sólo se necesitará una pequeña atención, mientras que algunos, como el correo y las news, requieren realizar tareas rutinarias para mantener actualizado el sistema. Discutiremos estas tareas en los capítulos finales.

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 shell y ejecutándolos periódicamente mediante el comando *cron*. La distribución fuente de algunas aplicaciones importantes como *smail* o C News, ya contiene esos scripts. Sólo tendrá que retocarlos para adecuarlos a sus necesidades y preferencias.

La salida de cualquiera de sus trabajos del *cron* se debería enviar a una cuenta de administración. Por defecto, muchas aplicaciones enviarán informes, estadísticas de uso, o resúmenes del fichero de registro a la cuenta de **root**. Esto sólo tiene sentido si accede como **root** frecuentemente; una idea mucho mejor es redirigir el correo del **root** a su cuenta personal estableciendo un alias de correo como se describe en el capítulo 14.

Por muy cuidadoso que sea configurando su máquina, la ley de Murphy garantiza que *surgirá* algún problema en cualquier momento. 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**), pero también hay otras direcciones que se usan con frecuencia para informar a la persona responsable de un aspecto concreto del mantenimiento. Por ejemplo, las quejas sobre una configuración de correo que funciona mal se dirigirán generalmente al **postmaster** (encargado del correo); y los problemas con el sistema de noticias pueden ser comunicados a **newsmaster** o **usenet**. El correo a **hostmaster** se debería redirigir a la persona encargada de los servicios básicos de red del nodo, y del servicio de nombres DNS si está corriendo 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 administrados sin ningún cuidado 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.

Esta sección comentará algunos ejemplos y técnicas básicas para pelearse con la seguridad del sistema. Por supuesto, los temas relatados 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. "Practical UNIX Security" de Simon Garfinkel (vease [Spaf93]) es una de las lecturas recomendadas.

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, comprobará su sistema de ficheros y ficheros de configuración comunes 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, y contienen tanto mayúsculas como minúsculas y números.

Cuando un servicio se hace accesible a la red, asegúrese de darle el "menor privilegio", lo que quiere decir que no se permita hacer cosas que no son imprescindibles para que trabaje como se diseñó. Por ejemplo, debería hacer sus programas con setuid **root** o alguna otra cuenta privilegiada sólo si realmente lo necesitan. También, si quiere usar un servicio sólo para una aplicación muy limitada, no dude en configuarla tan restrictivamente como su aplicación especial lo permita. Por ejemplo, si quiere permitir a máquinas sin disco arrancar desde su máquina, debe facilitar el TFTP (Trivial File Transfer Service) de modo que pueda 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, ¿por qué no restringir el servicio TFTP al directorio */boot*?[11]

Pensando en la misma línea, podría restringir ciertos servicios a usuarios que acceden desde ciertos nodos, digamos que sólo para su red local. En el capítulo 9, presentaremos *tcpd*, que hace esto para una variedad de aplicaciones de red.

[11] Volveremos sobre esto en el capítulo 9.

Otro punto importante es evitar software “peligroso”. Claro que cualquier software que utilice puede ser peligroso, porque el software puede tener fallos que algunos listos pueden explotar para acceder a su sistema. Cosas como ésta ocurren, y no hay protección segura contra ello. Este problema afecta al software libre y a productos comerciales por igual. [12] Sin embargo, programas que requieren privilegio especial son inherentemente más peligrosos que otros, ya que un agujero de estos puede tener consecuencias drásticas.[13] Si instala un programa setuid con propósitos de red, sea doblemente cuidadoso y no deje de leerse toda la documentación, de modo que no cree una brecha en la seguridad por accidente.

Nunca olvide que sus precauciones pueden fallar, por muy cuidadoso que haya sido. Por eso debería asegurarse de que detecta pronto a los intrusos. 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*[14] que permite comprobar ficheros vitales del sistema para ver si sus contenidos o permisos han cambiado. *tripwire* realiza varios checksums[15] fuertes sobre estos ficheros y los almacena en una base de datos. En las siguientes ejecuciones, se reevalúan y comparan los checksums con los almacenados para detectar cualquier modificación.

1.6 Vistazo a los Siguientes Capítulos

Los siguientes capítulos tendrán que ver con configurar Linux para redes TCP/IP, y con la ejecución de algunas aplicaciones importantes. Antes de que se manche las manos con la edición de ficheros y similares, examinaremos IP algo más de cerca en el capítulo 2. Si ya sabe como funciona el encaminamiento IP, y cómo se realiza la resolución de direcciones, quizá desee saltarse este capítulo.

El capítulo 3 trata los aspectos más básicos de configuración, como construir un kernel y configurar su tarjeta Ethernet. La configuración de los puertos serie se relata en un capítulo aparte, ya que la discusión no se aplica sólo a redes TCP/IP, sino que también es relevante para UUCP.

El capítulo 5 le ayudará a configurar su máquina para redes TCP/IP. Contiene pasos de instalación para nodos aislados (solamente con enlace local), y nodos conectados a una Ethernet. También le presentará unas pocas herramientas útiles que puede utilizar para

[12] Ha habido sistemas UN*X comerciales, por los que hay que pagar un montón de dinero, que venían con un script de shell con setuid-**root** que permitía a los usuarios obtener privilegios de **root** utilizando un conocido truco.

[13] En 1988, el gusano RTM llevo a gran parte de Internet a un colapso, en parte por explotar un agujero que había en algunos programas *sendmail*. Este agujero ya ha sido reparado con creces.

[14] Escrita por Gene Kim y Gene Spafford.

[15] N. del T.: Sumas de bytes con objeto de comprobar alguna modificación no “autorizada” en el fichero.

comprobar y retocar su configuración. El siguiente capítulo expone cómo configurar la resolución de nombres, y explica cómo montar un servidor de nombres.

Esto va seguido de dos capítulos que tratan de la configuración y uso de SLIP y PPP, respectivamente. El capítulo 7 explica cómo establecer conexiones SLIP, y da una referencia detallada de *dip*, una herramienta que le permite automatizar la mayoría de pasos necesarios. El capítulo 8 cubre PPP y *pppd*, el demonio que se necesita para ello.

El capítulo 9 da una corta introducción a la instalación de las aplicaciones de red más importantes, como *rlogin*, *rcp*, etc. Esto también abarca la gestión de los servicios por el *inetd*, y cómo puede restringir ciertos servicios de seguridad relevante a un grupo de nodos de confianza.

Los dos capítulos siguientes hablan de NIS, el sistema de información de red (Network Information System), y NFS, el sistema de ficheros de red (Network File System). NIS es una herramienta útil para distribuir información de administración, como las claves de usuario, en red de área local. NFS le permite compartir sistemas de ficheros entre varios nodos de su red.

El capítulo 12 le dará una amplia introducción a la administración del UUCP de Taylor, una implementación gratis del paquete UUCP.

El resto del libro será un viaje guiado por temas como el correo electrónico y las Noticias de Usenet. El capítulo 13 le conduce a los conceptos centrales del correo electrónico, como qué aspecto tiene una dirección de correo, y cómo se las arregla el sistema de manejo de correo para llevar su mensaje hasta el destinatario.

Los capítulos 14 y 15 cubren respectivamente la puesta en marcha de *smail* y *sendmail*, dos agentes de transporte de correo que puede utilizar en Linux. Este libro explica ambos, ya que *smail* es más fácil de instalar para un principiante, mientras que *sendmail* es más flexible.

Los capítulos 16 y 17 explican la forma en que se manejan las noticias en Usenet, y cómo instalar y usar C News, un paquete de software popular para la gestión de las noticias de Usenet. El capítulo 18 trata brevemente de cómo instalar un demonio NNTP para ofrecer el acceso a lectura de noticias para su red local. El capítulo 19 muestra finalmente cómo configurar y mantener varios lectores de noticias.

Capítulo 2

Cuestiones sobre redes TCP/IP

Vamos a entrar en los detalles que deben tenerse en cuenta cuando se conecta una máquina Linux a una red TCP/IP. De este modo, hablaremos de direcciones IP, nombres y cuestiones sobre el encaminamiento. Este capítulo le enseñará la base con la que podrá entender los pasos para su configuración particular, pasos que son cubiertos exhaustivamente en otros capítulos.

2.1 Interfaces de Red

Para ocultar la diversidad de hardware que puede usarse en una red, TCP/IP define una *interfaz* a través de la cual accedemos al hardware. Esta interfaz ofrece un conjunto de operaciones idénticas en cualquier tipo de hardware y que básicamente consisten en operaciones para enviar y recibir paquetes.

Para cada dispositivo que quiera utilizarse para conectarse a la red, se mantendrá en el núcleo del sistema la correspondiente interfaz. Por ejemplo, las interfaces con Ethernet en Linux son *eth0* y *eth1*, mientras que las interfaces SLIP se llaman *sl0*, *sl1*, etcétera. Estos nombres de interfaz se deben conocer durante la configuración de la red, cuando queramos referirnos a un dispositivo hardware concreto.

Para que podamos usarlo en una red TCP/IP, la interfaz deberá tener asignada una dirección IP que sirva como identificación de ésta ante los demás ordenadores de la red. Esta dirección es diferente del nombre de interfaz considerado anteriormente; puede realizarse la siguiente analogía: la interfaz sería la “puerta” de su sistema, mientras que la dirección vendría a ser un número enmarcado y colgado de la “puerta”.

Por supuesto, hay otros parámetros configurables para cada dispositivo, como el número máximo de datagramas que pueden ser procesados por el dispositivo, conocido como *Unidad*

Máxima de Transferencia o MTU[1]. Otros parámetros serán introducidos más tarde.

2.2 Direcciones IP

Como se dijo en el capítulo anterior, las direcciones utilizadas en el protocolo de red IP la forman números de 32 bits, y cada máquina debe tener una dirección propia. Si las máquinas se encuentran en una red TCP/IP que no se conecta a otras redes, dichas direcciones podrán asignarse a las máquinas librememente. Sin embargo, si las máquinas se conectan a Internet, las direcciones de los ordenadores serán asignadas por una autoridad principal, el NIC[2] o Centro de Información de la Red[3].

Para facilitar la lectura, las direcciones IP se dividen en cuatro números de 8 bits llamados *octetos*. Por ejemplo, si la máquina **quark.physics.groucho.edu** tiene una dirección IP **0x954C0C04**, normalmente la escribiremos con la notación de puntos divisorios, que separa los octetos, de esta forma: **149.76.12.4**.

Otra razón para usar esta notación es que las direcciones IP se pueden dividir en el número de *red* y el número de *nodo*. Cuando pedimos al NIC un conjunto de direcciones, este organismo nos concederá, no una dirección para nuestra máquina, sino un rango de direcciones válidas, que en realidad es un número de red concreto (el número de nodo lo pondremos nosotros, contando con todos esos nodos disponibles).

Dependiendo del tamaño de la red, la parte de la dirección correspondiente al nodo puede ser más o menos grande. Para adaptarse a diferentes necesidades, se conceden diferentes *clases* de redes, que definen diferentes maneras de dividir la dirección IP en parte de red y parte del nodo.

Clase A — La clase A comprende redes desde **1.0.0.0** hasta **127.0.0.0**. El número de red está en el primer octeto, con lo que solo hay 127 redes de este tipo, pero cada una tiene 24 bits disponibles para identificar a los nodos, lo que se corresponde con poder distinguir en la red unos 16 millones de nodos distintos.

Clase B — La clase B comprende redes desde **128.0.0.0** hasta **191.255.0.0**; siendo el número de red de 16 bits (los dos primeros octetos). Esto permite 16320 redes de 65024 nodos cada una.

[1]N. del T.: MTU son las siglas de "Maximum Transfer Unit".

[2]N. del T.: Network Information Center

[3]Frecuentemente, las direcciones IP le serán asignadas directamente por el proveedor que le conecte a la red. No obstante, se puede contactar directamente con el NIC para obtener direcciones, escribiendo a **hostmaster@internic.net**.

Clase C — Las redes de clase C tienen el rango de direcciones desde **192.0.0.0** hasta **223.255.255.0**, contando con tres octetos para identificar la red. Por lo tanto, hay cerca de 2 millones de redes de este tipo con un máximo de 254 nodos cada una.

Clases D, E, y F — Comprenden las direcciones entre **224.0.0.0** y **254.0.0.0**, y están reservadas para uso futuro, o con fines experimentales[4]. No especifican, pues, ninguna red de Internet.

Volviendo al ejemplo del capítulo anterior, veremos que en la dirección **149.76.12.4** de la máquina **quark**, **12.4** es el identificativo del nodo dentro de la red de clase B **149.76.0.0**.

Se puede observar que en la lista anterior no se consideraban todas las posibilidades en la parte que identifica al nodo; concretamente, se excluían siempre el identificador **0** y el **255**. Estos dos identificadores se reservan para propósitos especiales. Una dirección con los bits del nodo a cero identifica a la red, mientras que si tiene todos los bits a uno, identifica a todos los nodos de la red (lo que se conoce como dirección de *broadcast*, lo que indica que un mensaje enviado a esa dirección será procesado por todos los nodos de la red). Así pues, en nuestro ejemplo la dirección de la red sería **149.76.0.0** y la de *broadcast*, la **149.76.255.255**.

Además, otras dos direcciones de red están reservadas: la **0.0.0.0** y la **127.0.0.0**. La primera se conoce como dirección de *encaminamiento por defecto*, y la segunda, como *dirección de loopback*. El encaminamiento por defecto se utiliza para saber a dónde enviar los datagramas por defecto, tema que abordaremos después.

La red **127.0.0.0** se reserva para el tráfico local, dirigido al propio nodo. Normalmente, se asigna la dirección **127.0.0.1** a un dispositivo de la máquina llamado *interfaz de loopback*[5] o de circuito cerrado. Cualquier paquete enviado a esa dirección será recibido por el propio nodo. Esto permite probar aplicaciones de red con uno mismo, sin estar conectado a una red "real". Otra aplicación útil es la de ejecutar aplicaciones de red que afectan sólo al nodo local. Por ejemplo, muchos sistemas UUCP no tienen conexión IP pero ejecutan un sistema de noticias INN. Para que esto funcione, INN utiliza la interfaz de loopback.

2.3 Resolución de direcciones

Ahora que conocemos qué son las direcciones IP, nos preguntaremos cómo se utilizan en una red Ethernet. Después de todo, el protocolo Ethernet usa direcciones identificativas de

[4] Por ejemplo, se está utilizando la red 224.0.0.0 para la Internet multidestino o *multicast*

[5] N. del T.: También conocida como *interfaz de lazo*

seis octetos que no tienen que ver con los números IP.

En efecto, se necesita un mecanismo de traducción de direcciones IP a direcciones Ethernet o físicas. Esto se hace con el *Protocolo de Resolución de Direcciones* o ARP[6]. ARP no se limita a las redes Ethernet, sino que se extiende a otros tipos de redes como las de radio paquetes. La idea es la misma que tendríamos para localizar al señor X entre 150 personas: preguntar por su nombre a todo el mundo; y el señor X nos responderá.

Cuando queremos localizar la dirección física correspondiente a una dirección IP, haremos uso de una característica de la red Ethernet, que es la posibilidad de enviar mensajes a escuchar por todos los nodos, o mensajes *broadcast*. En el mensaje ARP, que es de este tipo, se incluye la dirección IP cuyo propietario estamos buscando. El nodo que posea esa dirección enviará una respuesta ARP al nodo llamante, con su dirección física.

Por supuesto, le preocupará saber cómo puede funcionar esto para localizar un nodo entre millones de Ethernets conectadas en el mundo. Esto se trata en la próxima sección: se trata del encaminamiento.

Sigamos hablando, de momento, sobre ARP. Una vez que se conoce la dirección física del nodo, el que hizo la petición guardará la información obtenida en una *cache ARP*, para así no preguntar por lo mismo cada vez que envíe un paquete a ese nodo. Sin embargo, no podemos guardar esa dirección para siempre ya que puede perder su validez (por ejemplo, si cambiamos la tarjeta de red a los nodos por avería, sus nuevas direcciones físicas serán distintas). Por ello, cada cierto tiempo, lo que hay en la cache ARP pierde su validez, obligando a realizar de nuevo la pregunta ARP.

A veces, un nodo necesita también conocer su dirección IP a partir de su dirección física. Por ejemplo, en terminales X o PCs sin disco, que cuando arrancan solo saben la dirección de su tarjeta pues está grabada en memoria no volátil. Para ello, se usa el *Protocolo de Resolución Inversa de Direcciones* o RARP[7]: la petición RARP la hace el nodo cuando arranca, mediante mensaje *broadcast*, y es contestado por un servidor de direcciones que, a partir de la dirección física, consulta su base de datos y conoce la dirección IP correspondiente. Existe además otro protocolo, el BOOTP o *protocolo de arranque*[8], que permite a las máquinas sin disco conocer cómo ponerse en marcha en la red.

[6] N. del T.: Address Resolution Protocol

[7] N. del T.: Reverse Address Resolution Protocol

[8] N. del T.: Del inglés *Boot Protocol*

2.4 Encaminamiento IP

2.4.1 Redes IP

◇ Cuando escribimos una carta a alguien, normalmente incluimos la dirección completa en el sobre: país, provincia, código postal, etc. De este modo el servicio de correos podrá llevar la carta a su destino: un servicio la enviará al del país que corresponda, y éste último la entregará al de la provincia o ciudad de destino. La ventaja de este esquema jerárquico es que el servicio postal del remitente apenas tiene que saber acerca del destino final, sino solo a qué país entregarla.

Las redes IP se organizan de manera similar. Internet consta de varias redes, conocidas como *sistemas autónomos* y cada una realiza por su cuenta el encaminamiento interno entre sus nodos miembro. Cuando un paquete tiene como destino un nodo de otra red, se entregará al encaminador correspondiente, sin preocuparse del destino final del paquete.

2.4.2 Subredes

La estructura de subredes se obtiene al dividir las direcciones IP en parte del nodo y parte de la red, como ya hemos explicado. Por defecto, la red de destino se deriva de la parte de red de la dirección IP. Es decir, los nodos con la misma dirección de red se encontrarán en la misma red TCP/IP[9].

Pero en una red puede interesar hacer una división en cientos de pequeñas redes, por ejemplo segmentos de Ethernet. Para ello se subdivide la red en *subredes*.

Una subred tiene la responsabilidad de la entrega de los datagramas a un determinado rango de direcciones IP de la red en la que se encuentra. Como sucede con las clases A, B o C, se identifica en la parte del número IP correspondiente a la red. Sin embargo, esa parte incluirá ahora algunos bits de la parte del nodo. Los bits que se interpretan como dirección de subred se obtienen con la llamada *máscara de red*. Es un número de 32 bits que especifica una máscara para identificar los bits de la subred.

La red del campus de la Universidad de Groucho Marx es un ejemplo de red de clase B, poseedora de la red **149.76.0.0**, con máscara **255.255.0.0**.

Internamente, la red de la UGM se divide en pequeñas subredes, como las LAN de cada departamento. Concretamente se divide en 254 subredes, desde la **149.76.1.0** hasta la **149.76.254.0**. Por ejemplo, el Departamento de Física Teórica tendrá asignada la subred **149.76.12.0**. La dorsal del campus es en sí mismo una subred, la **149.76.1.0**. Las subredes

[9]Los sistemas autónomos son algo más generales. Pueden tener más de una red IP.

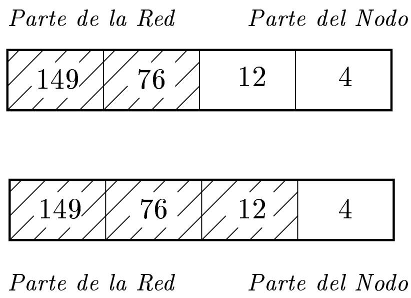

Figura 2.1: División de una red clase B en subredes

comparten el mísmo número de red, pero se usa el tercer octeto de ésta para distinguir las distintas subredes. Por lo tanto, tendrán una máscara de subred igual a **255.255.255.0**.

En la figura 2.1 se muestra cómo el nodo **quark** (**149.76.12.4**) se ve de distinta forma según se vea desde el punto de vista de la red de clase B, o desde el punto de vista de las subredes.

Nótese que la división en subredes es visible sólo internamente a la red. Normalmente las organiza el administrador de red para reflejar diferentes ubicaciones geográficas, distinguir segmentos de red, o bien por motivos administrativos (departamentos, redes de alumnos, etc). Pero esta división es totalmente invisible desde fuera de la organización.

2.4.3 Pasarelas

La organización en subredes no solo se hace por motivos administrativos, también es consecuencia de cuestiones del hardware. Lo que ve un nodo en una red es limitado: sólo ve los nodos con los que directamente esté conectado (por ejemplo, en la Ethernet), mientras que a los demás los accede a través de lo que se conoce como *pasarela*[10], que no es mas que un nodo conectado a dos o más redes físicas, configurado para pasar paquetes de una red a otra.

Para reconocer si una dirección IP se encuentra en la red local física, cada LAN debe tener una dirección de red IP diferente. Por ejemplo, las máquinas de la (sub)red **149.76.4.0** serían las que están en la LAN del Departamento de Matemáticas. Cuando se envía un datagrama a la máquina **quark**, el software de red de **erdos** ve que su dirección de red es otra (**149.76.12.4**) con lo que sabe que tiene que enviar los datagramas a través de la pasarela (**sophus** por defecto).

sophus se encuentra conectado a dos subredes: la del Departamento de Matemáticas

[10]N. del T.: Del inglés *gateway*

y la de la dorsal del campus, accediendo a cada una a través de una interfaz diferente, respectivamente *eth0* y *fddi0*. Nos preguntaremos entonces, qué dirección IP debe tener la pasarela, una de la subred de Matemáticas o bien una de la dorsal.

Pues bien, la respuesta es *ambas*. Cuando la pasarela comunique con un nodo de la LAN de Matemáticas, usará la dirección **149.76.4.1**, mientras que si lo hace con un nodo de la dorsal, usará **149.76.1.4**.

Es decir, la pasarela tiene tantas direcciones IP como conexiones a redes físicas tenga. En definitiva, éste será el esquema de interfaces, direcciones y máscara de **sophus**, nuestra pasarela:

interfaz	dirección	máscara
eth0	**149.76.4.1**	**255.255.255.0**
fddi0	**149.76.1.4**	**255.255.255.0**
lo	**127.0.0.1**	**255.0.0.0**

La última entrada describe el dispositivo *loopback*, que se comentó anteriormente.

En la figura 2.2 se muestra una parte de la topología de la red de la Universidad de Groucho Marx (UGM). Los nodos que están en dos subredes tendrán dos direcciones IP.

2.4.4 Tablas de Encaminamiento

Vamos ahora a centrarnos en cómo se selecciona una pasarela para entregar un datagrama a una red remota.

Hemos visto que **erdos**, cuando envía un datagrama para **quark**, comprueba que la dirección destino no se encuentra en la red local, por lo que lo envía a la pasarela, **sophus**, quien básicamente hace lo mismo: ve que **quark** no está en una de las redes a las que se conecta directamente y busca otra pasarela a quien entregarle el paquete. La elección correcta es **niels**, la pasarela del Departamento de Físicas. **sophus** necesita información para poder tomar estas decisiones.

La información de encaminamiento que se usa en IP es básicamente una tabla donde se relacionan (sub)redes y pasarelas. Además, debe incluirse una entrada de encaminamiento por defecto, que se asocia en la tabla a la red **0.0.0.0**. Todos los paquetes que van a una red desconocida, se enviarán a la pasarela del encaminamiento por defecto. Así pues, ésta sería la tabla para **sophus**:

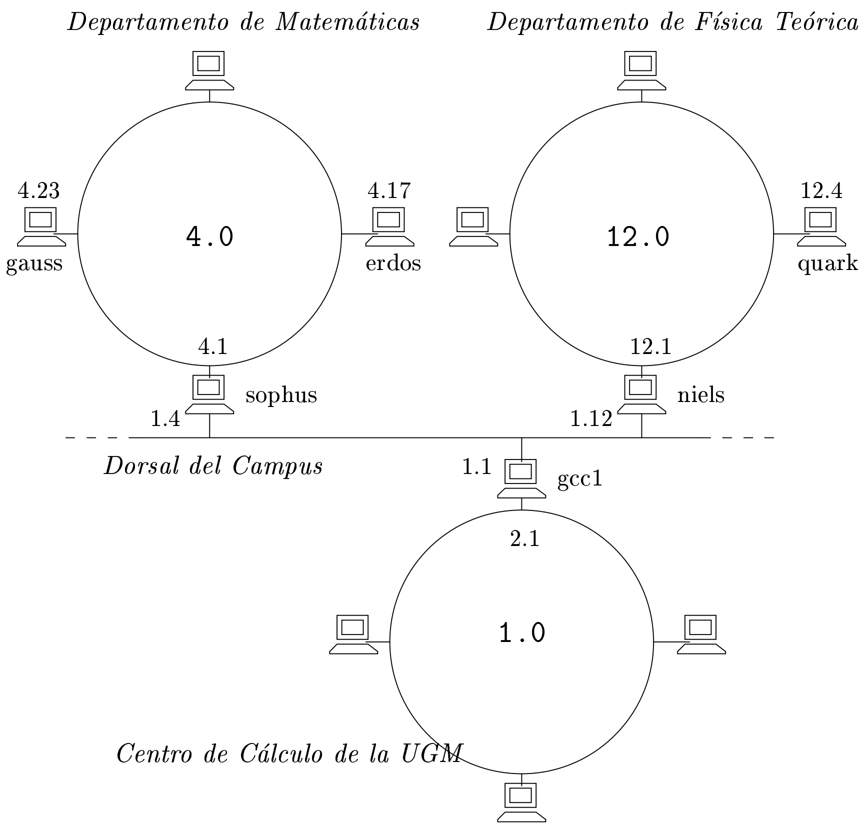

Figura 2.2: Vista parcial de la topología de la red de la UGM.

Red	Pasarela	Interfaz
149.76.1.0	-	**fddi0**
149.76.2.0	**149.76.1.2**	**fddi0**
149.76.3.0	**149.76.1.3**	**fddi0**
149.76.4.0	-	**eth0**
149.76.5.0	**149.76.1.5**	**fddi0**
...	...	...
0.0.0.0	**149.76.1.2**	**fddi0**

Las rutas a una red a la que **sophus** esté directamente conectado no necesitan pasarela, sino que los datagramas se entregan directamente. Esto se indica en la tabla anterior cuando en lugar de la pasarela aparece un "-".

Las tablas de encaminamiento pueden construirse de varias formas. Para redes pequeñas, será más eficiente construirlas a mano usando el comando *route* de Linux (véase el capítulo 5). Para redes más grandes, las tablas se mantienen y modifican automáticamente mediante los *demonios de encaminamiento*. Éstos corren en nodos centrales e intercambian información de encaminamiento entre ellos para tener en todo momento las rutas "óptimas" entre subredes.

Dependiendo del tamaño de la red, se utilizan distintos protocolos de encaminamiento. Dentro de los sistemas autónomos (como la Universidad Groucho Marx) se utilizan los protocolos *internos* o IGP[11]. El más utilizado es RIP[12] o *protocolo de información de encaminamiento*, que implementa el demonio *routed* de BSD. Para encaminamiento entre redes se usan protocolos EGP[13], como BGP. Se implementan en programas como *gated* de la Universidad de Cornell.[14].

2.4.5 Métricas de Encaminamiento

El encaminamiento dinámico basado en RIP elige la mejor ruta a un determinado nodo o red a partir del número de "saltos", es decir, las pasarelas que tiene que atravesar el datagrama hasta llegar a su destino. La ruta más corta será la elegida, y si hay 16 o más saltos se descartará por exceso de distancia.

Para usar RIP tiene que ejecutar *gated* en todas las máquinas. Al arrancar, *gated* comprueba cuántas interfaces están activas. Si hay más de una (sin contar la de *loopback*) asumirá que el nodo es una pasarela. Si no, entrará en modo pasivo, dedicándose a recibir cualquier actualización RIP y cambiando sus tablas en consecuencia.

[11]N. del T.: Internal Gateway Protocol

[12]N. del T.: Routing Information Protocol

[13]N del T.: External Gateway Protocol

[14]*gated* también implementa RIP y en general se recomienda usarlo en lugar de *routed*

Para enviar a las demás pasarelas la información de su tabla local de rutas, *gated* cuenta la longitud de cada una a partir de una *métrica* específica (que es decidida por el administrador del sistema y debe reflejar el coste de esa ruta). Así, la métrica de una ruta a una subred con conexión directa será siempre cero, mientras que una ruta que atraviese dos pasarelas deberá tener un coste de dos.

2.5 Protocolo de Mensajes de Control de Internet (ICMP)

IP tiene otro protocolo aún no mencionado. Es el *protocolo de mensajes de control de Internet* o ICMP[15], y lo usa el software de gestión de red para comunicar mensajes de error entre nodos. Por ejemplo, si estamos en la máquina **erdos** y hacemos un telnet al puerto 12345 del nodo **quark** y no hay procesos escuchando en ese puerto, recibirá un mensaje ICMP de "puerto inalcanzable".

Hay más mensajes ICMP, muchos de ellos referidos a condiciones de error. Sin embargo, hay uno interesante que es el de redirección. Lo genera el módulo de encaminamiento al detectar que otro nodo está usándolo como pasarela, a pesar de existir una ruta mucho más corta. Por ejemplo, tras configurarse la tabla de encaminamiento de **sophus**, ésta puede estar incompleta, conteniendo rutas a través del encaminador por defecto **gcc1**. Por lo tanto, los paquetes enviados inicialmente a **quark** irán por **gcc1** en lugar de **niels**. En este caso **gcc1** notificará a **sophus** que está usando una ruta costosa y reenviará el datagrama a **niels**, al mismo tiempo que devolverá un mensaje ICMP de redirección a **sophus** informándole de la nueva ruta.

Con lo visto, queda claro que se puede evitar tener que establecer las rutas a mano. Sin embargo, usar solo esquemas de encaminamiento dinámico no es siempre una buena idea. La redirección de ICMP y el protocolo RIP no incluyen mecanismos de verificación de la autenticidad de los mensajes. Esto permite a los piratas corromper el tráfico de la red mediante mensajes ICMP. Por ello, algunas versiones del código de Linux tratan los mensajes de redirección que afectan a rutas de red como si fueran redirecciones de rutas a nodos.

2.6 El sistema de nombres DNS

2.6.1 Resolución de nombres

◇ Como se comentó antes, el direccionamiento en TCP/IP se basa en números de 32 bits. Evidentemente, esos números no son fáciles de recordar, mientras que sí lo es el nombre que

[15] N. del T.: Internet Control Message Protocol

se le asigna a cada máquina, como **gauss** o **strange**. Existe una aplicación que es capaz de traducir nombres a direcciones IP, y es conocida como *sistema de resolución de nombres* o DNS[16].

Una aplicación que desee encontrar la dirección IP correspondiente a una máquina de la que conoce su nombre, no tiene que incluir rutinas para ello, ya que en las librerías estándares (*libc*) existen ya rutinas preparadas, como *gethostbyname(3)* o *gethostbyaddr(3)*. En otros sistemas se encuentran en otras librerías distintas de la *libc* pero esto no sucede en Linux. Al conjunto de rutinas que hacen estas tareas se les conoce como "sistema de resolución".

En una red pequeña no es difícil mantener una tabla */etc/hosts* en cada máquina, y modificarla al agregar, eliminar o modificar nodos. Pero resulta complicado cuando hay muchas máquinas ya que, en principio, cada una necesita una copia de */etc/hosts*.

Una solución a esto es compartir ésta y otras bases de datos con el NIS, o *sistema de información de red*[17], desarrollado por Sun Microsystems y conocido también como *páginas amarillas*. En este caso, las bases de datos como la de */etc/hosts* se mantienen en un servidor NIS central y los clientes accederán a ellas de forma transparente al usuario. En todo caso, esta solución solo es aconsejable para redes pequeñas o medianas, ya que implican mantener un fichero central */etc/hosts* que puede crecer mucho, y luego distribuirlo entre los servidores NIS.

En Internet, se comenzó almacenando la información en un fichero similar al *hosts*, mantenido por el NIC, y obtenido regularmente por los demás servidores. Cuando la red creció comenzaron los problemas de sobrecarga de servidores, además de que el NIC tenía que ocuparse de todos los nombres de los nodos de Internet, y evitar la duplicidad de los mismos.

Por esto, en 1984 se diseñó y adoptó oficialmente un nuevo sistema, el DNS o *sistema de nombres por dominios*, diseñado por Paul Mockapetris.

2.6.2 Introducción al DNS

DNS organiza los nombres de los nodos en una jerarquía de dominios. Un dominio es una colección de nodos relacionados de alguna manera, como estar en la misma red o pertenecer a una misma organización o país. Por ejemplo, las universidades norteamericanas se agrupan en el dominio **edu**, y cada universidad mantiene un subdominio dentro de **edu**. Nuestro ejemplo, la Universidad de Groucho Marx, mantendría el dominio **gmu.edu** y las máquinas del departamento de Matemáticas se encontrarían dentro del dominio **maths.gmu.edu**.

[16]N. del T.: Domain Name System
[17]N. del T.: Network Information System

De este modo el nombre completo de la máquina **erdos** será **erdos.maths.gmu.edu**. El nombre completo se conoce como *nombre totalmente cualificado* o FQDN[18], e identifica a ese nodo en todo el mundo.

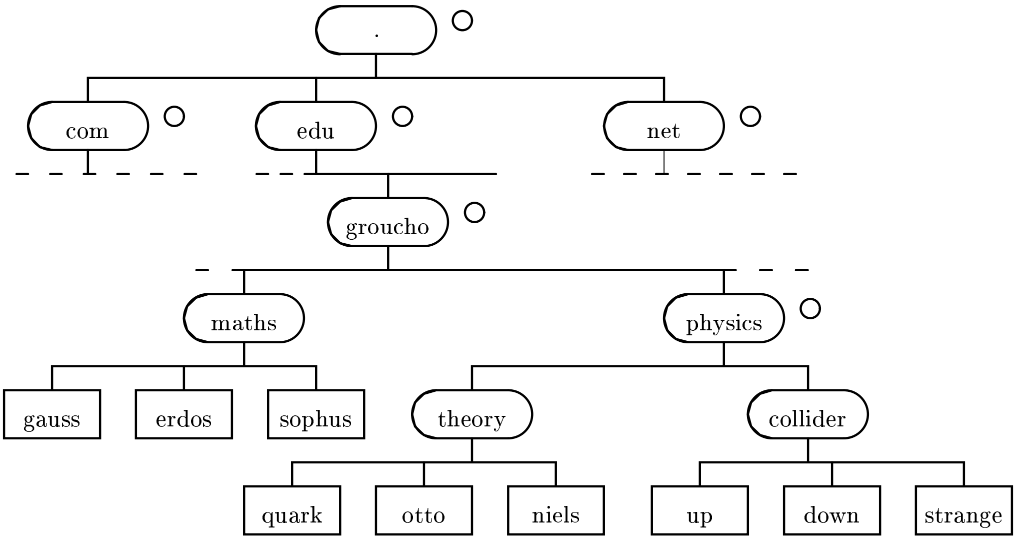

Figura 2.3: Parte del espacio de dominios

En la figura 2.3 se muestra una sección del espacio de dominios. La entrada de la raíz del árbol, que se indica con un punto, se puede denominar *dominio raíz*, y agrupa al resto de los dominios. Para indicar que un nodo se expresa en notación FQDN, se puede terminar el nombre en un punto, indicando así que el nombre incluye al del dominio raíz.

Dependiendo de su localización en la jerarquía, un dominio puede ser de primer, segundo o tercer nivel. Pueden existir otros niveles pero no son frecuentes. Por ejemplo, algunos dominios de primer nivel muy usuales son los siguientes:

edu Aquí se incluyen casi todas las universidades o centros de investigación norteamericanos.

com Compañías u organizaciones con fines comerciales.

org Organizaciones no comerciales. Las redes UUCP privadas se encuentran aquí.

net Pasarelas y otros nodos administrativos de la red.

mil Nodos militares norteamericanos.

[18]N. del T.: Fully Qualified Domain Name

gov Nodos del gobierno norteamericano.

uucp Oficialmente, todos los nombres de nodos UUCP sin dominio han sido movidos a este nuevo dominio, .uucp.

En general, los dominios anteriores pertenecen a redes norteamericanas. Algo especialmente cierto con los dominios **mil** o **gov**.

Fuera de los Estados Unidos, existe un dominio de primer nivel para cada país, de dos letras según se define en la norma ISO-3166. Finlandia, por ejemplo, usa el dominio **fi**; el dominio **de** corresponde a Alemania y el dominio **es** corresponde a España. Cada país organiza por debajo del primer nivel, los dominios de segundo nivel, de manera parecida a los americanos en algunos casos (por ejemplo, con dominios **com.au** o **edu.au**) o directamente por organizaciones, como sucede en España (con dominios como **upm.es** para la Universidad Politécnica de Madrid).

Por supuesto, un nodo dentro del dominio de un país puede no estar físicamente en él. El dominio únicamente identifica al nodo como registrado en el NIC de ese país. Así, un comerciante sueco puede tener una delegación en Australia, y tener sus nodos australianos registrados dentro del dominio de primer nivel sueco, **se**.

Esta organización por dominios soluciona el problema de la unicidad de nombres. Además, los nombres totalmente cualificados no son difíciles de recordar.

Pero DNS tiene otras ventajas: permite delegar la autoridad sobre un determinado subdominio a sus administradores. Por ejemplo, los subdominios **maths** y **physics** de la UGM son creados y mantenidos por el Centro de Cálculo de dicha universidad. Y si el mantenimiento del subdominio **maths.gmu.edu** fuese complicado (por número elevado de nodos, existencia de subdominios internos, etc), el Centro de Cálculo de la UGM puede delegar la autoridad sobre ese subdominio al departamento de Matemáticas. La delegación de un subdominio implica el control total del mismo por parte de la organización en la que se delegó, con total libertad para crear nuevos subdominios internos, asociar nombres a nodos, etc.

Para este fin, el espacio de nombres se divide en *zonas*, cada una asociada a un dominio. Nótese que existe una diferencia entre zona y dominio: el *dominio* **groucho.edu** incluye todos los nodos de la UGM, mientras que la *zona* **groucho.edu** incluye sólo los nodos que mantiene directamente el Centro de Cálculo, ya que los nodos del subdominio **physics.groucho.edu** pertenecen a la zona controlada por el Departamento de Físicas. En la figura 2.3 se marca el inicio de una zona con un pequeño círculo a la derecha del nombre del dominio.

2.6.3 Búsquedas de nombres con DNS

Trataremos aquí el problema de cómo resolver el nombre de un determinado nodo.

DNS es una gigantesca base de datos distribuida. Se implementa a través de los llamados *servidores de nombres.* Cada uno de éstos mantiene la información de uno o varios dominios. Para cada zona hay al menos dos (o más) servidores de nombres que mantienen información *autorizada* sobre los nodos de esa zona. Para obtener la dirección IP del nodo **erdos**, lo que hay que hacer es contactar con el servidor de nombres de la zona para **groucho.edu** y éste nos devolverá los datos pedidos.

Esto parece fácil de decir pero difícil de implementar pues nos preguntaremos cómo localizar al servidor de nombres de la UGM. Si su ordenador no implementa un adivino, le ayudará el DNS. Cuando su aplicación desea encontrar información acerca de **erdos**, contactará en primer lugar con un servidor de nombres local, quien realizará una búsqueda por otros servidores. Empieza por preguntar a un servidor de nombres raíz por **erdos.maths.groucho.edu**. Al comprobar éste último que él no mantiene ese dominio, contactará con los servidores del dominio **edu** y les preguntará las direcciones de los servidores de nombres, que retornará al servidor local. Ahora nuestro servidor preguntará a estos últimos y éstos a su vez irán haciendo llegar a nuestro servidor hasta los que mantienen la zona **groucho.edu**. Finalmente, se preguntará a uno de estos últimos por el nodo **erdos** y se enviará la respuesta al usuario.

Aparentemente esto provoca mucho tráfico, aunque en todo caso siempre será menor que preguntar siempre a los mismos servidores que mantenían el fichero *HOSTS.TXT* antes de que se diseñara el DNS.

Sin embargo, aún se puede mejorar algo más. La información obtenida en una búsqueda puede que se necesite después. Por ello, el servidor de nombres local la guardará en una *cache* local. Así, cuando volvamos a preguntar por un nodo de **groucho.edu**, el servidor local ya podrá dirigirse directamente el servidor de nombres de esa zona sin pasar por los servidores raíz.

Por supuesto, el servidor de nombres no puede mantener la cache eternamente; debe descartarla cada cierto tiempo. Este tiempo de expiración se conoce como TTL[19] o *tiempo de vida.* En la base de datos del DNS queda especificado este parámetro.

2.6.4 Servidores de Nombres

Cuando un servidor de nombres mantiene toda la información acerca de una zona se le llama *autorizado* para esa zona. Cualquier petición para esa zona será enviada a uno de

[19] N. del T.: Time To Live

esos servidores maestros.

Para tener una representación coherente de la zona, sus servidores maestros deben estar sincronizados. Para ello, a uno de ellos se le nombra servidor *primario*, que obtiene la información de zona a partir de unos ficheros locales, y a los demás se les nombra servidores *secundarios*. Estos últimos cargan la información de la zona pidiéndosela al primario cada cierto tiempo.

Las razones para que existan varios servidores autorizados por cada zona son dos: repartir la carga de trabajo y lograr tolerancia a fallos. Así, si un servidor cae, todas las peticiones se repartirán entre los demás servidores autorizados que haya. Por supuesto, esto no protege contra fallos internos o *bugs* del propio software DNS.

Pero además, también es posible tener servidores de nombres que no mantengan información autorizada de ningún dominio[20]. Este tipo de servidores es útil pues, al mantener una *cache* con los nombres que resuelven, disminuye la carga de la red y de otros servidores.

2.6.5 La Base de Datos DNS

En las bases de datos del DNS se mantiene más información que la necesaria para traducir nombres a direcciones IP. Dicho de otra forma, en DNS se mantienen *distintos tipos de registros*.

La unidad de información en el DNS se conoce como *registro de recurso* o RR. Cada registro tiene un tipo asociado a él, describiendo qué clase de datos contiene, y una clase indicando el tipo de red al que se aplica. Se trata de acomodarse a diferentes esquemas de red, aunque para direcciones IP se usa siempre la clase IN (INternet), pero hay otras como las redes Hesiod (que se usan en el MIT[21]). El registro más habitual es el de tipo A, que relaciona un nombre totalmente cualificado con una dirección IP.

Un nodo puede admitir más de un nombre. Pero sólo uno de ellos será "oficial" o *canónico*, mientras que los demás son alias del primero. La diferencia es que el canónico se define en un registro de tipo A, mientras que los alias se definen en registros CNAME que apuntan al nombre canónico.

En un capítulo posterior se trata todo esto en profundidad. Aquí nos vamos a limitar a ver algunos ejemplos. En la figura 2.4 se muestra una parte de la base de datos para la zona **physics.groucho.edu**.

Además de los registros A y CNAME, se puede ver que hay un registro especial al

[20] Aunque al menos deberá tener información autorizada para el **localhost** y la resolución inversa de **127.0.0.1**

[21] Instituto Tecnológico de Massachusets

```
;
; Informacion Autorizada de physics.groucho.edu
@                   IN    SOA         (
     niels.physics.groucho.edu.
     hostmaster.niels.physics.groucho.edu.
     1034            ; serial no
     360000          ; refresh
     3600            ; retry
     3600000         ; expire
     3600            ; default ttl )
;
; Servidores de nombres autorizados
      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 Collider (subred 14)
boson               IN    A        149.76.14.1
muon                IN    A        149.76.14.7
bogon               IN    A        149.76.14.12
...
```

Figura 2.4: Extracto del fichero *named.hosts* del departamento de Físicas.

principio del fichero, con varias líneas. Se trata del registro SOA o de *inicio de autoridad*, que mantiene información general sobre el servidor de nombres. Por ejemplo, el tiempo de vida por defecto de todos los registros que mantiene.

Nótese que aquellos nombres que no finalicen en un punto serán interpretados como relativos al dominio en cuestión. El nombre especial "@" usado en el registro SOA representa al dominio completo.

Hemos visto que los servidores para el dominio **groucho.edu** deben tener conocimiento sobre los servidores de la zona **physics** para poder reenviarles las peticiones para ésta. Esto se suele incluir en los registros NS que incluyen el nombre de los servidores en notación FQDN, y un registro A que da la dirección IP para ese servidor. Véase, por ejemplo, la figura 2.5.

```
;
; Datos de zona para groucho.edu.
@                  IN      SOA        (
     vax12.gcc.groucho.edu.
     hostmaster.vax12.gcc.groucho.edu.
     233               ; serial no
     360000            ; refresh
     3600              ; retry
     3600000           ; expire
     3600              ; default ttl )

....
;
; Registros de la zona.
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
...
```

Figura 2.5: Extracto del fichero *named.hosts* de la UGM.

2.6.6 Resolución inversa

Además de la obtención de una dirección IP a partir del nombre, a veces interesa lo contrario: conocida la dirección, obtener el nombre canónico. Esto se conoce como *traducción inversa* y la utilizan algunas aplicaciones de red para verificar la identidad del llamante. Cuando se usa un fichero *hosts*, la resolución inversa supone una simple búsqueda en el mismo. En cambio, para el DNS se ha creado un dominio especial, el **in-addr.arpa**, que contiene direcciones de los nodos en notación de puntos divisorios invertida. Por ejemplo, a la dirección IP **149.76.12.4** le corresponde el nombre **4.12.76.149.in-addr.arpa**. El tipo de registro para estos datos se llama PTR.

Cuando se crea una zona de autoridad suele significar que sus administradores tienen control total sobre cómo asignan sus nombres. Pero a una subred se le puede delegar un subdominio.

Esto sucede con la UGM, donde se delega la zona de Físicas (subdominio **physics**) al correspondiente Departamento. Las direcciones de resolución inversa también se delegan.

En la figura 2.6 se muestra el contenido del fichero de zona para el servidor de la subred 12. Los registros "importantes" de delegación se muestran en la figura 2.7.

```
;
; Dominio 12.76.149.in-addr.arpa
@               IN     SOA   (
     niels.physics.groucho.edu.
     hostmaster.niels.physics.groucho.edu.
     233 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.
```

Figura 2.6: Extracto del fichero *named.rev* de la subred 12.

Una importante consecuencia de esto es que las zonas sólo pueden crearse como superconjuntos de redes IP, y además, las máscaras de red deberán redondearse a nivel de octeto. Es decir, las subredes de la UGM tendrán una máscara **255.255.255.0**, y para cada subred deberá existir una zona **in-addr.arpa**. Sin embargo, si la máscara fuese **255.255.255.128**, la creación de zonas para la subred **149.76.12.128** sería imposible ya que no hay forma de decir en DNS que el dominio **12.76.149.in-addr.arpa** ha sido dividido en dos zonas de autoridad, con nombres de nodos desde el **1** al **127**, y desde el **128** al **255**, respectivamente.

```
;
; Dominio 76.149.in-addr.arpa domain
@                   IN       SOA          (
     vax12.gcc.groucho.edu.
     hostmaster.vax12.gcc.groucho.edu.
     233 360000 3600 3600000 3600 )

...
; subred 4: Departamento de Matematicas
1.4              IN     PTR      sophus.maths.groucho.edu.
17.4             IN     PTR      erdos.maths.groucho.edu.
23.4             IN     PTR      gauss.maths.groucho.edu.
...
; subred 12: Departamento de fisicas, zona separada
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
...
```

Figura 2.7: Extracto del fichero *named.rev* de la red **149.76**.

Capítulo 3

Configuración del Hardware de Red

3.1 Dispositivos, Controladores, y todo lo demás

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

Primero, evidentemente, está el hardware por sí mismo; por ejemplo, una tarjeta Ethernet es: una oblea de Silicio, atiborrada de montones de pequeños chips con estúpidos números en el lomo e insertada en una ranura de su PC. Esto es lo que por lo general denominamos un dispositivo.

Para poder utilizar la tarjeta Ethernet son necesarias una serie de funciones especiales definidas en el núcleo de Linux que serán capaces de entender la forma particular de acceso al dispositivo. Esta serie de funciones son los denominados controladores[1] del dispositivo. Por ejemplo, Linux tiene controladores de dispositivos para varias marcas de tarjetas Ethernet que son muy parecidas en su funcionamiento. Son conocidos como los "Controladores de la Serie Becker", debido a su autor, Donald Becker. Otro ejemplo puede ser el del controlador D-Link, que gestiona un adaptador de bolsillo D-Link conectado a un puerto paralelo.

Pero ¿qué es lo que queremos decir con que un controlador "gestione" un dispositivo? Volvamos a la tarjeta Ethernet que examinamos antes. 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

[1] N. del T.: Con frecuencia, la bibliografía especializada en español también los llama *manejadores*

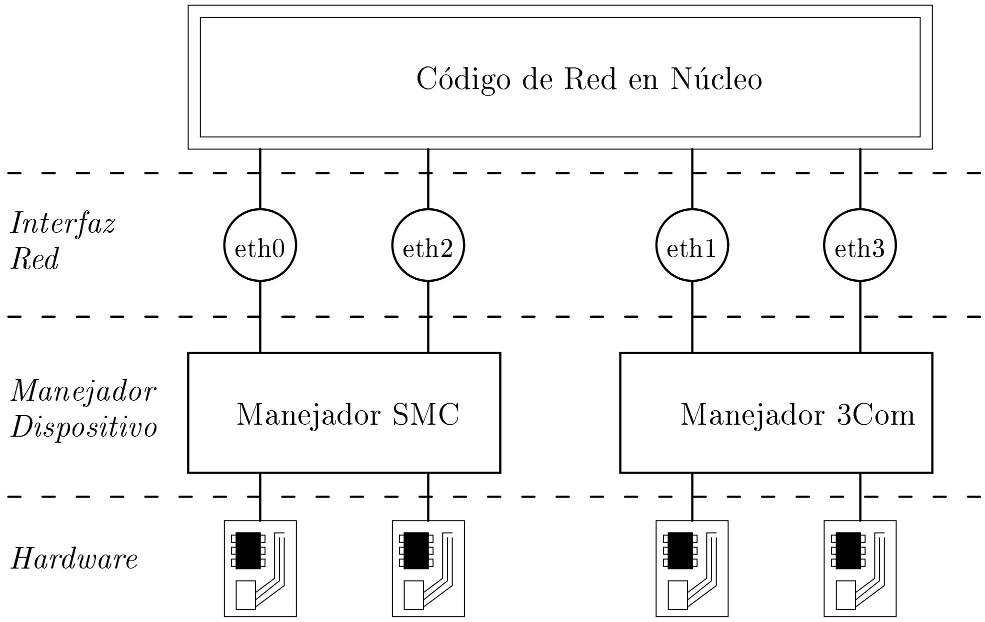

Figura 3.1: Relación entre controladores, interfaces, y hardware.

recibido.

En un PC, esta comunicación tiene lugar a través de un área de memoria de E/S que se corresponde con los registros internos de la tarjeta y a la inversa. Todas las órdenes y datos que el núcleo envía a la tarjeta tienen que ir a través de estos registros. El área de memoria de E/S de la tarjeta se describe por lo general mediante su *dirección base* o de comienzo. Direcciones habituales para tarjetas Ethernet son la `0x300` o la `0x360`.

Normalmente no hay que preocuparse por factores de hardware como las direcciones base, ya que en tiempo de arranque el núcleo hace un intento para detectar la localización de la tarjeta. Esto se denomina *autoverificación*[2], que implica que el núcleo lea varias posiciones de memoria y compare los datos leídos con los que debería haber si existiese una tarjeta Ethernet instalada allí. De todas maneras, puede haber tarjetas Ethernet que no sean detectadas automáticamente; esto ocurre a veces con tarjetas Ethernet baratas que no son réplicas exactas de tarjetas estándar de otros fabricantes. Por otro lado, el núcleo intentará detectar un único dispositivo Ethernet al arrancar. Si se utiliza más de una tarjeta, hay que indicárselo explícitamente al núcleo.

Otro de los parámetros que se pueden decir de forma explícita al núcleo es el canal de petición de interrupción. Los componentes hardware normalmente interrumpen al núcleo cuando tienen necesidad de que éste se ocupe de ellos, por ejemplo cuando han llegado datos,

[2]N. del T.: Del inglés *autoprobing*

o se presenta una condición especial. En un PC, las interrupciones pueden comunicarse mediante uno de los 15 canales de interrupción numerados 0,1 y del 3 al 15. El número de interrupción asignado a un componente hardware se denomina su *número de petición de interrupción*, o IRQ[3][4].

Como se explicaba en el capítulo 2, el núcleo accede a un dispositivo mediante lo que llamábamos una interfaz. Las interfaces ofrecen un conjunto abstracto de funciones que es el mismo para todo tipo de hardware. Por ejemplo, con funciones para enviar o recibir datagramas.

Los interfaces se identifican mediante nombres. Estos nombres se definen internamente en el núcleo, y no son ficheros de dispositivos del directorio */dev*. Nombres típicos para los interfaces Ethernet son *eth0*, *eth1*, etc. La asignación de interfaces a los dispositivos depende normalmente del orden en el que los dispositivos son configurados; por ejemplo la primera tarjeta Ethernet instalada será *eth0*, la siguiente *eth1*, y así sucesivamente. Una excepción a esta regla son las interfaces SLIP y algunas otras, que son asignadas de forma dinámica; es decir, al establecerse una conexión SLIP, se asigna una interface al puerto serie.

El gráfico de la figura 3.1 muestra la relación entre el hardware, los controladores de dispositivos e interfaces.

Al arrancar, el núcleo muestra cada dispositivo que es detectado, y qué interfaces se están instalando. Lo siguiente es un extracto de la pantalla de arranque:

```
.
.
This processor honours the WP bit even  when in supervisor mode. Good.
Floppy drive(s): fd0 is 1.44M
Swansea University Computer Society NET3.010
IP Protocols: ICMP, UDP, TCP
PPP: version 0.2.1 (4 channels) OPTIMIZE FLAGS
TCP compresssion code copyright 1989 Regents of the University of California
PPP line discipline registered.
SLIP: version 0.7.5 (4 channels)
CSLIP: code copyright 1989 Regents of the University of California
dl0: D-Link DE-600 pocket adapter, Ethernet Address: 00:80:C8:71:76:95
Checking 386/387 coupling... Ok, fpu using exception 16 error reporting.
Linux version 1.1.11 (okir@monad) #3 Sat May 7 14:57:18 MET DST 1994
```

Ésta indica que el núcleo ha sido compilado para TCP/IP, y se incluyen los controladores para SLIP, CSLIP, y PPP. La tercera línea antes del final indica que se ha detectado un

[3] Los IRQs 2 y 9 son los mismos debido a que el PC tiene dos procesadores de interrupciones en cascada con 8 IRQs cada uno; el procesador secundario está conectado al IRQ 2 del primario.

[4] N. del T.: Del inglés *Interrupt ReQuest*

adaptador de bolsillo D-Link e instalado como el interface *dl0*. Si usted tiene un tipo diferente de tarjeta Ethernet, el núcleo mostrará normalmente una línea comenzando por *eth0*, seguida por el tipo de tarjeta detectado. Si usted tiene una tarjeta Ethernet instalada pero no se refleja en ningún mensaje, significa que el núcleo es incapaz de detectar su tarjeta adecuadamente. Trataremos este problema posteriormente.

3.2 Configuración del núcleo

La mayoría de las versiones de Linux se distribuyen con discos de arranque que funcionan en casi cualquier PC. Esto significa que el núcleo tiene en esos discos todo tipo de controladores configurados que rara vez utilizará, pero que ocupan un espacio innecesario en el sistema de memoria ya que con el núcleo no puede hacerse *swapping*. Por tanto sería conveniente crear un núcleo propio, incluyendo sólo los controladores que realmente necesite o desee.

Al trabajar con un sistema Linux, le deberá resultar familiar el proceso de construcción del núcleo. Los conceptos básicos de cómo realizarlo se explican en la Guía de Matt Welsh: "Instalación y primeros pasos", que también forma parte de la serie de Documentación del Proyecto Linux. Por tanto, en esta sección solo trataremos las opciones de configuración que afectan a la red.

Al ejecutar `make config`, se le preguntará por una serie de configuraciónes generales, por ejemplo si desea emulación matemática del núcleo o no, etc. Una de las preguntas será si desea o no soporte para red TCP/IP. Si desea un núcleo capaz de trabajar con la red debe ser contestada con `y` (SÍ).

3.2.1 Opciones del núcleo de Linux 1.0 o Versiones Posteriores

Tras completar la parte de opciones generales, se pasará a configurar distintos componentes: controladores SCSI, etc. La siguiente lista contiene las preguntas que son sobre el soporte de red. Nótese que el conjunto de opciones de configuración está en constante cambio debido al continuo desarrollo. Una lista típica de opciones que ofrecen los núcleos de versiones entre la 1.0 y la 1.1 se parece a ésta (los comentarios están en cursiva):

```
*
* Network device support
*
Network device support? (CONFIG ETHERCARDS) [y]
```

A pesar de que se muestre la contestación por defecto entre corchetes, la pregunta debe ser contestada con `y` si desea utilizar *cualquier* tipo de dispositivos de red, no importa si son Ethernet, SLIP o PPP. Si contesta a la pregunta con `y`, se activará automáticamente el soporte para dispositivos tipo Ethernet. El soporte para otros tipos de controladores de red debe ser activado por separado:

```
SLIP (serial line) support? (CONFIG SLIP) [y]
SLIP compressed headers (SL COMPRESSED) [y]
PPP (point-to-point protocol) support (CONFIG PPP) [y]
PLIP (parallel port) support (CONFIG PLIP) [n]
```

Estas preguntas conciernen a los diversos protocolos de nivel de enlace soportados por Linux. SLIP permite transportar datagramas IP a través de líneas serie. La opción de compresión de cabecera proporciona el soporte CSLIP, una técnica que reduce las cabeceras TCP/IP a tres bytes. Tenga en cuenta que esta opción del núcleo no activa automáticamente el soporte para CSLIP, solamente proporciona las funciones necesarias del núcleo para ello.

PPP es otro de los protocolos para enviar tráfico a la red a través de lineas serie. Es mucho más flexible que SLIP, y no se limita a IP, sino que, una vez que se implemente, también soportará IPX. Ya que el soporte para PPP ha sido incluido hace poco, esto opción puede no aparecer en su núcleo.

PLIP proporciona una forma de enviar datagramas IP a través de un puerto paralelo. Se utiliza generalmente para comunicar dos PCs bajo DOS. El resto de las preguntas son acerca de tarjetas Ethernet de diversos fabricantes. A medida que se desarrollan más controladores, la lista de preguntas se hace mayor. Si desea construir un núcleo que quiera utilizar en varias máquinas, tiene la posibilidad de activar más de un controlador.

```
NE2000/NE1000 support (CONFIG EN2000) [y]
WD80*3 support (CONFIG WD80x3) [n]
SMC Ultra support (CONFIG ULTRA) [n]
3c501 support (CONFIG EL1) [n]
3c503 support (CONFIG EL3) [n]
HP PCLAN support (CONFIG HPLAN) [n]
AT1500 and EN2100 (LANCE and PCnet-ISA) support (CONFIG LANCE) [n]
AT1700 support (CONFIG AT1700) [n]
DEPCA support (CONFIG DEPCA) [n]
D-Link DE600 pocket adaptor support (CONFIG DE600) [y]
AT-LAN-TEC/RealTek pocket adaptor support (CONFIG ATP) [n]
*
* CD-ROM drivers
*
...
```

Por último, en la sección del sistema de ficheros, el script de configuración le preguntará, entre otras cosas, si desea soporte para NFS (networking filesystem), el sistema de ficheros en red. NFS le permitirá exportar sistemas de ficheros a diversos nodos, de forma que parezcan como si estuviesen en un disco duro normal conectado a la máquina local.

```
NFS filesystem support (CONFIG NFS FS) [y]
```

3.2.2 Opciones del núcleo de Linux 1.1.14 y Versiones Posteriores

Comenzando con Linux 1.1.14, que incluía una versión *alpha* de IPX, el proceso de configuración varió muy poco. Ahora las opciones de carácter general preguntan si se desea soporte de red en general. A continuación aparecen un par de preguntas adicionales.

```
*
* Networking options
*
TCP/IP networking (CONFIG INET) [y]
```

Para utilizar protocolos TCP/IP, se debe contestar con `y`. Pero aunque conteste de forma negativa todavía será capaz de poder compilar el núcleo para que soporte IPX.

```
IP fordwarding/gatewaying (CONFIG IP FORWARD) [n]
```

Tendrá que activar esta opción si su sistema actúa como un puente entre dos redes Ethernet, o entre una red Ethernet y un enlace SLIP, etc. Aunque no cuesta nada activar esta opción por defecto, podría querer desactivarla para configurar la máquina como un *cortafuegos*[5]. Los cortafuegos son nodos que se conectan a una o más redes, pero no encaminan tráfico entre ellos. Se utilizan normalmente para proporcionar a los usuarios en una empresa acceso a Internet con un riesgo mínimo para la red interna. A los usuarios se les permitirá acceder al cortafuegos y utilizar servicios Internet, pero las máquinas de la empresa estarán protegidas de ataques externos ya que cualquier conexión entrante no puede cruzar el cortafuegos.

```
*
* (it is safe to leave these untouched)
PC/TCP compatibility mode (CONFIG INET PCTCP) [n]
```

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 UN⋆X normales, pero bajará el rendimiento cuando el enlace sea lento.

[5]N. del T.: Del inglés *firewall*

```
Reverse ARP (CONFIG INET RARP) [n]
```

Esta función activa RARP, Protocolo de Resolución de Direcciones Inverso. RARP se utiliza en clientes sin disco y terminales X para pedir su dirección IP al arrancar. Deberá activar RARP sólo cuando planee que su máquina sea un servidor para este tipo de clientes. El último paquete de utilidades de red (*net-0.32d*) contiene una pequeña utilidad llamada *rarp* que permite añadir direcciones de nodos a una cache RARP.

```
Assume subnets are local (CONFIG INET SNARL) [y]
```

Al mandar datos TCP, el núcleo tiene que dividir los envíos en diversos paquetes antes de pasárselo al nivel IP. Para máquinas accesibles en redes locales como Ethernet , se utilizarán paquetes más grandes que para máquinas cuyos datos son enviados a través de enlaces de larga distancia.[6] Si no se activa la opción *SNARL*, el núcleo asumirá como locales sólo a aquellas redes con las que en ese momento tenga una interface. Si revisa la red de clase B en la Universidad Groucho Marx, toda la red de clase B es local pero la mayoría de los hosts mantienen una interface con sólo una o dos subredes. Si se activa la opción *SNARL*, el núcleo asumirá *todas* las subredes como locales y utilizará paquetes grandes cuando se comunique con todos los nodos del campus.

Si no desea utilizar tamaños de paquete pequeños para enviar datos a máquinas específicas (si, por ejemplo, utiliza un enlace SLIP para la transmisión de datos), tendrá que hacerlo mediante la opción *mtu* del *encaminamiento* (*route*), que se describe brevemente al final de este capítulo.

```
Disable NAGLE algorithm (normally  enabled) (CONFIG TCP NAGLE OFF) [n]
```

La fórmula de Nagle es un método heurístico para evitar enviar paquetes IP particularmente pequeños, también denominados *pequegramas*[7]. Los pequegramas son utilizados normalmente por herramientas de red interactivas que transmiten pulsaciones únicas de teclas, como *telnet* o *rsh* (remote shell). Los pequegramas pueden llegar a ser particularmente ineficientes bajo enlaces de banda estrecha como SLIP. El algoritmo de Nagle intenta evitarlos reteniendo por poco tiempo la transmisión de datos TCP en algunas circustancias. Es recomendable desactivar el algoritmo de Nagle si tiene graves problemas por paquetes perdidos.

```
The IPX protocol (CONFIG IPX) [n]
```

[6] Esto evita la fragmentación por enlaces que tienen un tamaño de paquete máximo muy pequeño.

[7] N. del T.: Del ingés *tinygrams*

Activa la capacidad de soportar el protocolo IPX, el protocolo de transporte utilizado por Novell Networking; que sigue todavía bajo desarrollo, y aún no es realmente útil. Una ventaja de esto será cuando algún día se intercambien datos con utilidades IPX basadas en DOS, y encaminen tráfico entre redes Novell mediante un enlace PPP. El soporte para protocolos de alto nivel de redes Novell no esta todavía a la vista, ya que las especificaciones de estos protocolos tienen un coste económico muy elevado.

A partir de la versión 1.1.16 del núcleo, 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 driver support (CONFIG DUMMY) [y]
```

El controlador vacío no hace realmente gran cosa, pero es bastante útil en máquinas aisladas o conectadas mediante SLIP. Es básicamente un interface en bucle cerrado. La razón de tener este tipo de interface es que en las máquinas que se conectan con SLIP que no disponen de Ethernet, es necesario tener un interface que continuamente maneje las direcciones IP. Esto se discute más profundamente en las sección La Interface Comodín del capítulo 5.

3.3 Una Visita a los Dispositivos de Red de Linux

El núcleo de Linux soporta controladores de hardware de diversas clases. En esta sección se introducen brevemente las familias de controladores disponibles, y los nombres de interfaces que utilizan.

Hay un conjunto de nombres estándares para los interfaces en Linux, que se enumeran a continuación. La mayoría de los controladores soportan más de un interface, en cuyo caso las interfaces se numeran de la forma: *eth0*, *eth1*, etc.

lo Interface de *bucle local* o *de lazo*. Se utiliza para realizar pruebas, y para un par de aplicaciones de red. Funciona como un circuito cerrado en el que cualquier datagrama que se le pase como parámetro será inmediatamente devuelto a la capa de red del sistema. En el núcleo siempre hay un dispositivo de bucle local, no tiene sentido tener más de uno.

eth`n` Tarjeta Ethernet `n`-sima. Este es el nombre de interface genérico para la mayoría de las tarjetas Ethernet.

dl`n` Esta interface accede a un adaptador de bolsillo D-Link DE-600, otro dispositivo Ethernet. Tiene un carácter un poco especial ya que esta conectado a un puerto paralelo.

sln Interface SLIP n-sima. Las interfaces SLIP se asocian a líneas serie en el orden en el que son instalados; por ejemplo, *sl0*, será la primera línea serie en ser configurada para SLIP, etc. El núcleo soporta hasta cuatro interfaces SLIP.

pppn Interface PPP n-sima. Como ocurre con las interfaces SLIP, una interface PPP se asocia a una línea serie una vez que se ha convertido a modo PPP. De momento, se pueden soportar hasta cuatro interfaces de este tipo.

plipn Interface PLIP n-sima. PLIP transporta datagramas IP en líneas paralelas. Se soportan hasta tres interfaces PLIP. El controlador PLIP asigna las interfaces en tiempo de arranque, y se mapean a los puertos paralelos.

Para otros controladores de interfaces que puedan ser añadidos en el futuro como RDSI (Red Digital de Servicios Integrados) o AX.25, se utilizarán otros nombres. Controladores como el de IPX (protocolo Novell de red) o AX.25 (utilizado por radio aficionados) están ya en desarrollo, aunque todavía en versiones preliminares (*alpha*).

En las secciones siguientes se discutirán los detalles de uso de los controladores anteriores.

3.4 Instalación Ethernet

El código de red actual de Linux soporta diversas marcas de tarjetas Ethernet. Donald Becker (**becker@cesdis.gsfc.nasa.gov**) desarrolló la mayoría de los controladores, una familia para tarjetas basadas en el chip 8390 de National Semiconductor; y se la conoce como los la Serie de Controladores Becker. También hay un par de productos de D-Link, entre ellos el adaptador de bolsillo D-Link que permite acceder a una red Ethernet a través de un puerto paralelo. Un controlador para este dispositivo fué programado por Bjørn Ekwall (**bjorn@blox.se**), mientras que el controlador DEPCA lo programó David C. Davies (**davies@wanton.lkg.dec.com**).

3.4.1 Cableado de Ethernet

Si usted está instalando una red Ethernet por primera vez en su vida, son necesarios algunos comentarios respecto al cableado. Ethernet tiene una características muy especiales de cableado. El cable debe terminar en ambos extremos con una resistencia de 50 Ohm, y no debe tener ninguna ramificación (p.e. tres cables conectados en estrella). Si está utilizando cable coaxial fino con conectores BNC en forma de T, estos conectores deben estar directamente conectados al de la tarjeta; no debe insertarse el cable directamente.

Si se conecta a una instalación con cable grueso, debe conectar el ordenador utilizando un transceptor (a veces denominado Unidad de Conexión Ethernet). Puede conectarse el transceptor a un puerto AUI de 25 pines de la tarjeta mediante un cable protegido.

3.4.2 Tarjetas Compatibles

Una lista completa de las tarjetas compatibles está disponible en los Ethernet HOWTOs, publicada mensualmente en **comp.os.linux.announce** por Paul Gortmaker.[8]

Incluímos una lista de las tarjetas más conocidas y utilizadas que soporta Linux. La lista actual del HOWTO es casi tres veces mayor. Aunque encuentre su tarjeta en esta lista, búsquela también en el HOWTO; a veces hay detalles importantes sobre el modo de operación de estas tarjetas. Por ejemplo, algunas tarjetas Ethernet basadas en DMA que utilizan los mismos canales DMA que los que el controlador de SCSI Adaptec 1542 usa por defecto. Si no se cambia el canal de DMA en alguno de las dos, la tarjeta Ethernet podría escribir paquetes de datos en posiciones arbitrarias del disco duro.

3Com EtherLink
: Se soporta 3Com EtherLink tanto 3c503 como 3c503/16, además de las versiones 3c507 y 3c509. La 3c501 también se soporta aunque es demasiado lenta como para merecer la pena.

Novell Eagle
: Se soportan Novell Eagle NE1000 y NE2000, así como diversas clónicas. También soporta las NE1500 y NE2100.

Western Digital/SMC
: Hay que incluir entre las compatibles a la Western Digital/SMC WD8003 y WD8013 (igualmente la SMC Elite y la SMC Elite Plus), y también la nueva SMC Elite 16 Ultra.

Hewlett Packard
: Las HP 27252, HP 27247B, y la HP J2405A.

D-Link
: El adaptador de bolsillo D-Link DE-600, DE-100, DE-200 y DE-220-T. También hay un kit de ampliación para la DE-650-T, denominado tarjeta PCMCIA[9].

DEC
: DEC DE200 (32K/64K), DE202, DE100, y DEPCA rev E.

[8] Paul puede ser localizado en **gpg109@rsphysse.anu.edu.au**.

[9] Puede conseguirse – junto con otro material relacionado con Laptop – en **tsx-11.mit.edu**, en *packages/laptops*.

Allied Teliesis

AT1500 y AT1700.

Para utilizar cualquiera de estas tarjetas con Linux, debe utilizar un núcleo precompilado de una de las distribuciones de Linux. Estas versiones incluyen normalmente controladores para todas las tarjetas. Aunque a la larga será mejor confeccionarse su propio núcleo y compilarlo sólo con los controladores que se necesiten en ese momento.

3.4.3 Autoverificación de red Ethernet

En tiempo de arranque, el código para Ethernet intentará localizar la tarjeta y determinar su tipo. Se comprueban por orden las siguientes direcciones:

Tarjeta	Verificación de las direcciones
WD/SMC	`0x300`, `0x280`, `0x380`, `0x240`
SMC 16 Ultra	`0x300`, `0x280`
3c501	`0x280`
3c503	`0x300`, `0x310`, `0x330`, `0x350`, `0x250`,
	`0x280`, `0x2a0`, `0x2e0`
NEx000	`0x300`, `0x280`, `0x320`, `0x340`, `0x360`
HP	`0x300`, `0x320`, `0x340`, `0x280`, `0x2C0`
	`0x200`, `0x240`
DEPCA	`0x300`, `0x320`, `0x340`, `0x360`

El código de autoverificación tiene dos limitaciones. Una de ellas es que no reconoce todas las tarjetas correctamente. Esto ocurre frecuentemente con algunas de las tarjetas clónicas más baratas, pero también con algunas tarjetas WD80x3. La otra limitación es que el núcleo no verificará más de una tarjeta por defecto. Esto se hace así porque se asume que se desea tener control sobre qué tarjeta es asignada a qué interface.

Si está utilizando más de una tarjeta, o si el proceso de autoverificación no consigue detectar su tarjeta, debe indicar explicítamente al núcleo el nombre y dirección base de la tarjeta.

Con Net-3, se tienen dos posibilidades diferentes ante un caso de fallo en la autoverificación. Una de ellas es cambiar o añadir información en el fichero del código fuente del núcleo *drivers/net/Space.c*, que contiene toda la información sobre controladores. Sólo es recomendable en el caso de que el codigo de red le sea familiar. Una forma mucho mejor es proporcionar al núcleo esta información en tiempo de arranque. Si utiliza *lilo* al arrancar el sistema, puede pasarle parámetros al núcleo, especificándolos mediante la opción *append* en el fichero *lilo.conf*. Por ejemplo, para informar al núcleo sobre la existencia de un dispositivo Ethernet, puede pasarse el siguiente parámetro:

```
ether=irq, dir_base, param1,param2,nombre
```

Los cuatro primeros argumentos son numéricos, mientras que el último es el nombre de un dispositivo. Todos los valores numéricos son opcionales: si se omiten o se dejan a cero, el núcleo intentará averiguar su valor mediante autoverificación, o utilizando un valor por defecto.

El primer parámetro indica el IRQ asignado al dispositivo. Por defecto, el núcleo intentará autodetectar el canal IRQ del dispositivo. El controlador 3c503 tiene un funcionamiento especial seleccionando un canal libre IRQ de la lista 5, 9, 3, 4, y configura la tarjeta para utilizar esta línea.

El parámetro ***dir_base*** define la dirección base de E/S de la tarjeta; si vale cero, el núcleo probará con las direcciones de la lista anterior.

Los dos parámetros restantes pueden ser utilizados de forma diferente por controladores diferentes. Para tarjetas de memoria compartida como la WD80x3, especifican la dirección de comienzo y de fin del área de memoria compartida. Otras tarjetas utilizan normalmente el ***param1*** para seleccionar el nivel de información de depuración para el usuario. Los valores del 1 al 7 denotan niveles de detalle en la información, mientras que el valor 8 desactiva todos; por defecto se toma el valor cero. El controlador 3c503 utiliza el ***param2*** para seleccionar el transceptor interno (por defecto) o un transceptor externo (valor 1). El primero utiliza un conector de tarjeta tipo BNC; el último utiliza su puerto AUI.

Si se tienen dos tarjetas Ethernet, puede hacerse que Linux autodetecte una de ellas, y pasar los parámetros de la segunda mediante *lilo*. Sin embargo, hay que estar seguros de que el controlador accidentalmente no encuentra la segunda tarjeta primero, en cuyo caso la otra no se detectará. Esto se consigue pasando la opción `reserve` a *lilo*, que indica explícitamente al núcleo que no verifique el espacio de E/S reservado para la segunda tarjeta.

Por ejemplo, para hacer que Linux instale una segunda tarjeta Ethernet en la dirección (0x300) como *eth1*, hay que pasarle los siguientes parámetros al núcleo:

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

La opción *reserve* asegura que ningún controlador accede al espacio de E/S del núcleo cuando verifica algún dispositivo. También pueden utilizarse los parámetros del núcleo para evitar realizar la verificación para *eth0*:

```
reserve=0x340,32 ether=0,340,eth0
```

Para evitar completamente la fase de autoverificación, se especifica el argumento ***dir_base*** a -1:

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

3.5 El controlador PLIP

PLIP permite trabajar con una *Línea Paralela IP*, y es una forma barata de interconexión cuando se desea conectar solamente dos máquinas. Utiliza un puerto paralelo y un cable especial, alcanzando velocidades entre 10kBps a 20 kBps.

PLIP fue desarrollado en principio por Crynwr, Inc. Su diseño es más que ingenioso (o, si se prefiere, más propio de un hacker): durante muchos años, los puertos paralelos del PC se utilizaban como puertos para impresora unidireccionales; es decir las ocho líneas de datos podrían ser utilizados solamente para envíos desde el PC al dispositivo periférico, pero no en sentido inverso. El PLIP saca un mejor provecho utilizando la línea de estado número cinco del puerto, que se limita a transferir todos los datos simplemente como *nibbles* (es decir medio byte). Hoy en día, estos puertos unidireccionales no son muy utilizados. Por tanto también hay una extensión denominada modo 1 que utiliza la interface completa de 8 bits.

Actualmente Linux sólo soporta el modo 0. A diferencia de versiones más antiguas del código que maneja el PLIP, ahora se intenta hacerlo compatible con las implementaciones PLIP de Cynwr, y para el controlador PLIP en el NCSA *telnet*[10]. Para conectar dos máquinas utilizando PLIP, se necesita un cable especial que se vende en algunas tiendas como cable de "Impresora Nula" ("Null Printer") o "Turbo Laplink". Es posible fabricarlo de forma casera, en el Apéndice A se indica cómo.

El controlador PLIP para Linux es el trabajo de un número casi incontable de personas. Actualmente esta mantenido por Niibe Yutaka. Si se compila en el núcleo, configura una interface de red para cada posible puerto de impresora, con *plip0* correspondiendo al puerto paralelo *lp0*, *plip1* correspondiendo a *lp1*, etc. El mapeado del interface a los puertos es ahora mismo el siguiente:

[10] El NCSA *telnet* es un programa bastante conocido para DOS que trabaja con TCP/IP en redes Ethernet o PLIP, y soporta *telnet*, FTP y algunas otras aplicaciones sencillas.

Interface	Puerto E/S	IRQ
plip0	`0x3BC`	7
plip1	`0x378`	7
plip2	`0x278`	5

Si se tiene configurado el puerto de la impresora de forma diferente, hay que cambiar estos valores en el fichero *drivers/net/Space.c* del código fuente de Linux, y construir un nuevo núcleo.

Este mapeado no implica que no se puedan utilizar los puertos paralelos de forma normal. Sólo son accedidos por el controlador de PLIP cuando el interface correspondiente ha sido configurado como *activo*.

3.6 Los controladores SLIP y PPP

SLIP (Serial Line IP, Protocolo Internet en Línea Serie), y PPP (Point-to-Point Protocol, Protocolo Punto-a-Punto) son protocolos muy utilizados para enviar paquetes IP a través de enlaces serie. Varias instituciones ofrecen acceso telefónico SLIP y PPP a máquinas conectadas a Internet: esto proporciona conectividad IP a los particulares (algo que de otra forma sería difícil de conseguir debido al elevado coste de otros tipos de conexiones).

Para trabajar con SLIP o PPP, no son necesarias modificaciones en el hardware; puede utilizarse cualquier puerto serie. Ya que es específica la configuración del puerto serie para interconexión TCP/IP, se le dedicará un capítulo aparte. Para más información consultar el capítulo 4.

Capítulo 4

Configuración del Software Serie

Casi todo el mundo dispone de un PC, pero no siempre hay dinero para gastarlo en un enlace Internet T1. Para conseguir su dosis diaria de noticias y mensajes, mucha gente depende de enlaces SLIP, redes UUCP y BBS, que usan las redes telefónicas públicas.

Este capítulo pretende ayudar a todas aquellas personas que dependen del módem para mantener sus comunicaciones. Sin embargo, hay muchos detalles que no podemos abordar, como por ejemplo cómo configurar el módem para marcar. Todos esos temas están contemplados en el "Serial HOWTO"[1] de Greg Hankins[2], que es enviado a **comp.os.linux.announce** regularmente.

4.1 Software de Comunicaciones con Módem

Existen varios paquetes de comunicaciones disponibles para Linux. Muchos de ellos son *emuladores de terminal*, que permiten a un usuario conectarse a otro ordenador como si estuviera frente a uno de sus terminales. El emulador de terminal tradicional en sistemas UN⋆X es *kermit*. Sin embargo resulta algo duro de usar. Hay programas disponibles más cómodos que soportan agenda telefónica y guiones para llamar y entrar en ordenadores remotos. Uno de estos es el *minicom*, muy parecido a los primitivos programas emuladores de terminal a los que tan acostumbrados están los usuarios de DOS. Hoy también existen paquetes de comunicaciones bajo X-11 como por ejemplo *seyon*.

Además, existe un buen número de programas para instalar BBS bajo Linux disponibles para aquellos que quieran ofrecer dicho servicio. Varios de esos paquetes se encuentran en **sunsite.unc.edu**, en el directorio */pub/Linux/system/Network*.

[1] N. del T.: Disponible en castellano como SERIE-COMO, en http://lucas.ctv.es/

[2] disponible en **gregh@cc.gatech.edu**.

Aparte de los programas de terminal, hay también software que usa la línea serie de forma no interactiva para el transporte de datos hasta su ordenador. Normalmente se invierte bastante más tiempo en visitar un BBS leyendo toda su información en la que podemos incluir las noticias y los mensajes, que el que se necesita empleando este tipo de software. La única desventaja es que se requiere más espacio de disco debido a la transferencia de cierta cantidad de información que al usuario le resulta inútil, y que de forma interactiva no se tranmitiría.

El compendio de esta clase de software de comunicaciones es UUCP. Éste es un conjunto de programas que copian ficheros de una máquina a otra, ejecutan programas en un ordenador remoto, etc. Se utiliza frecuentemente para transferir mensajes y noticias (news) entre redes privadas. El paquete UUCP de Ian Taylor, que funciona bajo Linux, será descrito en el capítulo 12 de este libro. Otro tipo de software de comunicaciones no interactivo es el utilizado en Fidonet, para el que también podemos encontrar algunos paquetes de software, como *ifmail*.

SLIP, el protocolo de Internet para línea serie, está de algún modo a medio camino: permite tanto el uso interactivo como el no interactivo. Mucha gente usa SLIP para telefonear a la red de su campus o algún otro tipo de servidor público y poder ejecutar sesiones FTP, etc. Sin embargo, SLIP también puede ser usado en conexiones permanentes o semipermanentes para uniones de LAN a LAN, aunque esto último sólo resulta interesante utilizando RDSI u otros enlaces de ancho de banda mayor.

4.2 Introducción a los Dispositivos Serie

Los dispositivos proporcionados por un núcleo UN⋆X para el acceso a dispositivos serie son llamados normalmente *ttys*. Ésta es una abreviatura de *Teletype*™, quienes eran unos de los mayores productores de terminales en los primeros días de Unix. El término se usa actualmente para cualquier terminal de texto. En este capítulo, lo usaremos exclusivamente para referirnos a los dispositivos del núcleo.

Linux distingue tres clases de ttys: consolas (virtuales), pseudo terminales (similares a las tuberías de doble vía, usadas por aplicaciones tales como X11), y dispositivos serie. Estos últimos son considerados también como ttys, porque permiten sesiones interactivas sobre conexiones serie, ya sea éste un terminal conectado por cable o un ordenador remoto a través de la línea telefónica.

Los ttys tienen cierto número de parámetros configurables mediante la llamada al sistema *ioctl(2)*. Muchos de estos parámetros sirven únicamente con dispositivos serie, ya que son éstos los que necesitan una mayor flexibilidad para poder manejar la gran variedad de tipos de conexión que son capaces de controlar.

Entre los parámetros más destacados para la línea se encuentran la velocidad y la paridad. Pero hay también elementos para la conversión de caracteres entre mayúscula y minúscula, de retorno de carro, de avance de línea, etc. El controlador de tty puede también soportar varias *líneas dedicadas*, las cuales hacen que el controlador de dispositivo se comporte de forma diferente. Por ejemplo, el controlador de SLIP para Linux está implementado como si fuera una línea dedicada.

Existe algo de ambigüedad sobre cómo medir la velocidad de la línea. El término correcto es *bit rate*, el cual está relacionado con la velocidad de transferencia de la línea medida en bits por segundo (bps para abreviar). Algunas veces se oye a la gente referirse a ella como velocidad en *baudios*, lo cual no es muy correcto, ya que estos dos términos no son sinónimos. La *velocidad en baudios* se refiere a una característica física de algunos dispositivos serie. En concreto, a la velocidad de reloj a la que se transmiten los impulsos. En cambio, el "bit rate", indica el estado actual de una conexión serie existente entre dos puntos, a saber, el número medio de bits transferidos por segundo. Es importante saber que estos dos valores suelen ser diferentes, ya que la mayoría de los dispositivos codifican más de un bit por cada impulso eléctrico.

4.3 Acceso a los Dispositivos Serie

Como ocurre con todos los dispositivos de un sistema UN⋆X, se accede a los puertos serie a través de ficheros especiales de dispositivo, localizados en el directorio */dev*. Cada puerto tiene su fichero de dispositivo. Hay dos tipos de ficheros de dispositivos relacionados con los controladores serie. Dependiendo del fichero por el que se acceda el dispositivo se comportará de forma diferente.

El primer tipo se utiliza para las llamadas entrantes y tiene un *número principal de dispositivo*[3] igual a 4. Sus ficheros son nombrados *ttyS0*, *ttyS1*, etc. El segundo tipo se utiliza para llamadas de salida a través de un puerto. Sus ficheros son llamados *cua0*, etc y tienen un número principal de dispositivo igual a 5.

Los *números secundarios*[4] son los mismos para los dos tipos. Si tiene su módem en uno cualquiera de los puertos *COM1* a *COM4*, su número secundario será el número de puerto *COM* más 63. Si su configuración es diferente a ésta, como sucede, por ejemplo, en placas que soportan múltiples líneas serie, debe en tal caso buscar en el documento COMO-SERIE o SERIAL-HOWTO.

Asumamos que su modem está en el *COM2*. En este caso su número secundario será 65, y su número principal será 5 para realizar llamadas. Debería existir por ello, un dispositivo

[3] N. del T.: Del inglés *major number*
[4] N. del T.: Del inglés *minor number*

cua1 que tuviera dichos números de dispositivo. A continuación vemos una lista de ttys serie del directorio */dev*. Las columnas 5 y 6 muestran los números principal y secundario respectivamente.

```
$ ls -l /dev/cua*
crw-rw-rw-   1 root     root       5,  64 Nov 30 19:31 /dev/cua0
crw-rw-rw-   1 root     root       5,  65 Nov 30 22:08 /dev/cua1
crw-rw-rw-   1 root     root       5,  66 Oct 28 11:56 /dev/cua2
crw-rw-rw-   1 root     root       5,  67 Mar 19  1992 /dev/cua3
```

Si no existiesen tales dispositivos, entonces tendría que crearlos. Para ello, conviértase en superusuario y teclee comandos como el siguiente:

```
# mknod -m 666 /dev/cua1 c 5 65
# chown root.root /dev/cua1
```

Hay quien propone la creación de un enlace simbólico del puerto serie en dónde tenga su módem, a un fichero */dev/modem*. De esta forma no es necesario recordar el poco intuitivo *cua1*. Sin embargo, podemos encontrarnos con problemas si empleamos el nombre real del dispositivo en unos programas y el simbólico en otros. La explicación es que las aplicaciones en Unix usan un convenio de *ficheros cerrojo* para indicar que cierto dispositivo está siendo utilizado por un proceso y evitar así que pueda ser utilizado por otro al mismo tiempo. Por convenio, el nombre del fichero de bloqueo para *cua1*, es *LCK..cua1*. El uso de distintos ficheros de dispositivo para el mismo puerto implica que se puede producir ausencia de exclusión mutua en el acceso al puerto si un programa usa un nombre de dispositivo y otro programa usa el otro nombre (el simbólico). Esto puede provocar un acceso simultáneo de ambos procesos al mismo puerto y que, por tanto, ninguna de ellas funcione correctamente.

4.4 Hardware Serie

Linux soporta, hoy por hoy, una amplia variedad de placas serie que utilizan el estándar RS-232. RS-232 es, en la actualidad, el estándar más común para comunicaciones serie en el mundo del PC. Éste usa un conjunto de circuitos tanto para transmitir simples bits así como para establecer sincronización. Pueden utilizarse cables adicionales para señalar la presencia de una *portadora* y para el *control de flujo*.

Aunque el *control de flujo* por hardware es opcional, resulta muy útil ya que permite a cada una de las dos estaciones señalar cuándo está lista para recibir más datos, o si la otra estación debe parar hasta que el receptor procese los datos de entrada. Las líneas usadas para esto son las llamadas *Clear to Send*, despejado para envíos, (CTS) y *Ready to Send*, listo para enviar (RTS), respectivamente.

En ordenadores PC, el interfaz RS-232 es controlado generalmente por un chip UART descendiente del chip 16450 de National Semiconductor, o bien de una nueva versión de éste: el NSC 16550A[5]. Algunas marcas (principalmente los modems internos equipados con un chip Rockwell) también usan chips completamente diferentes que han sido programados para comportarse como si fueran un 16550.

La principal diferencia entre los 16450 y los 16550 estriba en que el primero tiene un buffer de 1 byte mientras que el segundo lo tiene de 16 bytes. Esto hace al 16450 válido para velocidades máximas de 9600 baudios, mientras que para velocidades superiores se requiere un chip compatible con el 16550. Además de estos chips, Linux también soporta el chip 8250, que era el chip UART original de los PC de IBM.

En la configuración por defecto, el núcleo comprueba los cuatro puertos serie estándar, es decir, del *COM1* hasta el *COM4*, a los que les asignará los numeros secundarios desde el 64 hasta el 67, tal y como se ha descrito anteriormente.

Si desea configurar su puerto serie adecuadamente, tendría que incluir la orden *setserial* de Ted Tso en el fichero de comandos *rc.serial*, el cual es invocado durante el arranque del sistema desde el fichero de comandos de inicialización */etc/rc*. Este primer fichero, usa *setserial* para configurar los dispositivos serie del núcleo. Un típico fichero de comandos *rc.serial* tendrá el siguiente aspecto:

```
# /etc/rc.serial - guion de configuracion de la linea serie
#
# Deteccion de interrupciones libres
/sbin/setserial -W /dev/cua*

# Configurar dispositivos serie
/sbin/setserial /dev/cua0 auto_irq skip_test autoconfig
/sbin/setserial /dev/cua1 auto_irq skip_test autoconfig
/sbin/setserial /dev/cua2 auto_irq skip_test autoconfig
/sbin/setserial /dev/cua3 auto_irq skip_test autoconfig

# Muestra la configuracion de dispositivos serie
/sbin/setserial -bg /dev/cua*
```

Si desea conocer más sobre los parámetros de *setserial*, por favor, consulte la documentación que acompaña al programa.

[5] Había también un NSC 16550, pero este chip FIFO nunca funcionó realmente.

Si su tarjeta serie no es detectada, o la orden *setserial -bg* muestra una configuración incorrecta, tendrá que forzar la configuración suministrando explícitamente los valores correctos. Está comprobado que los modems internos equipados con los chips de Rockwell experimentan este tipo de problemas. Así, por ejemplo, si se obtiene que el chip de una UART es el NSC 16450, siendo en cambio del tipo NSC 16550, se tendrá que cambiar la configuración del puerto implicado de la forma siguiente:

```
/sbin/setserial /dev/cua1 auto_irq skip_test autoconfig uart 16550
```

Existen opciones similares para forzar los puertos *COM*, direcciones base, y configuración de petición de interrupción (IRQ). Por favor consulte la página del manual de *setserial(8)* para más información.

Si su módem soporta *control de flujo* mediante hardware, asegúrese de activarlo. Sorprendentemente, la mayoría de los programas de comunicaciones no intentan activarlo por defecto. Por ello, lo mejor es realizarlo manualmente, y la mejor forma de lograrlo es incluirlo en el fichero de comandos *rc.serial* usando la orden *stty*:

```
$ stty crtscts < /dev/cua1
```

Para comprobar si el *control de flujo* por hardware está activo use:

```
$ stty -a < /dev/cua1
```

Este comando le devolverá el estado de todos los parámetros de dicho dispositivo. Un parámetro precedido con un signo menos como en `-crtscts` significa que ha sido desactivado.

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. Algunos de los comando 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 una macro llamada *rc.net* o *rc.inet*. A veces también son llamadas *rc.inet1* o *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 todo lo que sigue, asumiré que es ésta la estructura presente en el sistema.

Más abajo describiré las acciones llevadas a cabo por *rc.inet1*, mientras que las aplicaciones son cubiertas por los capítulos posteriores. Al finalizar este capítulo, debería usted haber establecido la secuencia de comandos que configuran correctamente el protocolo TCP/IP en su ordenador. Substituya los comandos de ejemplo en *rc.inet1* por los suyos propios; asegúrese de que *rc.inet1* es ejecutada en el arranque y rearranque su máquina. Los scripts *rc* que acompañen a su distribución de Linux favorita deberían ser un buen ejemplo.

5.1 Configuración del Sistema de Ficheros *proc*

Algunas de las herramientas de configuración de Net-2 utilizan el sistema de ficheros *proc* para comunicarse con el núcleo. Se trata de una interface que permite el acceso a la información del kernel en funcionamiento a través de 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.

El código de redes añade a esto el directorio *net*. Este directorio contiene una serie de ficheros con información sobre 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*) es montado generalmente en */proc* durante el arranque. El mejor método consiste en añadir la siguiente línea al fichero */etc/fstab*:

```
# Lugar de montaje de procfs:
none            /proc           proc    defaults
```

y ejecutar "`mount /proc`" desde uno de los macros */etc/rc*.

El *procfs* viene configurado actualmente en la mayoría de los núcleos por defecto. Si no tiene el *procfs* en su núcleo, al intentar montarlo obtendrá el mensaje "`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 Linux, probablemente 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 conseguir un conjunto de ejecutables actualizados en ficheros llamados *net-`XXX`.tar.gz*, donde `XXX` es la versión de que se trate. En el caso de Linux 1.0 es la número 0.32b, y la versión del núcleo en el momento en que se escribió este libro (1.1.12 y posterior) requiere 0.32d.

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 Linux. Se trata de versiones modificadas de las fuentes de Net-BSD y otros. Otras aplicaciones, como *Xmosaic*, *xarchie*, o Gopher y los clientes IRC deben obtenerse por separado. La mayoría compila sin necesidad de modificaciones si se siguen las instrucciones particulares.

El servidor FTP oficial de Net-3 es **sunacm.swan.ac.uk**, que es replicado en **sunsite.unc.edu** bajo *system/Network/sunacm*. El último parche y los ejecutables de Net-2e están disponibles en **ftp.aris.com**. El código derivado de BSD, de Matthias Urlichs, se encuentra en **ftp.ira.uka.de**, en el directorio */pub/system/linux/netbsd*.

5.3 Otro Ejemplo

Para lo que queda de este libro, voy a utilizar un ejemplo menos complejo que el de la Universidad Groucho Marx, y que puede acercarse más a las tareas que realmente tendrá que realizar. La Cervecera Virtual, es una pequeña compañia que produce, como su nombre indica, cerveza virtual. Para gestionar su negocio más eficientemente, los cerveceros virtuales quieren conectar sus ordenadores, que casualmente son PCs bajo Linux 1.0, en red.

En el mismo piso, justo al otro lado del edificio se, encuentra la Vinatera Virtual, que trabaja de cerca con la cervecera. La vinatera tiene una red Ethernet propia. Naturalmente, ambas compañías quieren unir sus redes una vez que éstas se encuentren operacionales. Como un primer paso, quieren establecer una pasarela que pase los datagramas de una subred a otra. Para más tarde, tienen planeado establecer un enlace UUCP con el exterior, a través del cual intercambiarán noticias y correo electrónico. A largo plazo, quieren establecer una conexión ocasional usando SLIP con la Internet.

5.4 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 `nodo1` a un ordenador ejecutaría

```
# hostname nodo1
```

Es una práctica común usar el nombre sin cualificarlo con el dominio de red. Así pues, supongamos que las máquinas de la Cervecera Virtual se llamaran **vale.vbrew.com**,

vlager.vbrew.com, etc. Éstos son los nombres oficiales, los nombres *completamente cualificados de dominio* (FQDN[1]). 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 encontrarla. Esto generalmente equivale a añadir el nombre local al fichero */etc/hosts* (ver más abajo).

Algunas personas sugieren la utilización del comando *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). Hablaremos de NIS en el capítulo 10.

5.5 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 interface de lazo.

Las cosas son algo más complicadas en redes reales como las Ethernets. Si quiere conectar su ordenador 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 según se describe a continuación.

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 completamente diferentes a cada una o dividir el rango de direcciones IP disponibles en varias subredes.

Si su red no esta conectada con Internet, es libre de elegir cualquier dirección (válida). Sólo tiene que asegurarse de elegir una de entre los tipos A, B, o C, o, de otro modo, no irán bien las cosas. Sin embargo, si planea conectarse a la Internet en un futuro cercano, tiene usted que obtener una dirección IP oficial *ya*. La mejor forma es pedir ayuda a su proveedor de servicios de Internet. Si quiere pedir una dirección oficial en previsión de que se conecte a la Internet algún día, pida un formulario de solicitud de dirección de red a **hostmaster@internic.net**.

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 notar que esto es

[1]N. del T.: Del inglés fully qualified domain name

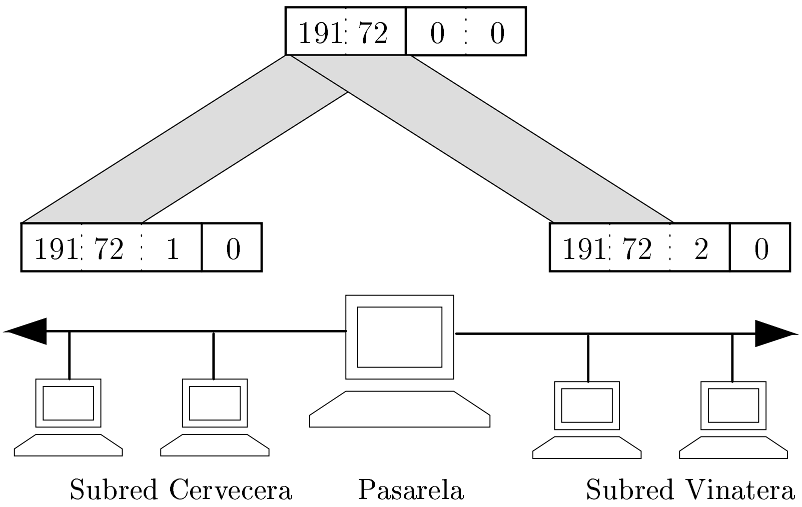

Figura 5.1: Cervecera Virtual y Vinatera Virtual – las dos subredes.

únicamente necesario si tiene más 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 más enlaces SLIP con el exterior no hace falta que divida su red. La razón se explica en el capítulo 7.

A modo de ejemplo, el administrador de la red de la cervecera solicita al NIC una dirección de red de tipo B, siéndole asignada la número **191.72.0.0**. Para acomodar dos redes ethernet, decide usar ocho bits de la parte de la dirección correspondiente a los ordenadores como dirección de subred. Eso deja otros ocho bits para las máquinas lo que equivale a 254 por cada subred. La red de la cervecera se convierte así en la subred 1 y la de la vinatera en la subred 2. Las direcciones de red serán por tanto **191.72.1.0** y **191.72.2.0**. La máscara de red será **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. La figura 5.1 muestra las dos subredes y la máquina que actúa de enlace.

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.[2]

[2]La última dirección en realidad se reserva como dirección de difusión, aquella a la que se envían mensajes destinados a todas las máquinas de la red correspondiente, lo cual en realidad deja sólo 63 por subred.

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*. Por un lado, hay situaciones en las que es necesario resolver algunos 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 las macros *rc.inet*. De esta forma, para cambiar las direcciones IP, sólo tiene que copiar el fichero *hosts* modificado a todas las máquinas y rearrancar, en vez de tener que modificar un gran número de macros *rc* por separado. Generalmente, también debe incluir los nombres y direcciones locales en *hosts*, añadiendo los de las máquinas que enlacen varias redes y los servidores NIS si existen.[3]

También, en la fase inicial de pruebas, debería asegurarse de que el subsistema de resolución utiliza la información del fichero *hosts* únicamente. Su software DNS o NIS puede incluir ficheros de configuración a modo de ejemplo que pueden producir resultados extraños si son usados. 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 librería de resolución se describe en detalle en el capítulo 6.

El fichero *hosts* contiene un registro por linea, 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 interpretado como un comentario y es consecuentemente ignorado.

Los nombres de las máquinas pueden ser con cualificación completa, o relativos al dominio actual. Para la máquina **vale**, el registro generalmente incluiría el nombre con cua-

[3]Sólo necesita incluir las direcciones de los servidores NIS si utiliza el servidor NYS de Peter Eriksonn. Otras implementaciones de servidores NIS son capaces de localizar los servidores cuando están siendo ejecutados, utilizando *ypbind*.

lificación 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.

Éste es un ejemplo del aspecto que el fichero *hosts* de la Cervecera Virtual podría tener. Hay dos nombres especiales **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            local       fully qualified domain name
#
127.0.0.1       localhost
#
191.72.1.1      vlager      vlager.vbrew.com
191.72.1.1      vlager-if1
191.72.1.2      vstout      vstout.vbrew.com
191.72.1.3      vale        vale.vbrew.com
#
191.72.2.1      vlager-if2
191.72.2.2      vbeaujolais  vbeaujolais.vbrew.com
191.72.2.3      vbardolino   vbardolino.vbrew.com
191.72.2.4      vchianti     vchianti.vbrew.com
```

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, podrímos instalar un fichero *networks* como éste:[4]

```
# /etc/networks para la Cervecera Virtual
brew-net      191.72.1.0
wine-net      191.72.2.0
```

5.7 Configuración de la Interface para IP

Una vez ha configurado su hardware según se ha explicado en el capítulo anterior, debe asegurarse de que el software de red del núcleo conoce esos dispositivos. Hay una serie de comandos que se usan con objeto de configurar las interfaces de red e inicializar la tabla de encaminamiento. Esas tareas son ejecutadas generalmente por la macro *rc.inet1* cada

[4]Es importante notar que los nombres en *networks* no deben coincidir con nombres de máquinas en *hosts*, o algunos programas pueden producir resultados extraños.

vez que el sistema es arrancado. Las herramientas básicas son *ifconfig* (donde "if" significa interface), y *route*.

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

```
ifconfig interface direccion-ip
```

que asigna ***direccion-ip*** a ***interface*** y la activa. Los otros parametros toman valores asignados por defecto. Por ejemplo, la máscara de subred 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 al final del capítulo.

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

```
route [add|del] destino
```

donde los argumentos `add` y `del` determinan, respectivamente si se debe añadir o borrar la ruta hacia ***destino***.

5.7.1 La Interface de Bucle o Loopback

La primera interface en ser activada es la interface de lazo o loopback:

```
# ifconfig lo 127.0.0.1
```

Ocasionalmente, también verá 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 ejemplo de localhost en /etc/hosts
localhost     127.0.0.1
```

Para ver la configuración de una interface, basta ejecutar el programa *ifconfig* usando el nombre de la interface como argumento:

```
$ ifconfig lo
lo        Link encap Local Loopback
          inet addr 127.0.0.1  Bcast [NONE SET]  Mask 255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU 2000  Metric 1
          RX packets 0 errors 0 dropped 0 overrun 0
          TX packets 0 errors 0 dropped 0 overrun 0
```

Como podrá observar, la máscara asignada a la interface de lazo es **255.0.0.0**, debido a que **127.0.0.1** es una dirección de clase A. La interface no tiene establecida ninguna dirección de difusión, ya que ésta no suele ser demasiado útil para lazos. Sin embargo, si va a ejecutar el demonio *rwhod* en su máquina, tendrá seguramente que fijar la dirección de difuón del dispositivo de lazo para que *rwho* funcione correctamente. El modo de fijar dicha dirección se explica en la sección "5.8", más abajo.

Ahora, ya casi puede empezar a jugar con su "mini-red". Sólo resta añadir una entrada en la tabla de encaminamiento que comunique al IP que puede usar esa interface 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.

Lo siguiente es comprobar que todo funciona como es debido, por ejemplo usando *ping*. *ping* es el equivalente a un sonar en una red[5] y 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=32 time=1 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=32 time=0 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=32 time=0 ms
^C

--- localhost ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0/0/1 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.

[5] ¿Alguno recuerda "Echoes" de Pink Floyd?

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 interface de red.

Si la salida de *ping* no se parece a la de más arriba, tiene usted 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 el comando *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 lineas son añadidas a *rc.inet1* y después de asegurarse de que las dos macros *rc.inet* son ejecutadas desde */etc/rc*, 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 interface de lazo 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 operación.[6] 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 interface Ethernet es más o menos igual que la de la interface de lazo. 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 interface reconozca esto, el comando usando *ifconfig* sería:

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

Esto asigna a la interface *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**. Una comprobación rápida nos da:

```
# ifconfig eth0
eth0      Link encap 10Mps Ethernet HWaddr  00:00:C0:90:B3:42
```

[6]Por ejemplo, todas las aplicaciones basadas en RPC utilizan la interface de lazo para registrarse en el demonio *portmapper*(mapa de puertos) durante el arranque.

```
inet addr 191.72.1.2 Bcast 191.72.1.255 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 interface) a un máximo de 1500 bytes. Todos estos valores pueden ser especificados mediante opciones especiales que se explican más tarde.

De forma semejante al caso de la interface de lazo, debe también ahora establecer 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 191.72.1.0
```

Inicialmente podría parecer algo mágico, pues no está claro cómo *route* detecta cuál es la interface 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 "Y" lógico de la dirección de la interface y la máscara de red). La única interface 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 interface Ethernet esta funcionando correctamente. Elija una máquina de su red, por ejemplo **vlager**, y escriba

```
# ping vlager
PING vlager: 64 byte packets
64 bytes from 191.72.1.1: icmp_seq=0. time=11. ms
64 bytes from 191.72.1.1: icmp_seq=1. time=7. ms
64 bytes from 191.72.1.1: icmp_seq=2. time=12. ms
64 bytes from 191.72.1.1: icmp_seq=3. time=3. ms
^C

----vstout.vbrew.com PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
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[7] 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 interface mediante *netstat*. Las estadísticas de paquetes producidas por *ifconfig* le indican si algún paquete ha sido enviado mediante esa interface. Si tiene acceso a una máquina remota, también debería dirigirse a esa máquina y comprobar las estadísticas de la interface. 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
191.72.1.0      *               255.255.255.0   U     1      0       10 eth0
```

El significado de cada uno de los campos se detalla más adelante en la sección "Comprobación mediante *netstat*". La columna `Flags` contiene una lista de los indicadores activos en cada interface. `U` indica que la interface 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 está siendo usada o no, debe mirar si el campo `Use` en la penúltima columna se incrementa entre dos ejecuciones sucesivas de *ping*.

[7] N. del T.: Del inglés packet loss rate

5.7.3 Encaminamiento a través de una Pasarela

En la sección anterior, cubrí 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 tambien servir de enlace con el exterior, con la 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 todos las máquinas 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 host 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 de **vstout** a **vbardolino**, pero la respuesta del segundo iría a parar al *cubo de la basura*.

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 defecto 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 defecto. No es necesario añadir este nombre a */etc/networks*, porque esta información está 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:

```
191.72.1.1      vlager      vlager.vbrew.com
191.72.1.1      vlager-if1
191.72.2.1      vlager-if2
```

La secuencia de comandos necesaria para establecer ambas interfaces será por tanto:

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

5.7.5 La Interface 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*, porque sólo requiere dos máquinas ("puntos"), en contraposición a las redes de difusión.

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 interface *plip1* mediante los siguientes comandos:

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

El primer comando configura la interface, diciéndole al núcleo que se trata de un enlace punto-a-punto, donde la parte remota tiene la dirección **vlager**. El segundo instala la ruta por defecto 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 interface *plip1* en **vlager** no necesita tener una dirección IP diferente, sino que puede usar la misma dirección **191.72.1.1**.[8]

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 sería 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 ARP sustituto (*proxy* ARP). Con ARP sustituto, **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 ARP sustituto en la sección "Comprobación de las Tablas ARP", más adelante.

Las versiones futuras de Net-3 contendrán una herramienta llamada *plipconfig* capaz de fijar el número de IRQ del puerto de la impresora. Más tarde se sustituirá por un comando *ifconfig* más general.

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 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 los capítulos 7 y 8.

5.7.7 La Interface Comodín

La interface comodín (dummy) parece un tanto exótica y sin embargo es bastante útil. Resulta especialmente ventajosa para máquinas aisladas que se conectan a una red IP

[8] 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.

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 lazo, 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 está conectado a ninguna red. Una aplicación en **vlite** que busque su dirección IP en el fichero */etc/hosts* dará como resultado **191.72.1.65**, y por tanto intentará enviar los datos a esa dirección. Como la única interface activa en ese momento es la de lazo, 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 interface comodín es útil, resolviendo el dilema actuando como *alter ego* de la interface de lazo. En el caso de **vlite**, simplemente debe asignarle la dirección **191.72.1.65** y añadir una ruta que apunte a ella. La forma correcta es pues:

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

5.8 Todo sobre *ifconfig*

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

```
ifconfig interface [[-net|-host] direccion [parametros]]
```

interface es el nombre de la interface y ***direccion*** es la dirección IP que se asigna a dicha interface. La dirección puede estar en forma de cuaterna o usando un nombre que *ifconfig* buscará en */etc/hosts* y */etc/networks*. La opciones **-net** y **-host** fuerzan a *ifconfig* a tratar las direcciones dadas como direcciones de red o de máquina respectivamente.

Si *ifconfig* es ejecutado añadiendo únicamente el nombre de la interface, presentará la información de la configuración de dicha interface. Si se ejecuta sin parametros, 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 interface Ethernet *eth0* sería:

```
# ifconfig eth0
eth0      Link encap 10Mbps Ethernet  HWaddr 00:00:C0:90:B3:42
          inet addr 191.72.1.2 Bcast 191.72.1.255 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 interface dada. El valor de la métrica es usado tradicionalmente por algunos sistemas operativos para calcular el coste de una ruta. 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 parametros con los que se puede ejecutar; estos parámetros son explicados más abajo.

A continuación tenemos una lista de los parámetros reconocidos por *ifconfig*. Los nombres de los indicadores correspondientes aparecen entre paréntesis. Las opciones que simplemente activan alguna característica pueden usarse para desactivarla precediéndolas de un guión (-).

`up` Marca la interface como "up" o activa, es decir, disponible para que sea usada por la capa IP. Esta opción va implícita cuando lo que se da en la línea de comandos es una *`dirección`*. También permite reactivar una interface que se ha desactivado temporalmente mediante la opción "`down`".

Esta opción corresponde a los indicadores `UP RUNNING`.

`down` Marca la interface como "down" o inactiva, es decir, inaccesible a la capa IP. Esto inhabilita cualquier tráfico IP a través de la interface. Es importante darse cuenta que esto no borra los registros de la tabla de encaminamiento correspondientes a esa interface de forma automática. Si pretende desactivar una interface de forma permanente, debería borrar estos registros de encaminamiento, aportando rutas alternativas si es posible.

`netmask` *`máscara`*

Esto asigna una máscara de subred a una interface. 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.

`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.

ifconfig confirma el establecimiento de una dirección punto-a-punto incluyendo 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 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`.)

`metric` *`número`*

Esta opción puede ser usada para asignar un valor de métrica a la tabla de encaminamiento creada para la interface. Esta métrica es usada por el Protocolo de Información de Encaminamiento (RIP, como ya hemos visto en capítulos anteriores) para construir las tablas de encaminamiento para la red. El valor usado por defecto por *ifconfig* es cero. Si no está ejecutando un demonio RIP, no necesita usar esta opción para nada; si por el contrario sí lo usa, al menos 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 interface es capaz de manejar en una única transacción. Para Ethernets, la MTU toma el valor 1500 por defecto; para interfaces tipo SLIP, el valor por defecto es 296.

`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 defecto.

ifconfig avisa que ARP ha sido inhabilitado mediante el indicador `NOARP`.

`-arp` Inhabilita el uso de ARP para esta interface.

`promisc` Pone la interface en modo promiscuo. En una red de difusión, esto hace que la interface 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 *fisgar*[9]. Se trata de una buena técnica para localizar problemas de red que de otra forma resultan difíciles.

[9] N. del T.: Del inglés *snooping*

Por otro lado, esto también posibilita ataques, permitiendo al atacante analizar el trafico de la red en busca de claves u otras cosas peligrosas. Una protección posible contra este tipo de ataques es impedir que cualquiera pueda conectarse a la Ethernet. Otra es la utilización de protocolos de autentificación seguros como Kerberos, o los programas SRA de ingreso en el sistema.[10]

Esta opción corresponde al indicador `PROMISC`.

`-promisc` Desactiva el modo promiscuo.

`allmulti` Las direcciones de envío múltiple son un tipo de difusión pero a un grupo de máquinas que no tienen necesariamente que pertenecer a la misma subred. El núcleo no soporta todavía direcciones de envío múltiple o de multidifusión[11].

Esta opción corresponde al indicador `ALLMULTI`.

`-allmulti` Desactiva las direcciones de envío múltiple.

5.9 Comprobación mediante *netstat*

A continuación describiré 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 routing table
Destination     Gateway         Genmask         Flags Metric Ref Use    Iface
127.0.0.1       *               255.255.255.255 UH    1      0       50 lo
191.72.1.0      *               255.255.255.0   U     1      0      478 eth0
191.72.2.0      191.72.1.1      255.255.255.0   UGN   1      0      250 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

[10]SRA se puede obtener del servidor **ftp.tamu.edu** en el directorio */pub/sec/TAMU*

[11]N.T.: Las versiones actuales del núcleo (2.X) sí permiten el uso de direcciones de envío múltiple.

si pretende evitar consultas para esos nombres a través de la red (por ejemplo consultas a un servidor NFS 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, el núcleo recorre la tabla registro a registro haciendo un "Y" 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 interface está activa.

H Esta interface permite el acceso a una sola máquina. Este es el caso de la interface de lazo **127.0.0.1** en nuestro ejemplo.

D Este indicador se activa cuando el registro es generado por un mensaje de redirección ICMP (ver sección 2.5).

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

La columna `Ref` de la salida de *netstat* muestra el número de referencias a esta ruta, esto es, cuantas otras rutas dependen de ésta (por ejemplo a través de pasarelas). Las dos últimas columnas muestran el número de veces que cada ruta ha sido usada y la interface que procesa los datagramas para dicha ruta.

5.9.2 Consulta de las Estadísticas de una Interface

Cuando se ejecuta con el indicador `-i`, *netstat* presenta las estadísticas de cada una de las interfaces de red configuradas en ese momento. Si se usa también la opción `-a`, el resultado son *todas* las interfaces presentes en el núcleo, no sólo aquéllas que ya han sido configuradas. En **vstout**, *netstat* producirá:

```
$ 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 informan de los valores de MTU y métrica configurados en la interface. Las columnas RX y TX muestran el total de paquetes recibidos y enviados respectivamente sin errores (RX-OK/TX-OK), dañados (RX-ERR/TX-ERR), cuántos han sido descartados (RX-DRP/TX-DRP), y cuántos se han perdido por desbordamiento (RX-OVR/TX-OVR).

La última columna muestra los indicadores activos para cada interface. Se trata de abreviaturas de una sola letra correspondientes a los nombres de los indicadores usados para configurar la interface mediante *ifconfig*.

B	Dirección de difusión activa.
L	Interface correspondiente a un dispositivo de lazo.
M	Recepción de todos los paquetes (modo promiscuo).
N	No se usan *pistas*[12]
O	ARP está desactivado en esta interface.
P	Se trata de una conexión punto-a-punto.
R	Interface en uso.
U	Interface activa.

5.9.3 Mostrar Conexiones

netstat soporta una serie de opciones que permiten mostrar los sockets[13] activos y pasivos. Las opciones `-t`, `-u`, `-w`, y `-x` muestran conexiones con sockets TCP, UDP, RAW, o UNIX. Si, adicionalmente incluye el indicador `-a`, también se muestran sockets en espera de una conexión (a la escucha). Esto le permite listar todos los servidores que se ejecutan en su sistema.

La ejecución de *netstat -ta* en **vlager** produce lo siguiente:

```
$ netstat -ta
Active Internet connections
Proto Recv-Q Send-Q Local Address    Foreign Address    (State)
```

[12]N.T.: Pistas aquí se refiere a un método de encapsulado de los datagramas IP en los que la información de la cabecera se coloca al final del paquete. Se usan en redes Ethernet para ayudar en el alineamiento de datos en los bordes de página.

[13]N. del T.: Literalmente significa enchufe en inglés, y se refiere a los descriptores de fichero abiertos para comunicaciones.

```
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
```

Vemos que la mayoría de los servidores están simplemente esperando una conexión entrante. Sin embargo, la cuarta línea muestra una conexión entrante SMTP desde **vstout**, y la sexta informa que hay una conexión saliente tipo *telnet* hacia **vbardolino**.[14]

El uso del indicador `-a` únicamente genera información de los sockets de todas las clases.

5.10 Comprobación de las Tablas ARP

En ciertas ocasiones, resulta útil poder ver o incluso alterar parte de las tablas ARP del núcleo, por ejemplo cuando sospecha que una dirección IP duplicada causa problemas intermitentes en su red. La herramienta *arp* fue creada con este objeto. Sus opciones son:

```
arp [-v] [-t tipohw] -a [maquina]
arp [-v] [-t tipohw] -s maquina direccionhw
arp [-v] -d maquina [maquina...]
```

Todos los argumentos ***maquina*** pueden ser nombres simbólicos o direcciones IP en notación de cuaterna.

Si usamos el primer comando, obtendremos el registro de la tabla correspondiente a la dirección IP o máquina especificada o, en el caso de que no se especifique ninguna, se muestran todas. Así, si ejecutáramos *arp* en **vlager** obtendríamos:

```
# arp -a
```

[14]Para saber si la conexión es entrante o saliente basta mirar los puertos. El puerto correspondiente a la máquina que llama es siempre un entero simple, mientras que la máquina receptora utiliza el puerto correspondiente al servicio en uso y que además es representado por *netstat* usando el nombre simbólico contenido en */etc/services*.

```
IP address      HW type              HW address
191.72.1.3      10Mbps Ethernet      00:00:C0:5A:42:C1
191.72.1.2      10Mbps Ethernet      00:00:C0:90:B3:42
191.72.2.4      10Mbps Ethernet      00:00:C0:04:69:AA
```

que muestra las direcciones Ethernet de **vlager**, **vstout** y **vale**.

Se puede usar la opción `-t` para mostrar la información referente a un tipo específico de hardware. Los valores posibles son *ether*, *ax25*, o *pronet*, y se refieren a Ethernet a 10Mbps, AMPR AX.25, y equipos token ring IEEE 802.5, respectivamente.

La opción `-s` se usa para añadir de forma permanente la dirección de hardware de ***maquina*** a las tablas ARP. ***direccionhw*** es la dirección de hardware y por defecto se supone que es Ethernet, especificada como una cadena de seis bytes en hexadecimal separados entre medias por dos puntos. Se puede también especificar la dirección de otro tipo de hardware usando la opción `-t`.

Un tipo de problema que puede requerir añadir una dirección IP manualmente a las tablas ARP es cuando, por alguna razón, una consulta ARP a una máquina remota falla, por ejemplo debido a que su controlador ARP no funciona correctamente o cuando alguna otra máquina en la red se identifica erróneamente como si ella misma tuviera esa dirección IP. También es un módo, aunque algo drástico, de protejerse frente a máquinas que, conectadas a la misma Ethernet, tratan de hacerse pasar por otras.

El uso de *arp* con el modificador `-d` borra todos los registros ARP que se refieran a la máquina dada. De este modo se puede forzar a una interface a que intente obtener de nuevo la dirección Ethernet que corresponda a la dirección IP en cuestión. Esto resulta útil cuando un sistema mal configurado ha emitido una información ARP incorrecta (por supuesto, primero habrá de asegurarse de que el error de configuración ha sido subsanado).

La opción `-s` se puede usar también para implementar ARP sustituto o *proxy* ARP. Se trata de una técnica especial en la que una máquina, digamos **gate**, actúa como pasarela para otra diferente llamada **fnord**, haciendo como que ambas direcciones pertenecen a la misma máquina, en este caso **gate**. Esto se consigue haciendo público un registro ARP para **fnord** que apunta a su propia interface Ethernet. De este modo, cuando cualquier máquina de la red realiza una consulta sobre **fnord**, **gate** responde con un registro que contiene su propia dirección Ethernet. La máquina que ha realizado la consulta enviará los datagramas a **gate**, quien se los pasa a **fnord**.

Este tipo de cosas puede ser necesario si, por ejemplo, pretende acceder a **fnord** mediante una máquina DOS que tiene una implementación de TCP incorrecta que no entiende el encaminamiento demasiado bien. Cuando usa ARP sustituto, a todos los efectos la máquina DOS ve a **fnord** en la subred local y, por tanto, no necesita preocuparse de cómo realizar el encaminamiento a través de una pasarela.

Otro aplicación muy útil del ARP sustituto es cuando una de sus máquinas actúa como pasarela para otra máquina aunque sólo de forma temporal, por ejemplo, en el caso de un enlace telefónico. En un ejemplo anterior, ya nos encontramos con el portátil **vlite** que se conectaba a **vlager** mediante un enlace PLIP de vez en cuando. Por supuesto, esto sólo funcionará si la dirección de la máquina para la que quiere actuar de sustituto ARP se encuentra en la misma subred IP que su pasarela. Así, por ejemplo, **vstout** podría ser el sustituto ARP de cualquier máquina de la subred de la Cervecera (**191.72.1.0**), pero nunca para máquinas de la subred de la Vinatera (**191.72.2.0**).

Abajo vemos el comando correcto para activar un ARP sustituto para **fnord**; 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 de ARP sustituto bastará:

```
# arp -d fnord
```

5.11 El Futuro

Las comunicaciones en red con Linux están en continua evolución. Cambios fundamentales en el núcleo permitirán un esquema de configuración muy flexible que permitirá que configure los dispositivos de red en tiempo de ejecución. Por ejemplo, *ifconfig* tendrá argumentos que permitan fijar la línea IRQ y el canal DMA.

Otro cambio que se espera pronto es el indicador adicional `mtu` en el comando *route*, que permita establecer la Unidad de Transferencia Máxima para una ruta en particular. Esta MTU específica de una ruta invalida el valor especificado para la interface. El uso típico de esta opción es para rutas a través de pasarelas, en las que el enlace entre la pasarela y la máquina destinataria requiere un MTU muy bajo. Por ejemplo, supongamos que la máquina **wanderer** esté conectada a **vlager** a través de un enlace SLIP. Entonces, al enviar datos de **vstout** a **wanderer**, la capa de red en **wanderer** enviaría paquetes de hasta 1500 bytes, porque los paquetes son transmitidos mediante una Ethernet. El enlace SLIP, sin embargo, opera con una MTU de 296, así que la capa de red en **vlager** tendría que dividir los paquetes IP en fragmentos más pequeños que quepan en 296 bytes. Si en vez de eso, configura la interface en **vstout** para que use una MTU de 296 desde el principio, se puede evitar el proceso de división, que es relativamente costoso.:

```
# route add wanderer gw vlager mtu 296
```

Debe notar que la opción `mtu` también permite que, de forma selectiva, evite los efectos de la política las 'Subredes son locales' (SNARL[15]). Se trata de una opción de configuración del núcleo descrita en el capítulo 3.

[15] Del inglés Subnets Are Local Policy

Capítulo 6

Servicio de nombres. Configuración

Como se comentó en el 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 el 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 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 *cache* 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. 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 Cricket Liu (vea [AlbitzLiu92]). Además, le interesará echar un vistazo a los comentarios adicionales que aparecen junto a las fuentes de su versión de BIND. También existe un grupo de *news* para cuestiones sobre DNS: el grupo **comp.protocols.tcp-ip.domains**.

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). Otras aplicaciones, como *smail*, pueden incluir diferentes rutinas para esto y necesitan cierto cuidado.

6.1.1 El fichero *host.conf*

El fichero *host.conf* es fundamental para controlar la configuración del sistema de resolución de nombres. 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, *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 el capítulo anterior, DNS le permite encontrar un nombre de máquina perteneciente a una dirección IP dada utilizando el dominio **in-addr.arpa**. Los intentos de los servidores de nombres de proporcionar un nombre falso se conocen en Inglés como “*spoofing*”[1]. 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*.

[1] N. del T.: literalmente, burla

alert Esta opción puede ir con las palabras *on* u *off*. Si se activa, cualquier intento de dar nombre falso 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.

Esta opción puede ir repetida con varios dominios, de modo que su máquina podría ser local a diversos dominios.

Un ejemplo de este fichero para la máquina **vlager** sería:

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

6.1.2 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*.

6.1.3 Configuración del fichero *resolv.conf*

Cuando se configura la librería 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.

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 hay ninguna línea *nameserver*, se intentará buscar el servidor en la propia máquina local.

Hay dos opciones más: *domain* y *search*, indicando la primera dominios alternativos a probar si la búsqueda inicial del nombre falla. Estos dominios irán separados por blancos o tabuladores.

Si no se incluye una opción *search*, se construirá una lista de búsqueda por defecto por el dominio local más todos los dominios padre hasta el raíz. El dominio local puede darse con la opción *domain*, y si no se da ninguno el sistema de resolución lo obtendrá mediante la llamada al sistema *getdomainname(2)*.

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:
nameserver      191.72.1.1
```

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

6.2 Ejecución de *named*

El programa que proporciona servicio de nombres en las máquinas UN⋆X suele ser *named*[2]. Es un servidor desarrollado inicialmente para Unix tipo BSD, con el propósito de proporcionar servicio de nombres a máquinas clientes y posiblemente a otros servidores de nombres.

La versión actualmente utilizada en casi todos los sistemas Linux es BIND-4.8.3. La nueva versión, BIND-4.9.3, está en este momento en versión Beta, y pronto estará disponible para Linux.

[2] N. del T.: Pronúnciese *n´eim-di:*

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 2, que le dará información acerca de cómo funciona básicamente el DNS.

El programa *named* suele iniciarse al arrancar la máquina, y ejecutarse hasta que se apaga. Obtiene la información que necesita de un fichero llamado */etc/named.boot*, y diversos ficheros que contienen datos acerca de nombres de dominio y direcciones, llamados *ficheros de zona*. Los formatos y semántica de estos ficheros serán explicados en la siguiente sección.

Para ejecutar *named*, solo 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.[3]

6.2.1 El fichero *named.boot*

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 la figura 6.1.[4]

Los comandos *cache* y *primary* sirven para cargar información en *named*. Esta información se obtiene de los ficheros especificados 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 el comando *primary*. La primera línea dice que *named* actúe como servidor principal para **vbrew.com**, tomando la información de zona del fichero *named.hosts*. El comando *directory* dice que todos los ficheros de zona se encuentran en el directorio indicado.

[3]Hay varios binarios de *named* disponibles en los servidores de FTP, cada uno configurado de forma diferente. Algunos anotan su fichero de número de proceso en el directorio */etc/*, otros en */tmp* y otros en */var/tmp*.

[4]Observar que los nombres de dominio del ejemplo se dan *sin* el punto final. Versiones anteriores del programa *named* parece que traten los puntos al final como errores y sin avisar descartan la línea afectada. En la versión BIND-4.9.3 se intenta arreglar este tema.

```
;
; Fichero /etc/named.boot para vlager.vbrew.com
;
directory     /var/named
;
;             dominio                 fichero
;---------------------------------------------------
cache         .                       named.ca
primary       vbrew.com               named.hosts
primary       0.0.127.in-addr.arpa    named.local
primary       72.191.in-addr.arpa     named.rev
```

Figura 6.1: El fichero *named.boot* para *vlager*.

La entrada iniciada con la palabra *cache* 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 *cache*, 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 un comando *primary* en cada fichero *named.boot*, para traducción inversa del IP **127.0.0.1**, que es el interface 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.

cache 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 el comando *cache* no existiera, *named* no haría una *cache* 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 *cache* 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.

6.2.2 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 *cache* 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 la figura 6.3 de la página 96, 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*[5]. 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.

[5]N. del T.: Time to Live

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.

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 *cache*. 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 será grande. Así, Craig Hunt ([Hunt92]) recomienda 42 días.

minimum Valor por defecto 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. Vea la sección 2.6 para obtener información de por qué es necesario. El campo de datos contiene el nombre del servidor. Para traducir ese nombre debe proporcionarse un registro A adicional, que se conoce como *glue record*, al proporcionar la dirección IP del servidor.

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 “Encaminado de correo en la Internet” del capítulo 13 se explica por qué son necesarios estos servidores. La sintáxis del registro MX es:

```
[domain] [ttl] [class] MX preference host
```

host es el nombre del servidor de correo para el dominio ***domain***. Cada servidor tiene un valor entero de preferencia (***preference***) asociado. Un agente de transporte de correo que desee entregar mensajes al dominio indicado en ***domain*** lo intentará con los servidores de estos registros hasta que uno responda. Se empieza probando con los de menor 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 utilizado. Existe un conjunto de convenciones sobre esto, el cual puede verse en el RFC 1340. Si el campo contiene blancos, debe encerrarse entre comillas dobles. El campo ***software*** especifica el software utilizado, para el que también existen convenciones en el mismo documento RFC.

6.2.3 Escribiendo los ficheros

Las figuras 6.2, 6.3, 6.4, y 6.5 son ejemplos de ficheros para un servidor de nombres en nuestra red ejemplo, localizado en la máquina *vlager*. El ejemplo es sencillo dada la simplicidad de nuestra red. Si tiene requisitos más complejos, léase el libro "DNS and BIND" de Cricket Liu y Paul Albitz ([AlbitzLiu92]).

El fichero *named.ca* mostrado en la figura 6.2 da ejemplos de registros de servidores raíz. Un fichero de *cache* típico suele tener información sobre una docena de servidores. Puede obtener la lista de servidores del dominio raíz mediante el programa *nslookup* descrito más adelante.[6]

```
;
; /var/named/named.ca          Fichero de cache.
;                No estamos en Internet, luego no necesitamos
;                servidores raiz. Elimine los puntos y coma
;                si desea activarlos.
;
; .                99999999   IN    NS  NS.NIC.DDN.MIL
; NS.NIC.DDN.MIL   99999999   IN    A   26.3.0.103
; .                99999999   IN    NS  NS.NASA.GOV
; NS.NASA.GOV      99999999   IN    A   128.102.16.10
```

Figura 6.2: Fichero *named.ca*.

[6]Esto no podrá hacerlo si no ha especificado algún servidor raíz. Puede en cambio ejecutar *nslookup* con un servidor diferente del suyo, o usar el fichero de ejemplo de la figura 6.2 y entonces obtener una lista de servidores válidos.

```
;
; /var/named/named.hosts       Maquinas locales en nuestra red
;                               El origen es vbrew.com
;
@                   IN  SOA   vlager.vbrew.com. (
                              janet.vbrew.com.
                              16         ; serial
                              86400      ; refresco: una vez al dia
                              3600       ; reintentos: una hora
                              3600000    ; expiracion:  42 dias
                              604800     ; minimo: 1 semana )
                    IN  NS    vlager.vbrew.com.
;
; el correo local se distribuye en vlager
                    IN  MX    10 vlager
;
; direccion de loopback
localhost.          IN  A     127.0.0.1
; Nuestra ethernet
vlager              IN  A     191.72.1.1
vlager-if1          IN  CNAME vlager
; vlager es tambien un servidor de USENET news
news                IN  CNAME vlager
vstout              IN  A     191.72.1.2
vale                IN  A     191.72.1.3
; Otra Ethernet
vlager-if2          IN  A     191.72.2.1
vbardolino          IN  A     191.72.2.2
vchianti            IN  A     191.72.2.3
vbeaujolais         IN  A     191.72.2.4
```

Figura 6.3: Fichero *named.hosts*.

```
;
; /var/named/named.local       Traduccion inversa para 127.0.0
;                               El origen es 0.0.127.in-addr.arpa.
;
@                   IN  SOA   vlager.vbrew.com. (
                              joe.vbrew.com.
                              1          ; serial
                              360000     ; refresco: 100 horas
                              3600       ; reintento:   1 hora
                              3600000    ; expiracion:  42 dias
                              360000     ; minimo: 100 horas )
                    IN  NS    vlager.vbrew.com.
1                   IN  PTR   localhost.
```

Figura 6.4: Fichero *named.local*.

```
;
; /var/named/named.rev         Traduccion inversa de nuestros numeros IP
;                               El origen es 72.191.in-addr.arpa.
;
@                   IN  SOA   vlager.vbrew.com. (
                              joe.vbrew.com.
                              16         ; serial
                              86400      ; refresco: una vez al dia
                              3600       ; reintento: una hora
                              3600000    ; expiracion:  42 dias
                              604800     ; minimo: 1 semana )
                    IN  NS    vlager.vbrew.com.
; nuestra red
1.1                 IN  PTR   vlager.vbrew.com.
2.1                 IN  PTR   vstout.vbrew.com.
3.1                 IN  PTR   vale.vbrew.com.
; la otra red
1.2                 IN  PTR   vlager-if1.vbrew.com.
2.2                 IN  PTR   vbardolino.vbrew.com.
3.2                 IN  PTR   vchianti.vbrew.com.
4.2                 IN  PTR   vbeaujolais.vbrew.com.
```

Figura 6.5: Fichero *named.rev*.

6.2.4 Comprobación del funcionamiento del servidor de nombres

Existe una utilidad que resulta interesante para comprobar el funcionamiento del servidor de nombres recién configurado. Se llama *nslookup*, y puede usarse tanto interactivamente como desde la línea de comandos. En el último caso, se invoca simplemente como:

```
nslookup nombre
```

y pedirá el nombre indicado al servidor de nombres que aparezca en *resolv.conf* (si aparece más de uno, *nslookup* cogerá uno al azar).

El modo interactivo, sin embargo, es mucho más interesante. Además de buscar máquinas por su nombre, se puede también preguntar por cualquier registro DNS, y transferir la información de zona completa de un dominio.

Cuando se invoca sin argumentos, *nslookup* mostrará el servidor de nombres en uso y entrará en modo interactivo. En el prompt '`>`' que se mostrará, puede teclear cualquier nombre de dominio por el que quiera preguntar. Por defecto, preguntará por registros de tipo A, es decir, aquellos que dan una dirección IP correspondiente al dominio introducido.

Esto se puede cambiar tecleando "`set type=`*tipo*", donde `tipo` es un nombre de registro de recurso (RR) como los descritos antes (en la sección 6.2) o bien la palabra ANY.

Por ejemplo, esta puede ser una sesión con *nslookup*:

```
$ nslookup
Default Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

> sunsite.unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

Non-authoritative answer:
Name:    sunsite.unc.edu
Address:  152.2.22.81
```

Si intenta preguntar por un nombre que no tiene dirección IP asociada, pero se encuentran otros registros relacionados en el DNS, el programa responderá con un error "`No type A records found`" (*no se encontraron registros de tipo A*). Sin embargo, puede hacer preguntas para otro tipo de registros sin más que usar el comando "`set type`". Por ejemplo, para obtener el registro SOA de **unc.edu**, podría escribir lo siguiente:

```
> unc.edu
*** No address (A) records available for unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

> set type=SOA
> unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

Non-authoritative answer:
unc.edu
        origin = ns.unc.edu
        mail addr = shava.ns.unc.edu
        serial = 930408
        refresh = 28800 (8 hours)
        retry   = 3600 (1 hour)
        expire  = 1209600 (14 days)
        minimum ttl = 86400 (1 day)

Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU       internet address = 128.109.157.30
```

De manera similar, se pueden pedir registros MX, etc. Y mediante la palabra ANY se obtendrán todos los RR asociados al nombre escrito.

```
> set type=MX
> unc.edu
Non-authoritative answer:
unc.edu preference = 10, mail exchanger = lambada.oit.unc.edu
lambada.oit.unc.edu     internet address = 152.2.22.80

Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU       internet address = 128.109.157.30
```

Una aplicación práctica de *nslookup* para la depuración es obtener la lista de servidores raíz para el fichero *named.ca*. Esto puede hacerse pidiendo todos los registros NS asociados al dominio raíz:

```
> set type=NS
> .
Name Server:  fb0430.mathematik.th-darmstadt.de
```

```
Address:  130.83.2.30

Non-authoritative answer:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL

Authoritative answers can be found from:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL
NS.INTERNIC.NET internet address = 198.41.0.4
AOS.ARL.ARMY.MIL        internet address = 128.63.4.82
AOS.ARL.ARMY.MIL        internet address = 192.5.25.82
AOS.ARL.ARMY.MIL        internet address = 26.3.0.29
C.NYSER.NET     internet address = 192.33.4.12
TERP.UMD.EDU    internet address = 128.8.10.90
NS.NASA.GOV     internet address = 128.102.16.10
NS.NASA.GOV     internet address = 192.52.195.10
NS.NASA.GOV     internet address = 45.13.10.121
NIC.NORDU.NET   internet address = 192.36.148.17
NS.NIC.DDN.MIL  internet address = 192.112.36.4
```

El conjunto completo de comandos disponibles en *nslookup* puede obtenerse con la orden interna `help`.

6.2.5 Otras utilidades interesantes

Hay algunas utilidades que pueden ayudarle en sus tareas de administrador de BIND. Describiremos dos de ellas. Por favor, eche un vistazo a la documentación que traen para saber cómo utilizarlas.

La utilidad *hostcvt* sirve para obtener una configuración inicial de BIND a partir del fichero */etc/hosts*. Genera tanto los ficheros de traducción directa (registros A) como los de traducción inversa (registros PTR) teniendo cuidado con los nombres de alias y otros. Por supuesto, no hará todo el trabajo, pues aun puede que necesite ajustar los registros SOA o añadir registros MX. Suponemos que también le ayudará tener cerca algunas aspirinas. El programa *hostcvt* forma parte de las fuentes de BIND, pero puede obtenerse por separado en algunos servidores FTP dedicados a Linux.

Después de configurar el servidor de nombres, puede que desee comprobar el resultado. La aplicación ideal para esto (al menos para mí) es el programa *dnswalk*, un paquete basado en *perl* que navega por la base de datos DNS, buscando errores habituales y verificando que la información es consistente. El programa *dnswalk* ha sido enviado recientemente al grupo **comp.sources.misc** de News, y debería estar en los servidores FTP que archiven este grupo (un servidor que seguro que lo tiene es **ftp.uu.net**).

Capítulo 7

SLIP: IP por línea serie

Los protocolos de línea serie, SLIP y PPP, permiten a los "pobres" tener conexión a Internet. Sólo se necesita un módem y un puerto serie con buffer FIFO. Utilizarlo no es más complicado que usar un buzón, y cada vez existen más proveedores que le ofrecen acceso telefónico IP a un coste asequible para todos.

En Linux hay controladores tanto de SLIP como de PPP. SLIP es más veterano y por tanto más estable. PPP para Linux ha sido recientemente desarrollado por Michael Callahan y Al Longyear; y se describirá en el próximo capítulo[1].

7.1 Requisitos generales

Para utilizar SLIP o PPP, hay que configurar algunas características de red que ya se han descrito en capítulos anteriores, por supuesto. Por lo menos, debe tener el interfaz de bucle (*loopback*) y el sistema de traducción de nombres. Cuando se conecte a Internet, querrá usar, por supuesto, el DNS. Lo más fácil es poner la dirección de algún servidor de nombres en el fichero *resolv.conf*; este servidor se usará tan pronto como SLIP conecte. Lo mejor es poner el servidor de nombres más cercano.

Sin embargo, esta solución no es la óptima, ya que las búsquedas de nombres seguirán yendo por la conexión SLIP o PPP. Si le interesa consumir menos ancho de banda, puede instalarse un servidor de nombres *sólo con cache*. No requiere un dominio ya que sólo actuará como *relevo*, es decir, pasará a otro servidor las peticiones que Vd. realice. La ventaja es que construirá una *cache* de modo que al pedir un nombre varias veces seguidas, sólo se contactará con el servidor externo la primera vez. Un fichero *named.boot* que sirva

[1]N. del T.: Actualmente, podemos decir que PPP ya es suficientemente estable. De hecho, SLIP cada vez se utiliza menos

para esto puede ser el siguiente:

```
; fichero named.boot para un servidor solo con cache
directory                           /var/named

primary       0.0.127.in-addr.arpa  db.127.0.0 ; interfaz "loopback"
cache         .                     db.cache   ; servidores raiz
```

Además debe tener un fichero *db.cache* con una lista de servidores raíz válidos. Este fichero está descrito al final del capítulo dedicado a la configuración del servidor de nombres.

7.2 Utilización de SLIP

Los servidores de IP por teléfono suelen ofrecer servicios SLIP mediante cuentas de usuario especiales. Después de entrar en una cuenta no se entra en un intérprete de comandos normal, sino en un programa o *shell script* que se ejecuta para activar el manejador SLIP del servidor y configurar la interfaz con la red. En ese momento tiene que hacer lo mismo en su máquina.

En algunos sistemas operativos, el manejador de SLIP es un programa de usuario, pero bajo Linux es parte del núcleo, cosa que lo hace mucho más rápido. Requiere, sin embargo, que la línea serie sea explícitamente convertida a modo SLIP. Esto se hace mediante una disciplina de línea especial llamada SLIPDISC. Mientras que un terminal (tty) está en modo normal (DISC0), intercambiará datos sólo con procesos de usuario, mediante las llamadas *read(2)* y *write(2)* habituales, y el manejador de SLIP no podrá escribir o leer del terminal. En el modo SLIPDISC se cambian los papeles: ahora los programas de usuario no podrán acceder a la línea pero todos los datos que lleguen se pasarán al manejador SLIP.

El manejador de SLIP entiende por sí mismo varias versiones del protocolo, incluyendo CSLIP, que realiza la llamada compresión de cabeceras de Van Jacobson en los paquetes IP salientes.[2] Esto aumenta el rendimiento de las sesiones interactivas. Además, hay versiones de seis bits de estos protocolos.

Una forma fácil de convertir una línea serie a modo SLIP es usar la utilidad *slattach*. Suponiendo que tenemos un módem en */dev/cua3* y que se ha entrado correctamente en el servidor de SLIP, se deberá ejecutar:

```
# slattach /dev/cua3 &
```

[2] La compresión de cabeceras de Van Jacobson se describe en el RFC 1441.

Esto cambiará el modo de línea de *cua3* a SLIPDISC, y la enganchará a uno de los interfaces SLIP disponibles. Si es la única conexión SLIP se enganchará al interface *sl1*, si es la segunda, a *sl2*, etc. Los núcleos actuales soportan hasta ocho enlaces SLIP simultáneos.

La encapsulación por defecto que elige *slattach* es CSLIP. Puede elegirse otra con la opción `-p`. Para usar SLIP sin compresión deberá ponerse:

```
# slattach -p slip /dev/cua3 &
```

Otros modos son `cslip`, `slip6`, `cslip6` (para la versión de 6 bits) y `adaptive` para SLIP adaptativo, que deja al núcleo averiguar qué encapsulación de SLIP usa el otro extremo de la comunicación.

Observe que debe utilizarse el mismo sistema de encapsulación que use el otro extremo. Por ejemplo, si **cowslip** usara CSLIP, tendrá que usarlo Vd. también. El síntoma típico de una selección incorrecta es que la orden *ping* a una máquina remota no tendrá respuesta. Si la otra máquina le hace *ping* a Vd, recibirá mensajes del tipo "`Can't build ICMP header`" *(no se puede construir la cabecera ICMP)* en la consola. Una forma de intentar evitar este tipo de problemas es usar SLIP adaptativo.

De hecho, *slattach* no sólo le permite activar SLIP, sino también otros protocolos serie como PPP o KISS (protocolo que se usa en packet-radio). Para más detalle, vea el manual en línea de *slattach(8)*.

Después de preparar la línea para SLIP, tendrá que configurar el interfaz de red. De nuevo, se hará esto mediante los programas estándares *ifconfig* y *route*. Suponiendo que desde la máquina **vlager** hemos llamado al servidor **cowslip**, se debería ejecutar:

```
# ifconfig sl0 vlager pointopoint cowslip
# route add cowslip
# route add default gw cowslip
```

El primer comando configura la interface como un enlace a **cowslip** punto a punto, mientras que el segundo y el tercero sirven para añadir la ruta correspondiente a **cowslip** como ruta por defecto y configurar esa máquina como pasarela de todos nuestros mensajes.

Cuando se quiera terminar el enlace SLIP, debe empezarse por eliminar todas las rutas a través de **cowslip** mediante el comando *route* con la opción `del`, desactivar el interface y enviar al proceso *slattach* la señal SIGHUP. Después de esto se deberá colgar el módem usando un programa de terminal de nuevo:

```
# route del default
```

```
# route del cowslip
# ifconfig sl0 down
# kill -HUP 516
```

7.3 Utilización de *dip*

Lo visto hasta ahora no es difícil de hacer. Sin embargo, puede que desee automatizar los pasos de modo que sólo tenga que invocar un comando. El programa *dip* hace esto.[3] La versión que existe en este momento es la 3.3.7. Ha sido parcheada por mucha gente, con lo que no podremos hablar simplemente de *el* programa *dip*. Las modificaciones serán incorporadas en futuras versiones.

dip tiene un intérprete de un lenguaje *script* sencillo que puede manejar automáticamente el módem, convertir la línea a modo SLIP y configurar las interfaces. Es bastante restrictivo por lo simple que es, pero suficiente para la mayoría de los casos. Una nueva versión de este programa podrá traer una versión más completa del lenguaje.

Para ser capaces de configurar el interfaz SLIP, *dip* necesita tener permisos de superusuario. Puede hacerse poniendo el programa con el bit *setuid* y de propiedad del usuario **root**, de modo que cualquier usuario sin privilegios podrá poner en marcha el programa. Esto es, sin embargo, muy peligroso, ya que una configuración incorrecta del encaminamiento de *dip* puede estropear el encaminamiento de su red local. Además, dará a los usuarios la posibilidad de conectarse a *cualquier* servidor SLIP, y lanzar ataques peligrosos a la red. Si aún quiere permitir a los usuarios activar conexiones SLIP, escriba pequeños programas para cada servidor de modo que cada uno invoque a *dip* con el *script* específico. Estos programas pueden tener privilegios sin peligro.[4]

7.3.1 Un script de ejemplo

Un script de ejemplo se encuentra en la figura 7.1. Puede utilizarse para conectarse a **cowslip** invocando a *dip* de esta forma:

```
# dip cowslip.dip
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation.

conectado a cowslip.moo.com with addr 193.174.7.129
#
```

[3] *dip* significa *Dialup IP* y fue escrito por Fred van Kempen.
[4] *diplogin* puede (y debe) ser ejecutado como superusuario. Vea la sección del final de este capítulo.

```
# Script de dip para conectarse al servidor cowslip

# Preparar nombres local y remoto
get $local vlager
get $remote cowslip

port cua3                 # seleccion de puerto serie
speed 38400               # poner velocidad maxima
modem HAYES               # poner tipo de modem
reset                     # reiniciar modem y terminal (tty)
flush                     # limpiar buffer de respuesta del modem

# Prepararse para marcado.
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

# Ahora ya estamos conectados
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 hey-jude\n
wait running 30
if $errlvl != 0 goto error

# Ahora ya estamos en la cuenta. Lancemos SLIP.
print Conectado a $remote with address $rmtip
default                   # Hacer que este enlace sea nuestra ruta por defecto
mode SLIP                 # Pasemos a modo SLIP
# en caso de error se ejecuta lo siguiente

error:
 print Fallo de la conextion SLIP con $remote .
```

Figura 7.1: Un script de ejemplo para *dip*

Después de conectar a **cowslip** y activar SLIP, *dip* pasará a ejecutarse en segundo plano. Ahora puede conectarse a través del enlace SLIP mediante los programas habituales de red. Para terminar la conexión, ejecute *dip* con la opción `-k`. Esto enviará una señal de colgar al proceso **dip**, cuyo número se encontrará almacenado en el fichero */etc/dip.pid*.[5]

```
# dip -k
```

En el lenguaje que interpreta *dip* las palabras precedidas con un signo de dólar se corresponden con nombres de variables. *dip* tiene un conjunto predefinido de variables que se listará a continuación. *$remote* y *$local*, por ejemplo, contienen los nombres de máquina local y remoto, respectivamente, involucrados en el enlace SLIP.

Las dos primeras sentencias del ejemplo son los comandos *get*, que sirven para establecer variables. Aquí, las máquinas local y remota han sido **vlager** y **cowslip**, respectivamente.

Las cinco sentencias que siguen preparan la línea serie y el módem. La palabra *reset* envía una cadena de reinicio al módem; que será el comando *ATZ* para módems compatibles con Hayes. La siguiente sentencia limpia el buffer de salida del módem, para conseguir que el diálogo de entrada (login y password) funcione correctamente. Este diálogo es extremadamente simple: llama al número 41988, que es el número de **cowslip**, entra en la cuenta *Svlager* mediante la clave de acceso *hey-jude*. El comando *wait* hace que se espere a la aparición de la cadena que sigue a esta orden, mientras que su segundo argumento especifica el tiempo de espera en segundos. Los comandos *if* sirven para ir comprobando la corrección del procedimiento de entrada en la cuenta.

Los comandos finales, ejecutados tras entrar en la cuenta, son *default*, que hace que el enlace SLIP sea la ruta por defecto para todos los destinos y *mode*, que pone la línea en modo SLIP y configura automáticamente el interface y la tabla de encaminamiento.

7.3.2 Guía de Referencia de *dip*

Aunque se utiliza mucho, *dip* aun no está muy documentado. En esta sección, daremos una pequeña guía de referencia de los comandos de *dip*. Puede obtenerse un resumen de los comandos ejecutando *dip* en modo de prueba (opción `-t`), e introduciendo el comando *help*. Para obtener ayuda sobre un comando se debe ejecutar sin argumentos; por supuesto esto no funcionará con comandos que no tengan argumentos.

```
$ dip -t
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation.
```

[5] Vea el grupo **alt.tla** si desea conocer más abreviaturas en Inglés que sean palíndromas, como este caso.

```
DIP> help
DIP knows about the following commands:

databits default  dial     echo     flush
get      goto     help     if       init
mode     modem    parity   print    port
reset    send     sleep    speed    stopbits
term     wait

DIP> echo
Usage: echo on|off
DIP> _
```

En los siguientes apartados, los ejemplos que muestran el *prompt* `DIP>` indican cómo se introduciría el comando en modo prueba, y qué salida produciría. Los ejemplos que no muestren este *prompt* deben tomarse como trozos de *scripts*.

Comandos del módem

Existe un conjunto de comandos de *dip* pensados para configurar la línea serie y el módem. Algunos son de uso obvio, como *port*, que sirve para elegir el puerto serie, y *speed*, *databits*, *stopbits*, y *parity*, que establecen los parámetros habituales de las líneas serie.

El comando *modem* selecciona el tipo de módem. Actualmente sólo está soportado el tipo *HAYES*[6]. Debe decirse el tipo, pues si no *dip* se negará a ejecutar los comandos *dial* y *reset*. Este último comando envía la cadena de reinicio al módem, la cual depende del tipo de módem elegido. Para modems compatibles Hayes, esta cadena es *ATZ*.

La orden *flush* puede utilizarse para vaciar las respuestas anteriores de la memoria del módem. De otro modo, un *script* de diálogo con el módem podría fallar, porque lea la respuesta *OK* que proceda de órdenes anteriormente enviadas al módem.

El comando *init* selecciona la cadena de inicialización enviada al módem antes de marcar, que para modems Hayes es, por defecto, la cadena "*ATE0 Q0 V1 X1*", que activa el eco de los comandos, hace que el módem dé los códigos de resultado en modo extendido (es decir, por palabras y no números de código) y selecciona marcado a ciegas, sin esperar tono de marcado.

El comando *dial* envía la cadena de inicialización al módem y llama al sistema remoto. El comando de marcado por defecto en los modems Hayes es *ATD*.

[6]Deben respetarse las mayúsculas

Comandos *echo* y *term*

El comando *echo on* se usa con propósitos de depuración, ya que hace que *dip* copie en la consola todo lo que envíe al puerto serie. Puede desactivarse después con una orden *echo off*.

dip también puede salirse temporalmente a un modo terminal, de modo que Vd. pueda dialogar manualmente con el módem. Para ello se usa el comando *term*, y para salir de este modo se pulsa Ctrl-].

Comando *get*

La orden *get* sirve para poner valores a las variables internas. Puede usarse como se vio en ejemplos anteriores, o bien de forma interactiva, añadiendo la palabra *ask*:

```
DIP> get $local ask
Enter the value for $local: _
```

Un tercer uso de este comando es intentar obtener el valor de la máquina remota. Aunque extraño pueda parecer, resulta útil en muchos casos: muchas veces el servidor SLIP no permite que nosotros nos pongamos cualquier dirección IP, sino que nos la asignará de un conjunto predeterminado y nos informará de ello mediante una frase tal como "`Your address: 193.174.7.202`" (*Su dirección: 193.174.7.202*). De esta frase querremos que *dip* ajuste automáticamente nuestra dirección IP, para lo que haremos lo siguiente (observar que se usa el parámetro *remote*):

```
... dialogo de entrada en la cuenta ....
wait address: 10
get $locip remote
```

Comando *print*

Este es el comando para enviar textos a la consola. Cualquier variable puede enviarse a la consola mediante comandos de este tipo, por ejemplo:

```
DIP> print Utilizando puerto $port con velocidad $speed
Utilizando puerto cua3 con velocidad 38400
```

Nombres de las variables

dip sólo entiende un conjunto predefinido de variables. Un nombre de variable siempre empieza con un símbolo de dólar y debe escribirse en minúsculas.

Las variables *$local* y *$locip* contienen respectivamente el nombre de nuestra máquina y su dirección IP. Poniendo el nombre de la máquina, *dip* guardará dicho nombre en la variable *$local*, al tiempo que guardará la dirección IP en la variable *$locip*.

Las variables *$remote* y *$rmtip* hacen lo mismo pero con la máquina remota. Por otro lado, *$mtu* contiene el valor del MTU para la conexión actual.

Esas cinco variables son las únicas que pueden actualizarse mediante un comando *get*. Otras deben actualizarse mediante comandos específicos, aunque siempre pueden sacarse por pantalla con el comando *print*. Esas variables son *$modem*, *$port*, y *$speed*.

La variable *$errlvl* sirve para conocer el resultado del último comando ejecutado, siendo de valor 0 si fue bien, o distinto de 0 si hubo algún problema.

Comandos *if* y *goto*

El comando *if* es un salto condicional. Su sintaxis es

```
if variable oper numero goto etiqueta
```

donde la expresión puede ser una simple comparación entre una de las variables siguientes: *$errlvl*, *$locip*, y *$rmtip*. El segundo operando debe ser un número entero; el operador `oper` puede ser uno de los siguientes: `==`, `!=`, `<`, `>`, `<=`, y `>=`.

El comando *goto* lanza la ejecución a partir de la situación de la etiqueta, que debe ponerse al principio de una línea seguida de dos puntos.

Comandos *send*, *wait* y *sleep*

Estos comandos ayudan a implementar sencillos *scripts* de diálogo. *send* envía su argumento a la línea serie. No pueden ponerse variables, pero entiende secuencias de escape al estilo del lenguaje C, como \n y \b. El carácter de tilde (~) puede usarse como abreviatura del retorno de carro.

El comando *wait* hace que *dip* espere a que por la línea serie se reciba la palabra pasada como primer argumento. El segundo argumento, que es opcional, fija un tiempo de espera máximo, en segundos. Si la palabra no se recibe en ese tiempo, el comando fallará actualizando la variable *$errlvl* con el valor 1.

La orden *sleep* puede usarse para esperar cierto tiempo, por ejemplo para esperar pacientemente una invitación a entrar en la cuenta. Una vez más, el tiempo se especificará en segundos.

Comandos *mode* y *default*

Se utilizan para cambiar el puerto entre modo SLIP o normal, y configurar la interface.

El comando *mode* es el último que ejecuta *dip* antes de pasar al segundo plano (daemon). A menos que suceda un error, de este comando no se retorna.

Este comando tiene un argumento, que es el protocolo. Actualmente se reconocen los protocolos *SLIP* y *CSLIP*, es decir, la versión actual de *dip* no soporta SLIP adaptativo.

Después de poner la línea en modo SLIP, *dip* ejecutará un comando *ifconfig* para configurar la interface como enlace punto a punto y otro *route* para cambiar las tablas de encaminamiento apuntando a la máquina remota.

Si, además, se ejecuta *default* antes que *mode*, el programa hará que el camino por defecto de nuestros paquetes vaya al enlace SLIP.

7.4 Funcionamiento en Modo Servidor

Curiosamente, configurar su máquina como servidor SLIP va a ser mucho más sencillo que configurarla como cliente.

Una forma de hacerlo es usar *dip* en modo servidor, que puede conseguirse si se ejecuta como *diplogin*. Su configuración principal se encontrará en */etc/diphosts*, que asocia nombres de cuenta con direcciones de máquina asignadas. Alternativamente, puede usar *sliplogin*, una utilidad procedente de BSD que proporciona un esquema de configuración que le permite ejecutar *shell scripts* cuando las máquinas se conectan y desconectan. Actualmente, su desarrollo está en fase beta.

Ambos programas necesitan que se tenga una cuenta por cada cliente SLIP. Por ejemplo, si proporcionáramos un servicio SLIP a Arthur Dent en **dent.beta.com**, debería crearse una cuenta **dent** añadiendo la siguiente línea al fichero */etc/passwd*:

```
dent:*:501:60:Cuenta SLIP de Arthur Dent:/tmp:/usr/sbin/diplogin
```

Luego, se pondría la clave usando el programa *passwd*.

Ahora, cuando **dent** entre, *dip* entrará en modo servidor. Para comprobar si está autorizado para usar SLIP, buscará su nombre de usuario en */etc/diphosts*. Este fichero detalla derechos de acceso y parámetros de conexión para cada usuario. Una entrada de este fichero será tal como:

```
dent::dent.beta.com:Arthur Dent:SLIP,296
```

El primer campo de los separados por dos puntos, es el nombre de la cuenta. El segundo campo puede contener una clave adicional (vea más adelante). El tercero es el nombre o dirección IP de la máquina llamante. El siguiente es un campo informativo acerca del usuario, por el momento sin utilidad. Por último, se describen separados por comas los parámetros de la conexión: el protocolo (*SLIP* o *CSLIP*) seguido del valor de MTU.

Cuando **dent** entra en su cuenta, *diplogin* extrae la información acerca de él y si hay clave de acceso en la línea correspondiente de */etc/diphosts*, la pedirá como "clave externa de seguridad", que se compara con la existente en el fichero (que no va encriptada). Si no coinciden, el intento de entrada será rechazado.

En otro caso, *diplogin* procederá a cambiar el modo a SLIP o CSLIP, y preparará la interfaz y el encaminamiento. Esta conexión permanecerá hasta que el módem opuesto cuelgue, momento en que *diplogin* dejará la línea en modo normal y terminará.

diplogin necesita privilegios de superusuario. Si no tiene puesto el bit setuid, deberá copiar el programa con el nombre *diplogin* y ponerle a éste los privilegios. A *diplogin* se le pueden dar sin miedo, sin afectar al estado de *dip* en sí mismo.

Capítulo 8

El Protocolo Punto_a_Punto (PPP)

8.1 Desenredando las Pes

Al igual que el SLIP, el PPP es un protocolo utilizado para enviar datagramas a través de una conexión serie, pero mejora algunas de las carencias del anterior. El PPP permite a las partes comunicantes negociar al principio de la conexión opciones como las direcciones IP y el tamaño máximo de los datagramas, y proporciona mecanismos de autentificación de los clientes. Para cada una de estas capacidades, el PPP tiene un protocolo concreto. A continuación, describiremos brevemente estos bloques básicos que constituyen el PPP. Esta descripción está muy lejos de ser completa; si quiere saber más sobre el PPP, lea sus especificaciones en el RFC 1548, así como en la docena de RFCs que le acompañan.[1]

En la parte más baja del PPP está el protocolo de *Control de Conexión de Datos de Alto-Nivel*, abreviadamente HDLC[2] [3], que define los límites de las tramas PPP individuales, y proporciona un control de errores de 16 bit. Al contrario de lo que ocurría con SLIP, una trama PPP es capaz de llevar paquetes de otros protocolos distintos al IP, como el IPX de Novell o el Appletalk. El PPP consigue esto añadiendo a la trama básica HDLC un campo de control que identifica el tipo de paquete contenido en la misma.

El LCP, Protocolo de Control de Enlace[4], 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.

[1] Los RFCs más relevantes están listados en la Bibliografía al final de este libro.
[2] N. del T.: Del inglés High-Level Data Link Control
[3] En realidad, el HDLC es un protocolo mucho más general publicado por la Organización Internacional de Estándares (ISO).
[4] N. del T.: Del inglés Link Control Protocol

Un paso importante en la configuración del enlace PPP corresponde a la autentificación de los clientes. Aunque no es obligatorio, es casi un deber para las líneas telefónicas. Normalmente el servidor pide al cliente que se identifique probando que se sabe alguna clave secreta. Si el llamante se equivoca, la conexión se termina. Con el 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. Se llaman el Protocolo de Autentificación por Contraseña, o PAP, y el Protocolo de Autentificación por Reto, o CHAP[5].

Cada protocolo de red que es encaminado a través de la conexión de datos, como el IP, el Appletalk, etc; es configurado 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 IPCP[6], el Protocolo de Control del IP.

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 meter las cabeceras de los paquetes TCP en un espacio de tan solo 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.

8.2 PPP en Linux

En el Linux, la funcionalidad del PPP está dividida en dos partes, un controlador de HDLC de bajo nivel situado en el kernel, y el demonio *pppd* del espacio del usuario que controla los diferentes protocolos de control. La versión actual del PPP para Linux es la *linux-ppp-1.0.0*, y contiene el módulo PPP para el kernel, el *pppd*, y un programa llamado *chat* utilizado para llamar al sistema remoto.

El controlador del PPP para el kernel fue escrito por Michael Callahan. El *pppd* fue escrito a partir de una implementación gratuita del PPP para máquinas Sun y 386BSD que a su vez fue escrita por Drew Perkins y otros programadores, y mantenida por Paul Mackerras. Fue transportada a Linux por Al Longyear.[7] El *chat* fue escrito por Karl Fox.[8]

[5] N. del T.: Del inglés Challenge Handshake Authentication Protocol

[6] N. del T.: Del inglés IP Control Protocol

[7] Los dos autores han dicho que van a estar muy ocupados por bastante tiempo. Así que si tiene alguna pregunta sobre el PPP en general, mejor pregunte a la gente en el canal de la lista de correo de los "Linux activists" en la RED.

[8] **karl@morningstar.com**.

Al igual que el SLIP, el PPP está implementado a través de una disciplina especial para la utilización de las líneas. Para utilizar una línea de 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 del PPP, que comprueba la validez de las tramas que llegan (cada trama HDLC trae un código de control de errores de 16 bit), las descompone y las despacha. Actualmente, es capaz de controlar datagramas IP, utilizando opcionalmente la compresión de cabeceras Van Jacobson. Tan pronto como Linux acepte IPX, el controlador PPP será ampliado para poder controlar también los paquetes IPX.

El controlador del kernel es ayudado por el *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 del *pppd*, sino solamente darle una introducción. Para más información, lea las páginas de manual y los ficheros *README* de la distribución con las fuentes del *pppd*, que deberían ayudarle a comprender la mayor parte de las cuestiones que este capítulo no trata. 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*.

8.3 Conexiones con *pppd*

Cuando quiere conectarse a Internet a través de un enlace PPP, tiene que configurar las capacidades básicas de red como el dispositivo de loopback y el sistema de resolución de direcciones. Las dos han sido explicadas en los capítulos previos. Hay algunas cosas que es necesario decir sobre la utilización del DNS en un enlace serie; por favor, lea el capítulo del SLIP para más información.

Como ejemplo introductorio de cómo establecer una conexión PPP con el *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, se ejecuta el siguiente comando:

```
# pppd /dev/cua3 38400 crtscts defaultroute
```

Esto cambiará a la línea de serie *cua3* 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 el *pppd* tras ejecutarse es negociar varias características para el enlace con el extremo remoto utilizando el LCP. Normalmente, el conjunto de opciones que intenta negociar el *pppd* funcionará, así que no nos meteremos más con este asunto. Volveremos a tratar el LCP con más detalle en alguna sección posterior.

Hasta ahora, también hemos asumido que **c3po** no necesita ninguna autentificación de nosotros, así que la fase configuración habrá sido completada con éxito.

El *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* se lo comunicará al otro extremo.

Normalmente no habrá ningún problema con esta configuración por defecto. Incluso si su máquina está en una Ethernet, puede utilizar la misma dirección IP tanto para la Ethernet como para el interface PPP. No obstante, el *pppd* le permite utilizar direcciones diferentes, o incluso pedir a su compañero que utilice alguna dirección específica. Estas opciones serán discutidas más adelante.

Tras pasar por la fase de configuración IPCP, el *pppd* configurará la red de su ordenador para utilizar el enlace PPP. En primer lugar, configurará el interface de red PPP como un enlace punto-a-punto, utilizando el *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 por defecto apunte a **c3po**, debido a que lo especificamos con la opción *defaultroute*.[9] Esto provoca que todos los datagramas dirigidos a ordenadores que no estén en su red sean enviados a **c3po**. Hay un variado número de formas de encaminamiento que acepta el *pppd*, y las cubriremos en mayor detalle más adelante.

8.4 Los Ficheros de Opciones

Antes de que el *pppd* procese los argumentos de su línea de comandos, echa un vistazo a varios ficheros para establecer sus opciones por defecto. Estos ficheros pueden contener cualquier argumento de línea de comando válido, distribuido 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*

[9] El encaminamiento por defecto es instalado sólamente si no hay ninguno establecido de antes.

arranca. El utilizarlo para establecer algunas opciones globales por defecto 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 PPP 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.

El otro fichero de opciones, que es leído después del */etc/ppp/options*, es el *.ppprc* situado en el directorio *home* del usuario. Permite que cada usuario especifique su propio conjunto de opciones por defecto.

Un fichero */etc/ppp/options* de ejemplo puede parecerse a éste:

```
# Opciones globales para el pppd de vlager.vbrew.com
auth                # obligar a autentificacion
usehostname         # usar el nombre del ordenador local para el CHAP
lock                # usar el bloqueo de dispositivo tipo UUCP
domain vbrew.com    # nombre de nuestro dominio
```

Las dos primeras opciones se utilizan para la autentificación y serán explicadas a continuación. La expresión `lock` hace que el *pppd* utilice el método de bloqueo de dispositivos de UUCP. De esta manera, cada proceso que accede a un dispositivo serie, por ejemplo el */dev/cua3*, crea un fichero de bloqueo llamado *LCK..cua3* en el directorio de spool del UUCP para señalizar que ese dispositivo está siendo usado. Esto es necesario para evitar que otros programas, como pueden ser el *minicom* o el *uucico*, abran el dispositivo de serie mientras éste es usado por el PPP.

La razón de poner éstas opciones en el fichero de configuración global es que éstas no pueden ser pasadas por alto, de forma que proporcionan un razonable nivel de seguridad. Pero tenga en cuenta que, a pesar de todo, algunas opciones podrán ser pasadas por alto más tarde; un ejemplo de esto es la cadena *connect*.

8.5 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 el *pppd*. Al contrario que el *dip*, el *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 al *pppd* con la opción *connect* en la línea de comando. El *pppd* redirigirá la entrada y salida estándar de comandos a la línea de serie. Un programa útil para esto es el *expect*, escrito por Don Libes. Tiene un lenguaje muy potente basado en el Tcl, y fue diseñado exactamente para este tipo de aplicación.

El paquete *pppd* incluye un programa similar llamado *chat* que le permite especificar un script del estilo de los de UUCP. Básicamente, un script del 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 del chat:

```
ogin: b1ff ssword: s3kr3t
```

Esto le indica al *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 *login* empiece por **l** mayúscula o minúscula, o si llega con basura. La siguiente cadena es una cadena esperada que hace que el *chat* espere al mensaje de petición de contraseña y la envíe nuestra contraseña como respuesta.

Esto es básicamente lo que tienen los scripts del chat. Un script completo para llamar a un servidor PPP debería, además, incluir los comandos 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 del *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 al chat tiene que ser una cadena esperada, pero como el módem no dirá nada hasta que hablemos con él, hacemos que el *chat* la ignore especificando una cadena vacía. Continuamos enviando `ATZ`, el comando de inicializaión para los modems compatibles Hayes, y esperamos a que nos responda con `OK`. La siguiente cadena envía al *chat* el comando 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 login. El resto del script del chat funciona exactamente como antes.

La opción `-v` hace que el *chat* capture todas las actividades hacia la facilidad *local2* del demonio *syslog*.[10]

El escribir el script de chat directamente en la línea de comando implica un cierto riesgo, pues los usuarios pueden ver la línea de comando de un proceso con el comando

[10] Si edita el *syslog.conf* para redirigir estos mensajes a un fichero, asegúrese de que éste fichero no pueda ser leído por cualquiera, pues el *chat* también captura todo el script de entrada por defecto – incluyendo las contraseñas.

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. Por lo tanto la invocación completa al *pppd* tendrá ahora un aspecto como éste:

```
# pppd connect "chat -f dial-c3po" /dev/cua3 38400 -detach \
crtscts modem defaultroute
```

Además de la opción *connect* que se refiere al script de llamada, hemos añadido dos opciones más a la línea de comando: *-detach*, que le indica al *pppd* que no se separe de la consola ni se vuelva proceso de segundo plano. La palabra *modem* activa algunas acciones específicas para módem sobre el dispositivo de serie, como colgar la línea antes y después de la llamada. Si no utiliza esta opción, el *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; el *chat* permite el uso de scripts mucho más complejos. Una característica muy útil es la capacidad de especificar cadenas frente a las cuales parar el 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 no descuelga. Para hacer que el *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 ...
```

De una forma parecida, puede variar el valor del tiempo de espera para algunas partes de los scripts de chat insertando opciones `TIMEOUT`. Para más detalles, vea la página de manual del *chat(8)*.

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 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 esperada del script. Consiste en en una secuencia de 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 del chat de la siguiente manera:

```
ogin:-BREAK-ogin: ppp ssword: GaGariN
```

Ahora, cuando el *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.

8.6 Depuración de la Configuración PPP

Por defecto, el *pppd* registrará todos los avisos y mensajes de error gracias a las facilidades *daemon* del *syslog*. Tiene que añadir una entrada al *syslog.conf* que redirija esto a un fichero, o incluso a la consola, pues de otra forma el *syslog* simplemente desechará estos mensajes. La siguiente entrada envía todos los mensajes a */var/log/ppp-log*:

```
daemon.*                /var/log/ppp-log
```

Si la configuración de su PPP no funciona, echar un vistazo a este fichero le debería dar una primera pista de qué es lo que va mal. Si esto no le ayuda, también puede activar la salida extra de depuración utilizando la opción `debug`. Esto hace que el *pppd* registre los contenidos de todos los paquetes de control enviados o recibidos a *syslog*. Todos los mensajes irán a la facilidad *daemon*.

Finalmente, la opción más drástica es el activar la depuración a nivel de kernel llamando al *pppd* con la opción *kdebug*. Esto es seguido por un argumento numérico que es el O exclusivo (*xor*) de los siguientes valores: 1 para mensajes de depuración generales, 2 para imprimir los contenidos de todas las tramas HDLC que nos llegan, y 4 para hacer al controlador imprimir todas las tramas HDLC salientes. Para capturar los mensajes de depuración a nivel de kernel, tiene que, o bien ejecutar un demonio *syslogd* que lea el fichero */proc/kmsg*, o si no el demonio *klogd*. Cualquiera de los dos dirige la depuración del kernel a la facilidad *kernel* del *syslogd*.

8.7 Opciones de Configuración IP

El IPCP se utiliza para negociar un par de 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 por defecto, y a qué valor. Tras la recepción, el extremo remoto inspecciona cada opción sucesivamente, y, o responde que la acepta, o la rechaza.

El *pppd* le da gran control sobre qué opciones intentará negociar el IPCP. Puede ajustar esto a través de varias opciones en la línea de comandos de las que hablamos a continuación.

8.7.1 Elección de las Direcciones IP

En el ejemplo anterior, hacíamos que el *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. Algunas veces, sin embargo, es útil el tener control sobre la dirección utilizada por alguno de los extremos de la conexión. El *pppd* soporta diferentes posibilidades sobre este aspecto.

Para pedir direcciones particulares, normalmente dé al *pppd* la siguiente opción:

```
dir_local:dir_remota
```

donde `dir_local` y `dir_remota` pueden ser especificadas en notación de cuádruplas numéricas o como nombres de ordenador. [11] Esto hace al *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.[12]

Si sólo quiere establecer la dirección local, y aceptar cualquier dirección que utilice el compañero, simplemente deseche la parte de la `dir_remota`. 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 comando. De forma similar, para establecer la dirección remota únicamente, dejaría el campo de la `dir_local` en blanco. Por defecto, el *pppd* utilizará entonces la dirección asociada al nombre de su ordenador.

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. Cuando llame a uno de éstos servidores, debe asegurarse de que el *pppd* no solicita una dirección IP particular, sino que acepta la dirección que el servidor le pide que utilice. Esto quiere decir que no tiene que poner el argumento `dir_local`. Además, tendrá que utilizar la opción `noipdefault`, que hace que el *pppd* espere a que el compañero le proporcione la dirección IP en vez de utilizar la dirección IP del ordenador local.

8.7.2 Encaminamiento a través de una Conexión PPP

Tras configurar el interface de red, el *pppd* preparará un encaminamiento que sólamente 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.

---

[11] El utilizar nombres de ordenador en esta opción tiene algunas consecuencias a la hora de la autentificación utilizando CHAP. Puede echar un vistazo a la sección sobre CHAP más adelante.

[12] Puede permitir al otro PPP sobreescribir sus ideas de direcciones IP dando al *pppd* las opciones `ipcp-accept-local` e `ipcp-accept-remote`. Eche un vistazo a la página del manual para más detalles.

Ya hemos visto antes que se puede pedir al *pppd* que configure el encaminamiento por defecto 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 un gateway 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 **loner**. Cuando esté conectando a **vlager** a través de PPP, él utiliza una dirección de la subred de la Cervecera. Podremos dar al *pppd* del ordenador **vlager** la opción `proxyarp`, que instalará una entrada proxy-ARP para el ordenador **loner**. Esto hará que **loner** 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 específica, porque estas redes tendrán ya sus propios encaminamientos por defecto. Por otra parte, el tener a 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 **191.72.3.0**, 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 **sub-etha** y tiene una dirección IP de **191.72.3.1.**.

Cuando **sub-etha** conecta a **vlager**, hará que el punto de encaminamiento por defecto 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 **sub-etha**. Para esto, utilizamos una característica del *pppd* de la que no hemos hablado hasta ahora – el comando *ip-up*. Es un script de shell situado en */etc/ppp* que se ejecuta después de que el interface PPP ha sido configurado. Cuando está presente, se le llama con los siguientes parámetros:

```
ip-up interface dispositivo velocidad dir_local dir_remota
```

donde ***`interface`*** se refiere al interface de red usado, ***`dispositivo`*** es la ruta al dispositivo serie utilizado, (*/dev/tty* si se utiliza la salida y entrada estándar), y ***`velocidad`*** es la velocidad del dispositivo. ***`dir_local`*** y ***`dir_remota`*** nos dan las direcciones IP usadas en dos extremos de la conexión en notación de cuarteto numérico. En nuestro caso, el script *ip-up* puede contener el siguiente fragmento de código:

```
#!/bin/sh
case $5 in
191.72.3.1)           # este es sub-etha
route add -net 191.72.3.0 gw 191.72.3.1;;
...
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.

A pesar de todo, la tabla de encaminamiento aún no está completa. Hemos configurado las entradas de la tabla de encaminamiento para las 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 por defecto encaminado a **sub-etha**, y todos los ordenadores de la Cervecera encaminan hacia **vlager** por defecto. Si éste no fuera el caso, su única posibilidad normalmente será usar un demonio de encaminamiento como el *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.

8.8 Opciones de Control de Enlace

Anteriormente, ya hemos tratado sobre el 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 de configuración LCP más, 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, es usado 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[13]. 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*). El PPP le permite rehuir de cualquiera de los caracteres con códigos ASCII comprendidos entre 0 y 31 especificándolos en el mapa asíncrono.

[13]N. del T.: Estos caracteres se conocen como *rehuidos*

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 más significativo corresponde al ASCII 31. Si un bit se pone a 1, indica que el caracter correspondiente debe de ser rehuido antes de ser enviado a través de la conexión. Inicialmente, el mapa asíncrono se establece como *0xffffffff*, lo que significa que todos los caracteres de control serán rehuidos.

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 rehuidos, utilice la siguiente opción:

```
asyncmap 0x000A0000
```

La unidad máxima de recepción, o MRU, señaliza al otro extremo el tamaño máximo de las tramas HDLC que queremos recibir. Aunque ésto 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 kernel, y describe el tamaño máximo de la trama que el interface 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 interface ha de ser capaz de recibir tramas de hasta 1500 bytes.

Por lo tanto, elegir un MRU no es tanto una cuestión de qué 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 (está activada por defecto).

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 está 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 está aún operativa. Puede habilitar esto 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`. Por defecto, esta característica también está desactivada.

8.9 Consideraciones Generales sobre Seguridad

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 del PPP segura.

Uno de los problemas del *pppd* es que el configurar el dispositivo de red y la tabla de encaminamiento requiere los privilegios de **root**. Normalmente resolverá esto ejecutándolo como *setuid* de **root**. A pesar de ello, el *pppd* permite a los usuarios establecer varias opciones de relevancia para la seguridad. Para protegerse contra cualquier ataque que pueda lanzar algún usuario manipulando estas opciones, se sugiere que establezca un par de valores por defecto en el fichero global */etc/ppp/options*, tal como los mostrados en el fichero de ejemplo en la sección "Utilización de los Ficheros de Opciones". 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.

Por supuesto, también tiene que protegerse de los sistemas con los que habla con 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 tratatá sobre estos asuntos.

8.10 Autentificación con PPP

8.10.1 CHAP frente a PAP

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.

El PAP trabaja básicamente de la misma forma que el procedimiento normal de *login*. 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. Esta técnica es vulnerable a los intrusos que pueden intentar obtener la contraseña escuchando en una línea de serie y a otros que hagan sucesivos intentos de ataque por el método de prueba y error.

El CHAP no tiene estos defectos. Con el 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 llega al mismo resultado.

Otra característica del 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.

El *pppd* mantiene las claves secretas para el CHAP y el PAP en dos ficheros separados, llamados */etc/ppp/chap-secrets* y *pap-secrets* respectivamente. Si introduce un ordenador remoto en alguno de los dos ficheros, tiene un buen control de cual de los protocolos CHAP o PAP se utilizará para autentificarnos con él y viceversa.

Por defecto, el *pppd* no pide autentificación al ordenador remoto, pero aceptará el autentificarse a sí mismo cuando se lo pida el ordenador remoto. Como el CHAP es mucho más fuerte que el PAP, el *pppd* intenta usar el anterior siempre que es posible. Si el otro ordenador no lo acepta, o el *pppd* no encuentra una clave CHAP para el sistema remoto es 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`, el *pppd* solicitará al otro ordenador que se autentifique. El *pppd* aceptará el uso del CHAP o el 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 en 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 el 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 del PPP, *pap-secrets* y *chap-secrets*. Están situados en */etc/ppp* y contienen tripletas 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 el CHAP y el PAP, y también depende de si nos autentificamos nosotros con el otro ordenador, o si solicitamos al servidor que se autentifique con nosotros.

8.10.2 El fichero de claves CHAP

Cuando tiene que autentificarse con algún servidor utilizando el CHAP, el *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 del *chap-secrets* para **vlager**:[14]

```
# claves CHAP para vlager.vbrew.com
#
# cliente         servidor          clave                 direccion
#---------------------------------------------------------------------
vlager.vbrew.com  c3po.lucas.com    "Use The Source Luke" vlager.vbrew.com
c3po.lucas.com    vlager.vbrew.com  "riverrun, pasteve"   c3po.lucas.com
*                 vlager.vbrew.com  "VeryStupidPassword"  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 del CHAP. El *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**,[15] y encuentra la primera línea mostrada anteriormente. Entonces produce la respuesta del 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, el *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 del CHAP de la manera que acabamos de decir, y se la devuelve a **vlager**. El *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

[14] Las comillas no son parte de la contraseña, simplemente sirven para proteger el espacio en blanco del interior de la contraseña.

[15] Este nombre de ordenador se toma del reto del CHAP.

segunda línea se corresponde con esto, así que el *pppd* combina el reto del CHAP y la clave `riverrun, pasteve`, las encripta, y compara el resultado con la respuesta del 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 de cuarteto numérico o como nombres de ordenador que son resueltos posteriormente. Por ejemplo, si **c3po** solicita usar una dirección IP que no está en esta lista durante la negociación IPCP, la petición será rechazada, y 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 válido 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 el *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 defecto 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:remoto`*", y *`local`* es un nombre en vez de un cuarteto numérico, el *pppd* utilizará éste como el nombre local. Para más detalles, lea la página del manual del *pppd(8)*.

8.10.3 El Fichero de Claves PAP

El fichero de claves PAP es muy similar al utilizado por el 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, el *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, el *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
#
# usuario       servidor        clave           direccion
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 cómo 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 defecto, el *pppd* tomará el nombre del ordenador local como el nombre de usuario, pero también se puede especificar un nombre diferente dando la opción `user`, seguida por el nombre deseado.

Para escoger una de las entradas del fichero *pap-secrets* para la autentificación con el compañero, el *pppd* tiene que saber el nombre del ordenador remoto. Como no tiene manera de averiguarlo, tiene que especificarlo en la línea de comando 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 comando 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 del CHAP. El otro ordenador sólo podrá pedir direcciones de esa lista. En el fichero de ejemplo, obligamos a **c3po** a usar su dirección IP auténtica.

Dese cuenta de que el 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 está interesado en utilizar el PAP, encontrará algunas características más de éste comentadas en la página del manual del *pppd(8)*.

8.11 Configuración de un Servidor PPP

Hacer funcionar el *pppd* como servidor es sólo cuestión de añadir las opciones adecuadas en la línea de comando. Idealmente, crearía una cuenta especial, digamos **ppp**, y le adjudicaría un script o programa como shell de entrada que llame al *pppd* con estas opciones. Por ejemplo, podría añadir la siguiente línea a */etc/passwd*:

```
ppp:*:500:200:Cuenta PPP Publica:/tmp:/etc/ppp/ppplogin
```

Por supuesto, puede usar uids y gids diferentes a los mostrados arriba. También tendrá que establecer la contraseña para la cuenta de arriba usando el comando *passwd*.

El script *ppplogin* tendrá entonces este aspecto:

```
#!/bin/sh
# ppplogin - script para lanzar el pppd al entrar
mesg n
stty -echo
exec pppd -detach silent modem crtscts
```

El comando *mesg* deshabilita la opción de que otros usuarios puedan escribir a la terminal (*tty*) usada utilizando, por ejemplo, el comando *write*. El comando *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, y `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 comando del *pppd*, o en el fichero de opciones globales. La página del manual también habla sobre opciones más específicas para activar o desactivar los protocolos de autentificación individuales.

Capítulo 9

Algunas Aplicaciones de Red

Después de instalar correctamente el IP y el sistema de resolución, tiene que dedicarse a los servicios que quiera proporcionar a través de la red. Este capítulo trata la configuración de algunas sencillas aplicaciones de red, incluyendo el servidor *inetd*, y los programas de la familia *rlogin*. El interface de *Llamada a Procedimiento Remoto* o RPC[1], en el que están basados servicios como el *Sistema de Ficheros en Red* o NFS[2] y el *Sistema de Información de Red* o NIS[3], también será tratado brevemente aquí. Las configuraciones de NFS y de NIS, sin embargo, ocupan más espacio y serán descritas en capítulos aparte. Lo mismo sucede con el correo electrónico y el sistema de noticias.

Por supuesto, no podemos cubrir todas las aplicaciones de red en este libro. Si desea instalar alguna no tratada aquí, como *talk*, *gopher*, o *Xmosaic*, por favor, refiérase a su documentación.

9.1 El Super-Servidor *inetd*

Frecuentemente, los servicios son llevados a cabo por los llamados *demonios*. Un demonio es un programa que abre un determinado puerto, y espera a recibir peticiones de conexión. Si se recibe una petición de conexión, lanza un proceso hijo que aceptará la conexión, mientras el padre continua escuchando a la espera de más peticiones. Este concepto tiene el inconveniente de que por cada servicio ofrecido, se necesita ejecutar un demonio que escuche las conexiones a un puerto, lo que generalmente significa un desperdicio de recursos de sistema como, por ejemplo, de espacio de intercambio.

Por ello, casi todas las instalaciones UN⋆X corren un "super-servidor" que crea sockets

[1] N. del T.: Del inglés Remote Procedure Call

[2] N. del T.: Del inglés Network File System

[3] N. del T.: Del inglés Network Information System.

para varios servicios, y escucha en todos ellos simultáneamente usando la llamada al sistema *select(2)*. Cuando un nodo remoto requiere uno de los servicios, el super-servidor lo recibe y llama al servidor especificado para ese puerto.

El super-servidor más usado es *inetd*, el demonio Internet. Es iniciado en tiempo de arranque del sistema, y toma la lista de servicios que debe tratar de un fichero de configuración denominado */etc/inetd.conf*. Aparte de esos servidores invocados por *inetd*, hay varios servicios triviales que el propio *inetd* se encarga de llevar a cabo, denominados *servicios internos*. Entre ellos, el *chargen* que simplemente genera una cadena de caracteres, y el *daytime* que devuelve la fecha y hora del sistema.

Una entrada de este fichero consiste en una única línea compuesta por los siguientes campos:

```
servicio tipo protocolo espera usuario servidor linea_de_comando
```

El significado de cada campo es como sigue:

`servicio` Proporciona el nombre del servicio. El nombre del servicio debe ser traducido a un número de puerto consultando el fichero */etc/services*. Este fichero será descrito más adelante en la sección 9.3.

`tipo` Especifica un tipo de socket, ya sea *stream* (para protocolos orientados a la conexión) o *dgram* (para protocolos no orientados a la conexión). Los Servicios basados en TCP deberán, por lo tanto, usar siempre *stream*, mientras que los servicios basados en UDP deberán usar siempre *dgram*.

`protocolo` Indica el protocolo de transporte usado por el servicio. Éste debe ser un nombre de protocolo válido que se pueda encontrar en el fichero *protocols*, también descrito más adelante.

`espera` Esta opción se aplica sólo a sockets de tipo *dgram*. Puede tomar los valores *wait* o *nowait*. Si se especifica *wait*, *inetd* ejecutará sólo un servidor cada vez para el puerto especificado. De otro modo, continuará escuchando por el puerto inmediatamente después de ejecutar el servidor.

Esto es útil para servidores "single-threaded" que leen todos los datagramas que entran hasta que no llegan más, y después acaban. La mayor parte de los servidores RPC son de este tipo y se deberá por ello especificar *wait*. El otro tipo de servidores, los "multi-threaded", permiten un número ilimitado de instancias corriendo concurrentemente. Con estos servidores se deberá especificar *nowait*.

Para sockets de tipo *stream* se deberá especificar siempre *nowait*.

`usuario` Éste es el identificador del usuario bajo el que se ejecutará el proceso. Por lo general, éste suele ser el usuario **root**, aunque algunos servicios pueden usar diferentes cuentas. Es una buena idea el aplicar aquí el principio del menor privilegio, que indica que uno no debería ejecutar un comando bajo una cuenta privilegiada si el programa no lo requiere para funcionar correctamente. Por ejemplo, el servidor de noticias NNTP se ejecutará como **news**, mientras que otros servicios que podrían significar un riesgo para la seguridad (como *tftp* o *finger*) son normalmente ejecutados como **nobody**.

`servidor` Proporciona el camino completo del programa servidor a ejecutar. Los servicios internos se indican con la palabra *internal*.

`linea_de_comando`
Ésta es la línea de comando a pasar al servidor. Esto incluye el argumento 0, es decir, el nombre del comando. Normalmente, éste será el nombre de programa del servidor, salvo que el programa se comporte de forma distinta cuando se le invoque con un nombre diferente.

Este campo se deja vacío para los servicios internos.

En la figura 9.1 se muestra un ejemplo de fichero */etc/inetd.conf*. La línea del servicio *finger* está comentada, de forma que no esté disponible. Esto se suele hacer normalmente por razones de seguridad porque podría ser usado por atacantes para obtener nombres de usuarios del sistema.

El *tftp* también se muestra deshabilitado. *tftp* implementa el *Trivial File Transfer Protocol* que permite transferir cualquier fichero del sistema que tenga permiso de lectura global sin chequeo de passwords, etc. Esto es especialmente peligroso con el fichero */etc/passwd*, sobre todo si no se usa *shadow password*.

TFTP es usado comúnmente por clientes y terminales X sin unidad de discos para obtener su *software* de un servidor de arranque. Si necesita ejecutar *tftp* por ésta razón, asegúrese de limitar su acción a los directorios de los que los clientes obtendrán los ficheros añadiendo esos nombres de directorio a la línea de comando del *tftpd*. Esto se muestra en la segunda línea *tftp* del ejemplo.

9.2 La herramienta de control de acceso *tcpd*

Ya que abrir un ordenador al acceso en red implica muchos riesgos de seguridad, las aplicaciones están diseñadas para protegerse ante varios tipos de ataques. Algunas de éstas aplicaciones, sin embargo, pueden ser reventadas (lo que quedó bastante demostrado con el *RTM Internet worm*), o pueden no distinguir entre un nodo seguro cuyas peticiones de

```
#
# servicios inetd
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
#
#       servicios internos inetd
#
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
```

Figura 9.1: Un ejemplo de fichero */etc/inetd.conf*.

un servicio particular deberían ser aceptadas, y otro nodo que no lo es y cuyas peticiones deberían ser rechazadas. Ya hemos discutido brevemente los servicios *finger* y *tftp* más arriba. Así, uno podría querer limitar el acceso a esos servicios solamente a los "nodos de confianza", lo cual es imposible con la configuración usual, donde *inetd* o proporciona un servicio a todos los clientes, o a ninguno.

Una herramienta útil para esto es *tcpd*,[4] el denominado *demonio envoltorio*[5]. Para los servicios TCP que quiera monitorizar o proteger, éste es invocado en lugar del programa servidor. *tcpd* informa de la petición al demonio *syslog*, chequea si el nodo remoto está autorizado para usar ese servicio, y sólo si la respuesta es satisfactoria, ejecutará el programa servidor real. Observe que esto no funciona con servicios basados en UDP.

Por ejemplo, para proteger el demonio *finger*, debe cambiar la línea correspondiente en *inetd.conf* así:

```
# Proteger el demonio de finger
finger  stream  tcp    nowait  root   /usr/sbin/tcpd   in.fingerd
```

Así, sin añadir ningún control de acceso, esto le hará parecer al cliente que es la típica configuración de *finger*, salvo que toda petición será registrada en la facilidad *auth* del *syslog*.

El control de acceso está implementado mediante dos ficheros llamados */etc/hosts.allow* y */etc/hosts.deny*. Estos ficheros contienen entradas permitiendo y denegando acceso, respectivamente, para ciertos servicios y nodos. Cuando *tcpd* trata una petición de un servicio como *finger* de un nodo cliente denominado **biff.foobar.com**, busca en *hosts.allow* y *hosts.deny* (en éste orden) una entrada en la que el servicio y el nodo cliente coincidan. Si la entrada coincidente aparece en *hosts.allow*, se garantiza el acceso, sin importar lo que haya en *hosts.deny*. Si la coincidencia se encuentra en *hosts.deny*, la petición se rechaza cerrando la conexión. Si no hay coincidencia en ninguno, la petición es aceptada.

Las entradas en los ficheros de acceso tienen la siguiente estructura:

```
lista_servicios:  lista_nodos [:cmd_shell]
```

`lista_servicios` es una lista de nombres de servicios de */etc/services*, o la palabra clave *ALL*. Para especificar todos los servicios excepto *finger* y *tftp*, usa "*ALL EXCEPT finger, tftp*".

`lista_nodos` es una lista de nombres de nodos o direcciones IP, o las palabras clave *ALL*, *LOCAL*, o *UNKNOWN*. *ALL* hace coincidir todos los nodos mientras que *LOCAL*

[4]Escrita por Wietse Venema, **wietse@wzv.win.tue.nl**.
[5]N. del T.: Del inglés Wrapper Daemon

hace coincidir todos los nombres de nodos que no contengan un punto[6]. *UNKNOWN* hace coincidir todos los nodos cuya búsqueda de nombre o dirección falló. Un nombre comenzado por un punto incluye a todos los nodos cuyo dominio es el mismo a ese nombre. Por ejemplo, **.foobar.com** encajará con **biff.foobar.com**. También hay formas de especificar direcciones de red IP y números de subred. Por favor, refiérase a la página del manual de *hosts_access(5)* para más detalles.

Para denegar acceso a los servicios *finger* y *tftp* a todos los nodos menos a los locales, ponga lo siguiente en */etc/hosts.deny*, y deje */etc/hosts.allow* vacío:

```
in.tftpd, in.fingerd:  ALL EXCEPT LOCAL, .su.dominio
```

El campo opcional ***cmd_shell*** puede contener un comando de shell para que sea invocado cuando una búsqueda coincida con la entrada. Esto es útil para establecer trampas que puedan delatar a atacantes potenciales:

```
in.ftpd: ALL EXCEPT LOCAL, .vbrew.com : \
      echo "peticion de %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* al nombre del nodo cliente y al nombre del servicio, respectivamente. Refiérase a la página del manual de *hosts_access(5)* para más detalles.

9.3 Los ficheros *services* y *protocols*

Los números de puerto en los que se ofrecen ciertos servicios "estándar" están definidos en el RFC de "Números Asignados"[7]. Para permitir a los programas cliente y servidor convertir nombres de servicios en estos números, se almacenan en un fichero llamado */etc/services*. Una entrada se construye así:

```
servicio puerto/protocolo [aliases]
```

Aquí, ***servicio*** especifica el nombre del servicio, ***puerto*** define el puerto por el que se ofrece el servicio, y ***protocolo*** define qué protocolo de transporte se usa. Comúnmente,

[6]Normalmente sólo los nombres de nodos locales obtenidos de búsquedas en */etc/hosts* no contienen puntos.

[7]N. del T.: A veces se conocen como *Well Known Ports*, es decir, *Puertos Bien Conocidos*

éste es *udp* o *tcp*. Es posible que un servicio sea ofrecido a más de un protocolo, lo mismo que es posible ofrecer distintos servicios por el mismo número de puerto, siempre que el protocolo sea distinto. El campo ***aliases*** permite especificar nombres alternativos para el mismo servicio.

Usualmente, no se necesita cambiar el fichero de servicios que viene con el software de red en su sistema Linux. De todas formas, presentaremos un pequeño extracto de ese fichero.

```
# El fichero services:
#
# servicios conocidos (well-known)
echo           7/tcp                # Eco
echo           7/udp                #
discard        9/tcp  sink null     # Descartar
discard        9/udp  sink null     #
daytime       13/tcp                # Fecha del sistema
daytime       13/udp                #
chargen       19/tcp  ttytst source # Generador de caracteres
chargen       19/udp  ttytst source #
ftp-data      20/tcp                # Protocolo FTP de ficheros (Datos)
ftp           21/tcp                # Protocolo FTP de ficheros (Control)
telnet        23/tcp                # Protocolo de Terminal
smtp          25/tcp                # Protocolo de Correo
nntp         119/tcp  readnews      # Protocolo de Noticias
#
# servicios UNIX
exec         512/tcp                # rexecd de BSD
biff         512/udp  comsat        # Notificacion de correo
login        513/tcp                # login remoto
who          513/udp  whod          # who y uptime remotos
shell        514/tcp  cmd           # comando remoto, si contrase~na
syslog       514/udp                # registro remoto del sistema
printer      515/tcp  spooler       # cola de impresion remota
route        520/udp  router routed # informacion de encaminamiento
```

Observe que, por ejemplo, el servicio *echo* es ofrecido en el puerto 7 tanto para TCP como para UDP, y que el puerto 512 es usado para dos servicios diferentes; el demonio COMSAT (que notifica a los usuarios de correo recién llegado, vea *xbiff(1x)*), mediante UDP, y la ejecución remota (*rexec(1)*), usando TCP.

Ocurre algo similar con el fichero de protocolos: la librería de red necesita una forma de convertir nombres de protocolo — por ejemplo, los usados en el fichero services — a números de protocolo entendibles por el nivel IP en otros nodos. Esto se hace buscando el nombre en el fichero */etc/protocols*. Contiene una entrada por línea, cada una con un nombre de protocolo y el número asociado. Necesitar modificar éste fichero es todavía más improbable que tener que hurgar en */etc/services*. Le mostramos un fichero ejemplo:

```
#
# Internet (IP) protocols
#
ip      0       IP              # protocolo internet, pseudo-protocolo
icmp    1       ICMP            # protocolo de mensajes de control
igmp    2       IGMP            # protocolo para mensajes multidestino
tcp     6       TCP             # protocolo de control de transmision
udp     17      UDP             # protocolo de datagramas de usuario
raw     255     RAW             # interfaz IP directa (modo "crudo")
"
```

9.4 Llamada a Procedimientos Remotos

Un mecanismo muy general para aplicaciones cliente-servidor lo proporciona RPC, el paquete *Remote Procedure Call*. RPC fue desarrollado por Sun Microsystems, y es una colección de herramientas y funciones de librería. Ejemplos de aplicaciones construidas sobre RPC son NFS, el sistema de ficheros en red, y NIS, el sistema de información de red, que serán presentados en próximos capítulos.

Un servidor RPC consiste en una colección de procedimientos a los que el cliente puede llamar enviando una petición RPC al servidor, junto con los parámetros del procedimiento. El servidor invocará al procedimiento indicado en nombre del cliente, devolviendo el valor del resultado, si lo hay. Para que sea independiente de la plataforma, todos los datos intercambiados entre el cliente y el servidor son convertidos al formato denominado de *Representación Externa de Datos* o XDR[8] por el segundo, y convertidos otra vez a la representación de la máquina local por el receptor.

A veces, las mejoras en una aplicación RPC introducen cambios incompatibles en el interface de llamada a procedimiento. Por supuesto, sólo cambiando el servidor dejaría de funcionar cualquier aplicación que todavía espere el comportamiento original. Por ello, los programas RPC tienen numeros de versión asignados, normalmente empezando con 1, y con cada nueva versión del interface RPC éste contador se incrementará. A menudo, un servidor puede ofrecer varias versiones a la vez; entonces los clientes indicarán en sus peticiones mediante el número de versión qué implementación del servicio desean usar.

[8] N. del T.: del inglés eXternal Data Representation

La comunicación por red entre servidores y clientes RPC es un poco peculiar. Un servidor RPC ofrece una o más colecciones de procedimientos; cada conjunto de éstos es llamado *programa*, y es identificado unívocamente por un *número de programa*. En */etc/rpc* se suele mantener una lista que mapea nombres de servicios con números de programa, reproducimos un extracto de éste en la figura 9.2.

```
#
# /etc/rpc - servicios variados 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
```

Figura 9.2: Un ejemplo de fichero */etc/rpc*.

En redes TCP/IP, los autores de RPC se encontraron con el problema de mapear números de programa a servicios de red genéricos. Decidieron que cada servidor proporcionara ambos, un puerto TCP y otro UDP, para cada programa y para cada versión. Generalmente, las aplicaciones RPC usarán UDP cuando envíen datos, y sólo recaerán en TCP cuando los datos a transferir no quepan en un datagrama UDP sencillo.

Por supuesto, los programas clientes tienen que tener una forma de encontrar a qué puerto mapea un número de programa. Usando un fichero de configuración para esto sería muy inflexible: ya que las aplicaciones RPC no usan puertos reservados, no hay garantías de que un puerto originalmente pensado para ser usado por nuestra aplicación de base de datos no haya sido cogido por algún otro proceso. Por lo tanto, las aplicaciones RPC escogen cualquier puerto que puedan utilizar, y lo registran con el denominado *demonio mapeador de puertos*[9]. Éste último actúa como un distribuidor de servicios para todos los servidores que corren en su máquina: un cliente que desee contactar con un servicio que tiene un número de programa dado, preguntará primero al mapeador de puertos del nodo del servidor quién devolverá los números de puerto TCP y UDP por los que el servicio puede ser accedido.

Este método tiene como mayor inconveniente que introduce un punto de ruptura único, muy parecido al que crea el demonio *inetd* en los servicios Berkeley estándar. De todas

[9] N. del T.: Del inglés Portmapper daemon

formas, éste caso es un poco más grave, porque cuando el mapeador de puertos cae, toda la información de puertos RPC se pierde; esto normalmente implica que hay que rearrancar todos los servidores RPC manualmente, o rearrancar toda la máquina.

En Linux, el mapeador de puertos se llama *rpc.portmap* y reside en */usr/sbin*. Aparte de asegurarse de que es arrancado desde *rc.inet2*, el mapeador de puertos no necesita más trabajo de configuración.

9.5 Configurar los Comandos *r*

Hay varios comandos para ejecutar programas en nodos remotos. Son *rlogin*, *rsh*, *rcp* y *rcmd*. Todos ellos lanzan un shell en el nodo remoto y permiten al usuario ejecutar comandos. Por supuesto, el cliente necesita tener una cuenta en el nodo en el que se van a ejecutar los comandos. Por ello todos estos comandos llevan a cabo un procedimiento de autorización. Normalmente, el cliente indicará el nombre de login del usuario al servidor, el cual requerirá un password que será validado de la forma habitual.

A veces, sin embargo, es deseable el relajar estos chequeos de autorización para ciertos usuarios. Por ejemplo, si usted tiene que entrar frecuentemente en otras máquinas de su LAN, tal vez desee ser admitido sin tener que escribir su password cada vez.

Deshabilitar autorizaciones sólo es aconsejable en un número reducido de nodos cuyas bases de datos de passwords estén sincronizadas, o para un número reducido de usuarios privilegiados que necesiten acceder a muchas máquinas por razones administrativas. Siempre que desee permitir a gente entrar en su nodo sin tener que especificar un login o password, debe asegurarse de que no permite acceso accidentalmente a nadie más.

Hay dos formas de deshabilitar chequeos de autorización para los comandos *r*. Una es que el super usuario permita a ciertos o a todos los usuarios el entrar, sin ser preguntados por un password, en ciertos o en todos los nodos (lo cual es ciertamente una mala idea). Este acceso es controlado por un fichero denominado */etc/hosts.equiv*. Este contiene una lista de nodos y nombres de usuarios que son considerados equivalentes a usuarios en el nodo local. Una opción alternativa es que un usuario permita acceso a otros usuarios de ciertos nodos a su cuenta. Éstos serían listados en el fichero *.rhosts* en el directorio *home* del usuario. Por razones de seguridad, este fichero debe pertenecer al usuario o al super usuario, y no debe ser un enlace simbólico, de otro modo será ignorado.[10]

Cuando un cliente pide un servicio *r*, su nodo y nombre de usuario son buscados en el fichero */etc/hosts.equiv*, y después en el fichero *.rhosts* del usuario con cuyo nombre se

[10] En un entorno NFS, podría necesitar darle una protección de 444, porque el superusuario por lo general está muy restringido en el acceso a ficheros en discos montados via NFS.

pretende entrar. Como ejemplo, asumamos que **janet** está trabajando en **gauss** e intenta entrar en la cuenta de **joe** en **euler**. A partir de ahora, nos referiremos a Janet como el usuario cliente, y a Joe como el usuario *local.* Ahora, cuando Janet escriba

```
$ rlogin -l joe euler
```

en **gauss**, el servidor primero chequeará en *hosts.equiv*[11] si a Janet se le puede proporcionar acceso libre y, si esto falla, intentará localizarla en el fichero *.rhosts* del directorio *home* de **joe**.

El fichero *hosts.equiv* en **euler** es algo así:

```
gauss
euler
-public
quark.physics.groucho.edu     andres
```

Una entrada consiste en un nombre de nodo, seguido opcionalmente por un nombre de usuario. Si aparece un nombre de nodo y nada más, todos los usuarios de ese nodo serán admitidos en sus cuentas locales sin ninguna comprobación. En el ejemplo anterior, Janet hubiera sido autorizada a entrar en su cuenta **janet** si llamaba desde **gauss**, y lo mismo se aplicaría a cualquier otro usuario exceptuando a **root**. De todas formas, si Janet desea entrar como **joe**, se le pediría un password como siempre.

Si un nombre de nodo va seguido de un nombre de usuario, como en la última línea del fichero ejemplo, a ese usuario se le permite acceso libre de password a *todas* las cuentas excepto a la cuenta **root**.

El nombre de nodo también puede ir precedido de un signo menos, como en la entrada "**-public**". Esto requiere autorización para todas las cuentas en **public**, sin importar lo que permitan los usuarios individuales en sus ficheros *.rhosts.*

El formato del fichero *.rhosts* es idéntico al del *hosts.equiv*, pero su significado es un poco diferente. Consideremos el siguiente fichero *.rhosts* de Joe en **euler**:

```
chomp.cs.groucho.edu
gauss      janet
```

La primera entrada permite a **joe** acceso libre cuando entra desde **chomp.cs.groucho.edu**, pero no afecta a los permisos de ninguna otra cuenta en

[11] Observe que el fichero *hosts.equiv no* es examinado cuando alguien intenta entrar como **root**.

euler o **chomp**. La segunda entrada es una pequeña variación de esto en que permite a **janet** acceso libre a la cuenta de Joe cuando entra desde **gauss**.

Observe que el nombre de nodo del cliente se obtiene mediante la resolución inversa de la dirección del que llama a un nombre, de forma que esta característica fallará con nodos deconocidos para el sistema de resolución. El nombre de nodo del cliente se considera que coincide con el nombre en los ficheros de nodos en uno de los siguientes casos:

- El nombre canónico del cliente (no un alias) coincide literalmente con el nombre de nodo en el fichero.
- Si el nombre de nodo del cliente es un nombre de dominio completamente cualificado (como el devuelto por el sistema de resolución cuando se tiene DNS en marcha), y no coincide literalmente con el nombre de nodo en el fichero de nodos, se compara con el nombre de nodo que se forma al expandirlo con el nombre de dominio local.

Capítulo 10

El Sistema de Información de Red (NIS)

Cuando se usa una red de área local, su objetivo fundamental es, normalmente, proporcionar a sus usuarios un entorno que haga a la red transparente. Para este fin una importante piedra de toque es mantener datos vitales, como la información de cuentas de usuario, sincronizadas entre todos los nodos. Vimos anteriormente que para resolver nombres de nodos existe un potente y sofisticado servicio denominado DNS. Para otras tareas, sin embargo, no existe un servicio especializado similar. Más aún, si usted sólo está administrando una pequeña LAN sin conexión a Internet, puede que no le merezca la pena el esfuerzo de instalar un DNS.

Esta es la razón por la que Sun desarrolló NIS, el *Sistema de Información de Red*. NIS proporciona facilidades de acceso genérico a bases de datos que pueden ser usadas para distribuir información como la contenida en los ficheros *passwd* y *groups* entre todos los nodos de su red. Esto hace que la red aparezca como un sistema único, con las mismas cuentas en todos los nodos. De forma similar usted puede usar NIS para distribuir el fichero de información de nombres de nodos */etc/hosts* entre todas las máquinas de la red.

NIS está basado en RPC, e incluye un servidor, una biblioteca para la parte del cliente, y varias herramientas de administración. Originalmente NIS se llamaba *Yellow Pages*[1], o YP, que todavía son términos ampliamente usados para referirse informalmente a este servicio. Por otra parte Yellow Pages es una marca registrada de la compañía British Telecom, la cual pidió que Sun dejara de utilizar ese nombre. Pero, como algunos nombres impactan mucho entre la gente, YP continúa viviendo como prefijo en los nombres de la mayoría de los comandos relacionados con NIS como *ypserv*, *ypbind*, etc.

[1]N. del T.: Páginas Amarillas

Hoy en día NIS está disponible en prácticamente todos los sistemas UN⋆X, y hay incluso implementaciones gratuitas de él. Una de ellas es de la edición BSD Net-2, derivada de una implementación de referencia de dominio público donada por Sun. El código de la biblioteca cliente de esta versión existe en la GNU *libc* desde hace mucho tiempo, mientras que los programas de administración han sido recientemente portados a Linux por Swen Thümmler.[2] Falta, sin embargo, un servidor NIS en la implementación de referencia. Tobias Reber ha escrito otro paquete NIS incluyendo todas las herramientas y un servidor; se llama *yps*.[3]

Actualmente, el código de NIS está siendo reescrito por completo por Peter Eriksson[4], se denominará NYS y soportará tanto el NIS normal como la revisión ampliada de Sun, el NIS+. NYS no sólo proporciona un conjunto de herramientas NIS y un servidor, sino que también añade un nuevo y completo conjunto de funciones de biblioteca que muy probablemente se incluirán con el tiempo en la *libc* estándar. Esto incluye un nuevo sistema de configuración para resolver nombres de nodos que reemplace el sistema actual que usa el fichero *host.conf*. Las características de estas funciones serán discutidas más adelante.

Este capítulo se centrará en NYS más que en los otros dos paquetes, a los que nos referiremos como el código NIS "tradicional". Si usted desea utilizar alguno de esos paquetes, las instrucciones de este capítulo podrían ser suficientes o tal vez no. Para obtener información adicional, por favor, consiga un libro estándar sobre NIS como el *NFS y NIS* de Hal Stern (véase [Stern92]).

Por el momento NYS está todavía en desarrollo y por lo tanto las utilidades estándar de Linux como los programas de red o el programa de *login* todavía no tienen en cuenta el sistema de configuración de NYS. Por lo tanto, hasta que NYS no sea incluido en la *libc* principal tendrá que compilar todos esos programas usted mismo si quiere conseguir que usen NYS. Para ello, en los *Makefile*s de cualquiera de esas aplicaciones deberá especificar *-lnsl* como la última opción antes de *libc* al enlazador. Esto enlazará las funciones relevantes de *libnsl*, la biblioteca NYS, en lugar de la biblioteca C estándar.

10.1 Familiarización con NIS

NIS mantiene información de la base de datos en los llamados *mapas* que contienen pares clave-valor. Los mapas se almacenan en un nodo central que está ejecutando el servidor NIS y del que los clientes pueden obtener la información a través de varias llamadas RPC.

[2]Que puede ser localizado en **swen@uni-paderborn.de**. Los clientes NIS están disponibles como `yp-linux.tar.gz` en **metalab.unc.edu** en el directorio *system/Network*.

[3]La última versión (en la fecha en que se escribió este documento) es `yps-0.21` y puede obtenerse en **ftp.lysator.liu.se**, en el directorio */pub/NYS*.

[4]Localizable en **pen@lysator.liu.se**.

Muy frecuentemente, los mapas se almacenan en ficheros DBM.[5]

Los mapas en sí mismos suelen ser generados a partir de ficheros de texto maestros como */etc/hosts* o */etc/passwd*. Para algunos ficheros se crean varios mapas, uno por cada tipo de clave de búsqueda. Por ejemplo, usted podría buscar en el fichero *hosts* tanto por un nombre de nodo como por su dirección IP. Así pues, de él se derivan dos mapas NIS, llamados *hosts.byname* y *hosts.byaddr* respectivamente. La tabla 10.1 lista los mapas típicos y los ficheros de los que son generados.

Fichero Maestro	Mapa(s)	
/etc/hosts	*hosts.byname*	*hosts.byaddr*
/etc/networks	*networks.byname*	*networks.byaddr*
/etc/passwd	*passwd.byname*	*passwd.byuid*
/etc/group	*group.byname*	*group.bygid*
/etc/services	*services.byname*	*services.bynumber*
/etc/rpc	*rpc.byname*	*rpc.bynumber*
/etc/protocols	*protocols.byname*	*protocols.bynumber*
/usr/lib/aliases	*mail.aliases*	

Tabla 10.1: Algunos mapas NIS estándar y los ficheros correspodientes.

Hay otros ficheros y mapas para los que puede encontrar soporte en uno u otro paquete NIS. Éstos pueden contener información sobre aplicaciones no tratadas en este libro, como el mapa *bootparams*[6] que puede ser usado por algunos servidores BOOTP, o mapas que actualmente no tienen ninguna función en Linux (como los mapas *ethers.byname* y *ethers.byaddr*).

La gente usa habitulamente apodos para algunos mapas, ya que son más cortos y por lo tanto más faciles de escribir. Para obtener una lista completa de los apodos[7] reconocidos por sus herramientas NIS, ejecute el siguiente comando:

```
$ ypcat -x
NIS map nickname translation table:
        "passwd" -> "passwd.byname"
        "group" -> "group.byname"
        "networks" -> "networks.byaddr"
        "hosts" -> "hosts.byname"
        "protocols" -> "protocols.bynumber"
```

[5]DBM es una biblioteca de manejo de bases de datos sencillas que usa técnicas hashing para acelerar operaciones de búsqueda. Existe una implementación gratuita de DBM perteneciente al proyecto GNU llamada *gdbm*, que forma parte de la mayoría de las distribuciones de Linux.

[6]N. del T.: relacionado con los parámetros de arranque

[7]N. del T.: del inglés *nicknames*

```
"services" -> "services.byname"
"aliases" -> "mail.aliases"
"ethers" -> "ethers.byname"
"rpc" -> "rpc.bynumber"
"netmasks" -> "netmasks.byaddr"
"publickey" -> "publickey.byname"
"netid" -> "netid.byname"
"passwd.adjunct" -> "passwd.adjunct.byname"
"group.adjunct" -> "group.adjunct.byname"
"timezone" -> "timezone.byname"
```

El servidor NIS suele llamarse *ypserv*. Para una red de tipo medio un único servidor suele ser suficiente; en redes mayores pueden elegir ejecutar varios en máquinas diferentes y en diferentes segmentos para aliviar la carga en los servidores y en los encaminadores. Estos servidores están sincronizados haciendo que uno de ellos sea el *servidor maestro* y que los demás sean *servidores esclavos*. Los mapas se crearán sólo en la máquina del servidor maestro. A partir de ahí son distribuidos a todos los esclavos.

Habrá notado usted que hemos estado hablando de "redes" todo el rato muy vagamente; por supuesto existe un concepto diferenciado en NIS sobre lo que es un *dominio* en la red: es el conjunto de todos los nodos que comparten parte de sus datos de configuración del sistema mediante NIS. Desafortunadamente los dominios NIS no tienen absolutamente nada que ver con los dominios que podemos encontrar en DNS. Por ello, para evitar cualquier tipo de ambigüedad a lo largo de este capítulo, especificaremos en todo momento el tipo de dominio al que nos estemos refiriendo.

Los dominios NIS tienen sólo una función puramente administrativa. Son además invisibles para los usuarios. Por ello el nombre dado a un dominio NIS es sólo relevante para administradores. Por lo general cualquier nombre valdrá con tal de que sea distinto de cualquier otro nombre de dominio NIS de su red local. Por ejemplo, el administrador de la Cervecera Virtual puede decidir crear dos dominios NIS, uno para la Cervecera en sí, y otro para la Vinatera, a los que llama **cervecera** y **vinatera** respectivamente. Otra idea bastante utilizada es usar simplemente el nombre de dominio DNS también para el NIS. Para establecer y ver el nombre de dominio NIS de su nodo puede usar el comando *domainname*. Cuando se ejecuta sin ningún argumento, muestra el nombre de dominio NIS actual; para establecer el nombre de dominio, debe usted entrar como superusuario y escribir:

```
# domainname cervecera
```

Los dominios NIS determinan a qué servidor NIS preguntarán las aplicaciones. Por ejemplo, el programa *login* de un nodo de la Vinatera debería, por supuesto, pedir información de la contraseña de un usuario sólo al servidor NIS de la Vinatera (o a uno de ellos si es que hay varios), mientras que una aplicación de un nodo de la Cervecera debería arreglárselas con el servidor de la Cervecera.

Queda un misterio por resolver: cómo sabe un cliente a qué servidor conectarse. La solución más simple sería tener un fichero de configuración que diga el nodo en el que encontrar el servidor. Sin embargo, esta solución es bastante inflexible porque no permite a los clientes usar servidores diferentes (del mismo dominio, se entiende), dependiendo de su disponiblidad. Por ello las implementaciones tradicionales de NIS se apoyan en un demonio especial denominado *ypbind* para detectar un servidor NIS adecuado dentro de su dominio NIS. Cualquier aplicación, antes de poder realizar cuaquier consulta NIS, debe averiguar primero, a través de *ypbind*, qué servidor usar.

ypbind busca los servidores mandando un mensaje de difusión por toda la red IP local. El primero en responder se supone que será el más rápido potencialmente y será el que se use en todas las consultas NIS subsiguientes. Después de un cierto intervalo de tiempo, o si el servidor se vuelve inaccesible, *ypbind* volverá a buscar los servidores activos.

Ahora bien, hay dos aspectos discutibles sobre el enlazado dinámico: uno es que raramente es necesario y el otro es que introduce un problema de seguridad: *ypbind* cree a ciegas a cualquiera que conteste, que podría ser lo mismo un humilde servidor NIS que un intruso malicioso. No hace falta decir que esto es especialmente problemático si usted maneja sus bases de datos de contraseñas a través de NIS. Para protegerse contra esto, NYS *no* usa *ypbind* por defecto, sino que obtiene el nombre de nodo del servidor de un fichero de configuración.

10.2 NIS frente a NIS+

NIS y NIS+ comparten poco más que su nombre y un objetivo común. NIS+ está estructurado de una forma completamente diferente. En lugar de un simple espacio de nombres con dominios NIS inconexos, usa un espacio de nombres jerárquico similar al de DNS. En lugar de mapas, usa *tablas* que están compuestas por filas y columnas, donde cada fila representa un objeto en la base de datos NIS+, mientras que las columnas cubren aquellas propiedades de los objetos que NIS+ conoce y que le interesan. Cada tabla de un dominio NIS+ dado incluye las de sus dominios padre. Además, una entrada en una tabla puede contener un enlace a otra tabla. Todas estas características hacen posible la estructuración de la información de muchas formas.

El NIS tradicional tiene un número de versión RPC de 2, mientras que NIS+ tiene un número de versión RPC de 3.

NIS+ no parece ser ampliamente usado todavía [8] , y no conozco tanto sobre él realmente (bueno, prácticamente nada). Por esta razón, no lo trataremos aquí. Si está interesado en aprender más sobre él, por favor, refiérase al manual de administración de NIS+ de Sun ([NISPlus]).

10.3 El lado cliente de NIS

Si está usted familiarizado con la escritura o el portado de aplicaciones de red, notará que la mayoría de los mapas NIS listados arriba corresponden a funciones de la biblioteca C. Por ejemplo, para obtener información del fichero *passwd*, se suelen usar normalmente las funciones *getpwnam(3)* y *getpwuid(3)* que devuelven información de la cuenta asociada al nombre de usuario dado, o al identificador de usuario dado, respectivamente. En circunstancias normales, estas funciones realizarán la búsqueda requerida en el fichero estándar */etc/passwd*.

Una implementación de estas funciones que tenga en cuenta NIS, sin embargo, modificará este comportamiento, y realizará una llamada RPC para que sea el servidor NIS el que realice la búsqueda del nombre o identificador de usuario. Esto ocurre de forma completamente transparente a la aplicación. La función podría “añadir” el mapa NIS o “sustituir” el fichero original con él. Por supuesto, esto no implica una modificación real del fichero, sólo significa que a la aplicación le *parece* que el fichero ha sido sustituido o que le han añadido algo.

En las implementaciones tradicionales de NIS solía haber ciertas convenciones sobre qué mapas se sustituían y cuáles eran añadidos a la información original. Alguno, como el mapa *passwd*, requería modificaciones extrañas del fichero *passwd* que, si se hacían mal, podrían abrir agujeros de seguridad. Para evitar estos obstáculos, NYS usa un modo general de configuración que determina si un conjunto de funciones cliente en particular usa los ficheros originales, NIS o NIS+, y en qué orden. Esto será descrito en otra sección más adelante en este mismo capítulo.

10.4 Ejecución de un servidor NIS

Después de todo este parloteo técnico teórico, ya empieza a ser hora de que metamos mano al verdadero trabajo de configuración. En esta sección cubriremos la configuración de un servidor NIS. Si ya hay un servidor NIS corriendo en su red, no necesita configurar su propio servidor; en este caso puede usted saltarse esta sección.

[8] N. del T.: En el momento de escribirse este manual esta afirmación era cierta. Sin embargo, hoy en día NIS+ es ampliamente usado por sus ventajas frente a NIS.

◇ Observe que si únicamente va usted a experimentar con el servidor, tiene que asegurarse de que no lo configura con un nombre de dominio NIS que ya esté en uso en su red. Ello podría desbaratar todo el servicio de red y hacer a mucha gente desdichada, y muy enfadada.

Actualmente hay disponibles dos servidores NIS de forma gratuita para Linux, uno contenido en el paquete *yps* de Tobias Reber, y el otro en el paquete *ypserv* de Peter Eriksson. No debería importar cuál utilice usted, independientemente de que usted use NYS o el código de cliente NIS estándar que existe actualmente en *libc*. En el momento de escribir esto, el código para manejar servidores NIS esclavos parece ser más completo en *yps*. Así que si tiene que tratar con servidores esclavo, *yps* puede ser una opción mejor.

Tras instalar el programa servidor (*ypserv*) en */usr/sbin*, deberá crear el directorio que va a contener los ficheros mapa que su servidor va a distribuir. Al establecer un dominio NIS para el dominio **cervecera**, los mapas irían al fichero */var/yp/cervecera*. El servidor determina si está sirviendo un dominio NIS en particular comprobando si el directorio mapa está presente. Si va a deshabilitar el servicio para algún dominio NIS, asegúrese de eliminar el directorio también.

Los mapas normalmente se almacenan en ficheros DBM para acelerar las búsquedas. Se crean a partir de los ficheros maestro usando un programa llamado *makedbm* (para el servidor de Tobias) o *dbmload* (para el servidor de Peter). Estos pueden no ser intercambiables. Transformar un fichero maestro a una forma entendible por *dbmload* normalmente requiere un poco de magia *awk* o *sed*, lo que tiende a ser un poco tedioso de escribir y difícil de recordar. Por ello, el paquete *ypserv* de Peter Eriksson contiene un Makefile (llamado *ypMakefile*) que realiza todos esos trabajos por usted. Debería instalarlo como *Makefile* en su directorio de mapas, y editarlo para que refleje los mapas que desee distribuir. Hacia el principio del fichero encontrará la etiqueta *all* que lista los servicios que *ypserv* ofrece. Por defecto, la línea es algo parecido a esto:

```
all: ethers hosts networks protocols rpc services passwd group netid
```

Si no desea producir los mapas *ethers.byname* y *ethers.byaddr*, por ejemplo, simplemente elimine la palabra *ethers* de la línea. Para probar su configuración, puede ser suficiente con empezar con sólo uno o dos mapas, como los mapas *services.**.

Tras editar el *Makefile*, y sin salir del directorio de mapas, teclee "`make`". Esto generará e instalará automáticamente los mapas. Debe asegurarse de actualizar los mapas cada vez que cambie los ficheros maestros, de otro modo los cambios seguirán siendo invisibles para la red.

La siguiente sección explica cómo configurar el código de cliente NIS. Si su configuración no funciona, debería comprobar si llega alguna petición a su servidor o no. Si especifica el parámetro `-D` al servidor NYS, éste imprimirá mensajes de depuración en la consola sobre todas las peticiones NIS entrantes, y los resultados devueltos. Esto debería darle una idea sobre dónde puede estar el problema. El servidor de Tobias no tiene esa opción.

10.5 Configurar un Cliente NIS con NYS

A lo largo de lo que queda de este capítulo, cubriremos la configuración de un cliente NIS.

Su primer paso debería ser indicarle a NYS qué servidor usar para el servicio NIS, estableciéndolo en el fichero de configuración */etc/yp.conf*. Un fichero de ejemplo muy sencillo para un nodo en la red de la Vinatera sería algo así:

```
# yp.conf - Configuracion YP para la biblioteca NYS.
#
domainname vinatera
ypserver vbardolino
```

La primera sentencia indica a los clientes NIS que pertenecen al dominio NIS **vinatera**. Si omite esta línea, NYS usará el nombre de dominio que usted asignó a su sistema con el comando *domainname*. La sentencia *ypserver* indica el servidor a usar. Por supuesto, la dirección IP correspondiente a **vbardolino** debe estar establecida en el fichero *hosts*; alternativamente, podría usar directamente la dirección IP en la sentencia *ypserver*.

En el fichero mostrado arriba, el comando *ypserver* indica a NYS que use el servidor indicado sea cual sea el dominio NIS actual. Sin embargo, si mueve frecuentemente su máquina entre diferentes dominios NIS, tal vez le interesaría mantener la información de varios dominios en el fichero *yp.conf*. Puede tener información sobre los servidores para varios dominios NIS en *yp.conf* añadiendo el nombre de dominio NIS a la sentencia *ypserver*. Por ejemplo, podría cambiar el fichero del ejemplo anterior para que sea algo así:

```
# yp.conf - Configuracion YP para la biblioteca NYS.
#
ypserver vbardolino vinatera
ypserver vstout     cervecera
```

Esto le permite mover su máquina a cualquiera de los dos dominios simplemente con establecer el dominio NIS deseado durante el arranque con el comando *domainname*.

Una vez creado este fichero de configuración básico y de asegurarse de que tiene permiso de lectura para todo el mundo, debería realizar su primera prueba para comprobar si puede conectar con su servidor. Asegúrese de elegir cualquier mapa que su servidor distribuya, como *hosts.byname*, e intente obtenerlo usando la utilidad *ypcat*. *ypcat*, como todas las demás herramientas NIS, debe encontrarse en */usr/sbin*.

```
# ypcat hosts.byname
191.72.2.2      vbeaujolais  vbeaujolais.linus.lxnet.org
191.72.2.3      vbardolino   vbardolino.linus.lxnet.org
191.72.1.1      vlager       vlager.linus.lxnet.org
191.72.2.1      vlager       vlager.linus.lxnet.org
191.72.1.2      vstout       vstout.linus.lxnet.org
191.72.1.3      vale         vale.linus.lxnet.org
191.72.2.4      vchianti     vchianti.linus.lxnet.org
```

La salida que obtenga debe ser algo parecido a lo expuesto arriba. Si recibe un mensaje de error en su lugar que diga "`Can't bind to server which serves domain`" (*no se puede conectar al servidor del dominio*), o algo similar, entonces, o el nombre de dominio NIS que ha establecido no tiene definido su servidor correspondiente en *yp.conf*, o el servidor, por alguna razón, no está disponible. En este último caso, asegúrese de que un *ping* a esa máquina da un resultado positivo, y de que está en efecto ejecutando un servidor NIS. Puede verificar esto último usando *rpcinfo*, que debería producir la siguiente salida:

```
# rpcinfo -u servidor ypserv
program 100004 version 2 ready and waiting
```

10.6 Elección de los Mapas Correctos

Una vez que esté seguro de que puede llegar al servidor NIS, debe decidir qué ficheros de configuración sustituir o aumentar con mapas NIS. Normalmente deseará usar mapas NIS para las funciones de busqueda de nodos y de claves de usuario. El primero es especialmente útil si no utiliza DNS. El segundo permite a todos los usuarios entrar en su cuenta desde cualquier sistema dentro del dominio NIS; esto suele requerir compartir un directorio */home* central entre todos los nodos mediante NFS. Todo esto se explica en detalle en la sección 10.7 más abajo. Otros mapas, como *services.byname*, no proporcionan una ganancia tan clara, pero ahorran algo de trabajo de edición si instala alguna aplicación de red que use un nombre de servicio que no esté en el fichero *services* estándar.

Por lo general, usted deseará tener alguna libertad de elección acerca de cuándo una función de búsqueda usará ficheros locales y cuándo hará una petición al servidor NIS.

NYS le permite configurar el orden en que una función accede a estos servicios. Esto se controla mediante el fichero */etc/nsswitch.conf*, que quiere decir *Selector del Servicio de Nombrado*[9] pero por supuesto no está limitado a los servicios de nombres. Para cualquiera de las funciones de búsqueda de datos soportadas por NYS, contiene una línea citando los servicios a usar.

El orden correcto de los servicios depende del tipo de datos. Es improbable que el mapa *services.byname* contenga entradas diferentes que las que se encuentran en el fichero *services* local; únicamente podría contener más. Así que una buena elección sería consultar los ficheros locales primero, y probar con NIS sólo si el nombre del servicio no fue encontrado. Por otro lado, la información de nombres de nodos puede cambiar muy frecuentemente, de forma que el DNS o el servidor NIS tendrían siempre la información más precisa, mientras que el fichero *hosts* local sólo se mantiene como copia de respaldo por si el DNS y NIS fallasen. En este caso, habría que comprobar el fichero local en último lugar.

El siguiente ejemplo muestra cómo configurar las funciones *gethostbyname(2)*, *gethostbyaddr(2)*, y *getservbyname(2)* de la forma descrita anteriormente. Probarán los servicios listados por orden; si una busqueda es satisfactoria, se devuelve el resultado, si no, se intentará con el siguiente servicio.

```
# /etc/nsswitch.conf de ejemplo
#
hosts:     nis dns files
services:  files nis
```

Más abajo se muestra la lista completa de servicios que pueden ser usados en una entrada del fichero *nsswitch.conf*. Los mapas, ficheros, servidores y objetos que se pueden consultar dependen del nombre de la entrada.

nisplus o *nis+*
: Usa el servidor NIS+ para este dominio. La situación del servidor se obtiene del fichero */etc/nis.conf*.

nis
: Usa el servidor NIS actual de este dominio. La situación del servidor consultado está configurada en el fichero *yp.conf* como se mostró en la sección previa. Para la entrada *hosts* se consultan los mapas *hosts.byname* y *hosts.byaddr*.

dns
: Usa el servidor de nombres DNS. Este tipo de servicio sólo es útil con la entrada *hosts*. Los servidores de nombres consultados siguen estando determinados por el fichero estándar *resolv.conf*.

[9]N. del T.: del inglés *Name Service Switch*.

files Usa el fichero local. Como el fichero */etc/hosts* para la entrada *hosts*.

dbm Busca la información en ficheros DBM localizados en */var/dbm*. El nombre usado para el fichero es el del mapa NIS correspondiente.

Actualmente NYS soporta las siguientes entradas en *nsswitch.conf*: *hosts*, *networks*, *passwd*, *group*, *shadow*, *gshadow*, *services*, *protocols*, *rpc* y *ethers*. Es probable que se añadan más entradas.

La figura 10.1 muestra un ejemplo más completo que introduce otra característica del fichero *nsswitch.conf*: la palabra clave *[NOTFOUND=return]* en la entrada *hosts* indica a NYS que retorne si el elemento deseado no pudo ser encontrado en la base de datos NIS o DNS. Esto es, NYS continuará y buscará en los ficheros locales *sólo* si las llamadas a los servidores NIS y DNS fallaron por alguna otra razón. Los ficheros locales serán usados entonces sólo durante el arranque y como copia de respaldo cuando el servidor NIS se haya caído.

```
# /etc/nsswitch.conf
#
hosts:      nis dns [NOTFOUND=return] files
networks:   nis [NOTFOUND=return] files

services:   files nis
protocols:  files nis
rpc:        files nis
```

Figura 10.1: Fichero *nsswitch.conf* de ejemplo.

10.7 Uso de los mapas *passwd* y *group*

Uno de las usos más importantes de NIS es sincronizar usuarios e información de cuentas en todos los nodos de un dominio NIS. Para este fin, sólo se suele mantener un pequeño fichero local */etc/passwd*, al que se le añade información de todas las demás cuentas mediante los mapas NIS. Sin embargo, sólo con habilitar búsquedas NIS para este servicio en *nsswitch.conf* no es suficiente.

Cuando se base en la información de contraseñas distribuida por NIS, debe primero cerciorarse de que los números identificadores de cualquier usuario que tenga en su fichero *passwd* local coincidan con la idea que tiene el servidor NIS de identificadores de usuarios. Usted también deseará esto para otros propósitos, como montar volúmenes NFS de otros nodos de su red.

Si alguno de los números de usuario de */etc/passwd* o de */etc/group* son distintos de los que aparecen en los mapas, tiene que ajustar el propietario de todos los ficheros que pertenezcan a ese usuario. Primero debería cambiar todos los números de uid y gid en *passwd* y *group* a los nuevos valores; después, encontrar todos los ficheros que pertenezcan a los usuarios recién modificados, y finalmente cambiarles el propietario. Supongamos que **news** tenía un número de usuario de 9, y que **okir** tenía un número de usuario de 103, y que éstos fueron cambiados a algún otro valor; entonces debería teclear los siguientes comandos:

```
# 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 estos comandos con el *nuevo* fichero *passwd* ya instalado, y que recoja todos los nombres de ficheros antes de cambiar el propietario de cualquiera de ellos. Para cambiar el grupo propietario de los ficheros, se usará un comando similar.

Una vez hecho esto, los números de uid y gid de su sistema estarán de acuerdo con los de los demás nodos de su dominio NIS. El siguiente paso será añadir las líneas de configuración a *nsswitch.conf* que habiliten las búsquedas NIS de información de usuario y grupo.

```
# /etc/nsswitch.conf - tratamiento de contrase\~na y grupo
passwd: nis files
group:  nis files
```

Esto hace que el comando *login* y otros de esta familia consulten primero los mapas NIS cuando un usuario intenta entrar, y si esta búsqueda falla, sigan con los ficheros locales. Normalmente usted borrará la mayor parte de los usuarios de sus ficheros locales y sólo dejará en ellos entradas para **root** y para cuentas genéricas como **mail**. Esto es porque algunas tareas vitales del sistema pueden requerir mapear uids a nombres de usuario o viceversa. Por ejemplo, las tareas *cron* administrativas pueden ejecutar el comando *su* para convertirse temporalmente en **news**, o el subsistema UUCP puede enviar un informe de estado en un mensaje. Si **news** y **uucp** no tienen entradas en el fichero *passwd* local, estas tareas fallarán miserablemente durante un apagón de NIS.

Dos observaciones importantes: por un lado, la configuración descrita hasta aquí sólo funciona para sistemas de login que no usan contraseña *shadow*, como los incluidos en el paquete *util-linux*. Los intrincados métodos para usar contraseñas *shadow* con NIS serán cubiertos más adelante. Por otro lado, los comandos login no son los únicos que acceden al fichero *passwd* – por ejemplo la orden *ls* que la mayor parte de la gente usa constantemente. Cada vez que se saca un listado con la opción -l, *ls* mostrará los nombres simbólicos del propietario y del grupo; esto es, por cada uid y gid que encuentre, deberá hacer una petición al servidor NIS. Esto ralentizará mucho todo el sistema cuando su red local se atasque, o, aún peor, cuando el servidor NIS no esté en la misma red física, de forma que los datagramas tengan que pasar a través de un encaminador o puente.

Y esto no es todo. Imagine lo que pasa si un usuario quiere cambiar su contraseña. Normalmente invocará el comando *passwd*, que leerá la nueva contraseña y actualizará el fichero *passwd* local. Esto es imposible con NIS, puesto que ese fichero no está disponible localmente, y tampoco es una opción que los usuarios entren en el servidor NIS cada vez que quieran cambiar la clave. Por ello, NIS proporciona un sustituto para *passwd* llamado *yppasswd*, que realiza una tarea análoga en NIS. Para cambiar la contraseña en la máquina servidora, contacta con el demonio *yppasswdd* en ese nodo mediante RPC, y le proporciona información de la contraseña actualizada. Generalmente, *yppasswd* se instala sobre el programa normal haciendo algo así:

```
# cd /bin
# mv passwd passwd.old
# ln yppasswd passwd
```

Al mismo tiempo tendrá que instalar *rpc.yppasswdd* en el servidor y arrancarlo desde los guiones *rc*. Esto ocultará de forma efectiva todas las complicaciones de NIS a sus usuarios.

10.8 Uso de NIS con Soporte *Shadow*

Todavía no existe soporte NIS para instalaciones que usan el conjunto de utilidades *shadow password*. John F. Haugh, el autor del conjunto *shadow*, publicó recientemente una versión de las funciones de biblioteca *shadow* en **comp.sources.misc**, cubiertas por la licencia LGPL de GNU. Ya tiene algún soporte para NIS, pero no está completa, y los ficheros no han sido añadidos todavía a la biblioteca C estándar. Por otro lado, publicar la información de */etc/shadow* via NIS rompe de alguna manera con el propósito del conjunto *shadow*.

Aunque las funciones de búsqueda de contraseña en NYS no usan un mapa *shadow.byname* ni nada parecido, NYS soporta el uso de un fichero */etc/shadow* local de forma transparente. Cuando la implementación NYS de *getpwnam* es llamada para buscar información relacionada con un login dado, se consultan las facilidades especificadas por la entrada *passwd* en *nsswitch.conf*. El servicio *nis* seguirá mirando en el mapa *passwd.byname* del servidor NIS. En cambio, el servicio *files* mirará si existe */etc/shadow* y lo intentará abrir. Si no existe, o si el usuario no tiene privilegios de **root**, volverá al comportamiento tradicional de mirar la información del usuario sólo en */etc/passwd*. Sin embargo, si el fichero *shadow* existe y puede ser abierto, NYS extraerá la contraseña del usuario de *shadow*. La función *getpwuid* se implementa similarmente. De esta forma, los binarios compilados con NYS funcionarán de forma transparente con una instalación *shadow* local.

10.9 Uso del Código NIS Tradicional

Si está usando el código de cliente existente en la *libc* estándar, la configuración de un cliente NIS es un poco diferente. Por un lado, se usa un demonio *ypbind* para buscar por la red servidores activos en vez de obtener esta información de un fichero de configuración. Usted tendrá por ello que cerciorarse de que arranca *ypbind* durante la inicialización del sistema. Debe ser invocado después de que el dominio NIS haya sido establecido y de que el mapeador de puertos RPC haya sido arrancado. Después podrá invocar *ypcat* para comprobar el servidor como se mostró más arriba.

Reciéntemente ha habido numerosos informes de error indicando que NIS falla con un mensaje de error que dice: "`clntudp_create: RPC: portmapper failure - RPC: unable to receive`". Estos mensajes de error son debidos a un cambio incompatible en el modo en que *ypbind* comunica la información de enlazado a las funciones de biblioteca. Con obtener las fuentes más recientes de las utilidades NIS y recompilarlas se debería solucionar este problema.[10]

Del mismo modo, la forma en que el NIS tradicional decide si hay que mezclar información NIS, y cómo, con la de los ficheros locales es distinta que la usada por NYS. Por ejemplo, para usar los mapas de contraseña NIS, debe incluir la siguiente línea en algún lugar de su mapa */etc/passwd*:

```
+:*:0:0:::
```

Esto marca el lugar donde las funciones de búsqueda de contraseñas "insertan" los mapas NIS. Insertando una línea similar (menos los dos últimos dos puntos) en */etc/group* obtenemos lo mismo para los mapas *group.**. Para usar los mapas *hosts.** distribuidos por NIS, cambie la línea *order* del fichero *host.conf*. Por ejemplo, si desea usar NIS, DNS, y el fichero */etc/hosts* (por ese orden), necesita cambiar esa línea por:

```
order yp bind hosts
```

La implementación NIS tradicional no soporta ningún otro mapa más por el momento.

[10] El código fuente de *yp-linux* puede obtenerse de **ftp.uni-paderborn.de**, en el directorio */pub/Linux/LOCAL*.

Capítulo 11

El Sistema Ficheros en Red (NFS)

NFS, acrónimo de *Network File System*, que nosotros llamaremos *Sistema de Ficheros en Red*, es probablemente el servicio más complejo de los que se ofrecen usando RPC. Permite acceder a los ficheros remotos exactamente igual que si fueran locales. Esto se hace programando parte de la funcionalidad a nivel del núcleo (en el lado del cliente) y la otra parte como un demonio servidor. El acceso a los ficheros es totalmente transparente al cliente, funcionando con muchas arquitecturas de servidores.

NFS ofrece numerosas ventajas:

- Los datos accedidos por todo tipo de usuarios pueden mantenerse en un nodo central, con clientes que montan los directorios en el momento de arrancar. Por ejemplo, puede mantener todas las cuentas de usuario en una máquina, y hacer que las demás monten dichas cuentas en su directorio */home* por NFS. Si además se instala NIS, los usuarios podrían entrar y trabajar de forma transparente en cualquiera de las máquinas.
- Los datos que consumen grandes cantidades de espacio de disco pueden mantenerse en un nodo. Por ejemplo, mantener una sola copia de LaTeX en lugar de copiarlo en cada nodo.
- Los datos de administración pueden también mantenerse en un solo nodo. Ya no será necesario usar *rcp* para instalar el mismo fichero en 20 máquinas distintas.

El NFS de Linux es, principalmente, obra de Rick Sladkey,[1], pues escribió el código que corresponde al núcleo y buena parte del código del servidor NFS. Este último es una modificación del servidor *unfsd* que corre en espacio de usuario, escrito originalmente por Mark Shand, y el servidor *hnfs* (Harris NFS) escrito por Donald Becker.

[1]Puede contactar con Rick en la dirección **jrs@world.std.com**

Veamos ahora un poco cómo funciona NFS: un cliente puede solicitar montar un directorio desde un servidor remoto, de forma similar a como montaría un directorio local. Sin embargo, la sintaxis no es exactamente igual. Por ejemplo, para montar el directorio */home* del nodo **vlager** en el directorio */users* de **vale**, el administrador escribiría el siguiente comando en **vale**:[2]

```
# mount -t nfs vlager:/home /users
```

mount intentará conectar con el demonio remoto *mountd* mediante RPC. El servidor comprobará si la máquina **vale** tiene permiso para montar el directorio pedido, y si es así retornará un descriptor de fichero. Este descriptor se utilizará en todas las peticiones que sobre ficheros de */users* se realicen posteriormente.

Cuando alguien accede a un fichero remoto, el núcleo manda una llamada RPC al programa *nfsd* (demonio de NFS) del nodo remoto. Esta llamada incluye el descriptor de fichero, el nombre del fichero a acceder y los identificadores de usuario y de grupo del demandante. Estos identificadores se usan para chequear permisos de acceso en la máquina remota, con lo que los usuarios de ambas máquinas deberían ser los mismos.

En varias implementaciones de UN*X, las funcionalidades de cliente y servidor NFS se realizan como demonios de nivel de núcleo que se arrancan desde el espacio de usuario al arrancar la máquina. Se trata del programa *nfsd* en el servidor y del programa *biod* (*Block I/O Daemon*, o demonio de E/S[3] por bloques) en el cliente. Para aumentar el rendimiento, *biod* realiza E/S asíncrona, y a veces corren concurrentemente varios servidores de NFS.

La implementación de NFS en Linux es algo diferente: el código de cliente está integrado en la capa de sistema de ficheros virtual (VFS) y no requiere control adicional mediante el programa *biod*. Por otro lado, el código de servidor corre totalmente en el espacio de usuario, por lo que ejecutar varias copias del *nfsd* resulta imposible debido a los problemas de sincronización que originaría.

El mayor problema con el código NFS de Linux es que el núcleo 1.0 no puede manejar bloques de memoria de más de 4Kb, por lo que el código de red no puede manejar datagramas de un tamaño mayor que 3500 octetos una vez eliminadas las cabeceras. Esto significa que las transferencias con servidores NFS que utilicen datagramas grandes por defecto (por ejemplo, los 8Kb de SunOS) necesitan ser reducidos artificialmente. Esto produce pérdidas de rendimiento en ciertas circunstancias.[4] Esta limitación desapareció en los núcleos posteriores al 1.1, reescribiéndose el código del cliente para aprovechar la nueva situación.

[2] Observar que puede omitirse la opción `-t nfs`, ya que el programa *mount* sabe por la aparición de los dos puntos (`:`) que se trata de un sistema NFS.

[3] N. del T.: E/S es Entrada/Salida

[4] Como me explicó Alan Cox: La especificación de NFS requiere que el servidor guarde cada escritura en disco antes de retornar un reconocimiento al cliente (ACK). Como los núcleos de BSD sólo manejan

11.1 Preparación de NFS

Antes de usar NFS, sea en cliente o servidor, debe asegurarse de que el núcleo tiene el soporte incluido. Los núcleos modernos informan de ello a través del sistema */proc*, con un comando como el siguiente:

```
$ cat /proc/filesystems
minix
ext2
msdos
nodev proc
nodev nfs
```

Si no aparece la palabra *nfs*, tendrá que recompilar el núcleo con el soporte NFS habilitado. Sobre cómo configurar el núcleo hablamos en la sección "Configuración del Núcleo" del capítulo 3.

Con versiones del núcleo anteriores a la 1.1, la forma de comprobarlo es intentar montar un sistema NFS de prueba, de la siguiente forma:

```
# mkdir /tmp/test
# mount localhost:/etc /tmp/test
```

Si el comando *mount* falla con el mensaje "`fs type nfs no supported by kernel`" (*sistema tipo NFS no soportado por el núcleo*), deberá recompilar el núcleo habilitando NFS. Otro tipo de errores no implican recompilar el núcleo, ya que se producen al no estar corriendo el programa *nfsd*.

11.2 Montaje de un volumen NFS

Los volúmenes NFS[5] se montan como los sistemas de ficheros usuales. Se trata de llamar al comando *mount* con la sintaxis:

```
# mount -t nfs volumen_nfs directorio_local opciones
```

escrituras del tamaño de página (4 Kb), escribir en 4 veces 1Kb cada una en un servidor NFS basado en BSD tiene como consecuencia hacer cuatro operaciones de escritura de 4 Kb cada una.

[5]Hablamos de volúmenes, y no de sistemas de ficheros, porque no lo son realmente: pueden ser sólo directorios de un sistema.

La parte `volumen_nfs` se especifica con la sintaxis "`nodo_remoto:directorio_remoto`". Dado que esta notación es propia del NFS, la opción `-t nfs` resulta redundante.

Hay otras opciones que pueden incluirse en el programa *mount*, que van tras el modificador `-o` en la línea de comando o en el campo de opciones de la entrada correspondiente en el fichero */etc/fstab*. En ambos casos, las distintas opciones deben separarse por comas. Las opciones que se especifiquen en la línea de comandos tendrán preferencia sobre otras que se indiquen en */etc/fstab*.

Una entrada de ejemplo del fichero */etc/fstab* podría ser

```
# volumen              directorio        tipo  opciones
news:/usr/spool/news   /usr/spool/news   nfs   timeo=14,intr
```

Ahora el volumen anterior puede montarse con la orden

```
# mount news:/usr/spool/news
```

Ante la ausencia de una entrada en *fstab*, las llamadas al programa *mount* se hacen más incómodas. Por ejemplo, puede que tenga que teclear cosas como ésta, para especificar que se limite el tamaño del datagrama a 2 Kb:

```
# mount moonshot:/home /home -o rsize=2048,wsize=2048
```

La lista de todas las opciones válidas para *mount* se encuentra descrita en la página de ayuda *nfs(5)* que viene con la utilidad de montaje de Rick Sladkey, que forma parte del paquete *util-linux* de Rik Faith. Las opciones más interesantes son las siguientes:

rsize=`n` y *wsize=*`n`
: Especifican el tamaño de datagrama utilizado por el cliente NFS en las peticiones de lectura y escritura, respectivamente. Por defecto, cada una de ellas vale 1024 octetos, dados los límites del tamaño de datagrama UDP ya comentados.

timeo=`n`
: Esta opción establece el tiempo máximo de espera de respuesta a una petición del cliente NFS; en centésimas de segundo. Por defecto, este valor es de 0.7 segundos.

hard
: Marca el montaje del volume como físico. Es un valor por defecto.

soft
: Hace que el montaje sea sólo lógico (opuesto al anterior).

intr Esta opción habilita la posibilidad de que una señal interrumpa una espera por NFS. Es útil para poder abortarla cuando el servidor no responde.

Cuando el cliente realiza una petición al servidor NFS, esperará un tiempo máximo (el que se especifica en la opción *timeout*). Si no hay confirmación tras ese tiempo (tiempo que se denomina "de expiración" o *timeout*) tiene lugar otra espera, "de expiración secundaria" o *minor timeout*, en el que la operación se reintenta pero doblando el tiempo de expiración inicial. Tras 60 segundos, se retorna a la expiración principal o *major timeout*.

Por defecto, la expiración principal hará que el cliente envíe un mensaje a la consola y empiece de nuevo, con una expiración del doble de tiempo. Potencialmente, esto podría mantenerse eternamente. En este caso se habla de montaje físico o *hard-mount*. La otra variedad, el montaje lógico o *soft-mount*, genera un mensaje de error de E/S al proceso llamante cuando se produce la expiración principal. El error no se propaga al proceso hasta que hace una nueva llamada a *write(2)*, por lo que esto, junto con la política de escritura desde la cache, hace que no se sepa realmente si una operación de escritura ha tenido éxito o no, a menos que el volumen esté montado de forma física.

En general, se recomienda el montaje físico salvo en caso de tratarse de información no crítica, como la de servidores de FTP o particiones de noticias. En entornos críticos (por ejemplo, estaciones de trabajo X con dependencia de servidores de aplicaciones X Window) no debe usarse el montaje lógico a riesgo de perder las conexiones si en un momento se satura o desactiva la red por algún motivo. Una solución alternativa a usar montajes físicos es aumentar el valor de la opción *timeo*, o bien usar montajes físicos pero permitiendo el envío de señales para interrumpir las esperas en caso de necesidad.

Normalmente, el demonio *mountd* llevará de alguna forma un registro de qué directorios están montados desde qué máquinas. El programa *showmount*, incluido en el paquete de aplicaciones NFS, permite consultar esta información. De todas formas, el *mountd* de Linux aun no lleva estos registros.

11.3 Demonios de NFS

Si desea proporcionar un servicio NFS a otras máquinas, deberá ejecutar en el servidor los programas *nfsd* y *mountd*. Son programas basados en RPC, por lo que no son arrancados desde el *inetd*, sino lanzados como demonios en tiempo de arranque, y registrados en el mapeador de puertos de RPC. Por lo tanto, debe asegurarse que previamente ha sido lanzado el programa *rpc.portmap*. Normalmente, esto implica las siguientes líneas en los *scripts* de arranque *rc*:

```
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 servidor NFS proporciona a sus clientes viene dada en valores numéricos de identificador de usuario (*uid*) y de grupo (*gid*). Por lo tanto, esto resultará útil si clientes y servidores tienen el mismo mapa de usuarios y grupos, lo que sucede cuando dicho mapa se obtiene en todos los nodos desde un servidor NIS central.

Sin embargo, hay veces que esto no sucede. En lugar de actualizar los uids y gids del cliente para ponerse de acuerdo con los del servidor, puede usarse el demonio *ugidd* para hacer este trabajo. Utilizando la opción *map_daemon* explicada después, se indicará a *nfsd* que establezca una correspondencia entre uid/gid del servidor y del cliente, con la ayuda, en el cliente, de *ugidd*.

ugidd es un servidor basado en RPC, y se inicia también en los scripts *rc*, con una línea:

```
if [ -x /usr/sbin/rpc.ugidd ]; then
        /usr/sbin/rpc.ugidd; echo -n " ugidd"
fi
```

11.4 El fichero *exports*

Mientras que las opciones anteriores se aplican a la configuración del cliente NFS, hay otras opciones que se aplican al servidor, que afectan a su relación con cada posible cliente. Estas opciones se incluyen en el fichero de sistemas exportados */etc/exports*.

Por defecto, *mountd* no permitirá a nadie montar directorios de su máquina. Para permitir que algún nodo monte un directorio, éste debe estar *exportado*, es decir, especificado en el fichero de exportación. Un ejemplo de dicho fichero es el siguiente:

```
# Fichero de exportacion para vlager (/etc/exports)
/home           vale(rw) vstout(rw) vlight(rw)
/usr/X386       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 la lista de máquinas que pueden acceder a él por NFS. Un nombre de máquina puede especificarse con su nombre internet completo, aunque también se permite el uso de los comodines *** y *?*, que se interpretan como en el shell de Bourne. Por ejemplo, **lab*.prueba.com** encaja con cualquier nodo con nombre similar a **laboratorio.prueba.com** o **lab12.prueba.com**, etc. Cuando en una línea de */etc/exports* no se indique el nombre del nodo, se asume que cualquier máquina podrá montar el directorio (así sucede en nuestro ejemplo con */home/ftp*).

mountd usa la llamada *gethostbyaddr(2)* para comprobar si el cliente demandante tiene un nombre de los que aparecen en */etc/exports*. Con DNS, la llamada retorna el nombre canónico con lo que debe evitar usar nombres de alias en el fichero de exportación[6]. Si no usa DNS, el nombre devuelto por la llamada anterior será el primer nombre que coincida con el IP del demandante, en el fichero */etc/hosts*.

Tras el nombre del nodo autorizado, se puede encerrar entre paréntesis un conjunto de opciones separadas por comas. Dichas opciones son:

insecure Permitir acceso no autentificado desde ese nodo.

unix-rpc Requerir autentificación RPC del dominio Unix para este nodo. Se trata simplemente de que las peticiones se originen en un puerto reservado (es decir, inferior al 1024). Esta opción está activa por defecto.

secure-rpc Requerir autentificación RPC segura para este nodo. Aún no está implementado. Se sugiere ver la documentación de Sun al respecto (véase, "Secure RPC").

kerberos Requerir autentificación Kerberos. Tampoco se ha implementado aún. Se sugiere consultar la documentación del MIT.

root_squash Se trata de una opción de seguridad que deniega acceso a nivel de superusuario, traduciendo el identificador uid recibido (0) al del usuario **nobody**. Es decir, cualquier petición NFS del usuario **root** será tomada como si fuera del usuario **nobody**.

no_root_squash
Evita la restricción anterior. Es una opción por defecto.

ro Monta la jerarquía de ficheros en modo de sólo lectura. Es una opción por defecto.

rw Monta el directorio con permiso para leer y escribir en él.

[6]Ver capítulo 6

link_relative Convierte enlaces simbólicos absolutos (que empiezan con una barra de directorio, "/") en enlaces relativos colocando los prefijos "../" que sean necesarios para hacer que apunten a la raíz del servidor. Esta opción sólo tiene sentido cuando se monta un sistema de ficheros completo y no sólo un directorio. Así, si montamos dicho sistema bajo */mnt* y existe en */mnt/sub* un enlace *fichero* → */tmp/fichero* se convertirá a *fichero* → *../tmp/fichero* logrando así que el enlace sirva para algo.

Es una opción activa por defecto.

link_absolute Deja los enlaces absolutos como estaban (es la opción habitual en servidores NFS de Sun).

map_identity La opción *map_identity* indica al servidor que asuma que el cliente utiliza el mismo mapa de uids y gids que el servidor. Es una opción por defecto.

map_daemon Esta opción indica al servidor NFS que no comparte el mapa de usuarios con el del cliente. Con ello, las comparaciones de uids y gids se harán mediante una lista de mapeado entre ambos que se construirá llamando al demonio *ugidd* del cliente.

Cualquier error analizando el fichero de exportaciones durante el arranque del servidor *nfsd* o *mountd* será enviado a nivel de notificación (*notice*) al registro del sistema (*syslogd*).

Obsérvese que los nombres de los nodos se obtienen a partir de las direcciones IP mediante resolución inversa, con lo que el sistema de resolución deberá tener una adecuada configuración en este punto. Si utiliza BIND y la seguridad le preocupa especialmente, deberá activar chequeo de nombres falsos (*spoofing*) en el fichero *host.conf*.

11.5 El sistema de automontado en Linux

A veces es ineficiente mantener montados todos los volúmenes NFS de uso potencial. Una alternativa es usar un demonio de *automontado*. Se trata de un demonio que automáticamente monta los volúmenes cuando se necesitan y los desmonta tras un tiempo de inactividad. Además, sirve para poder montar los mismos ficheros de un lugar diferente. Por ejemplo, puede mantener varias copias de las utilidades de X Window y a la hora de ser necesitadas, intentar montar cada copia hasta conseguirlo.

El programa de automontado para Linux se llama *amd*. Ha sido escrito inicialmente por Jan-Simon Pendry para luego encargarse Rick Sladkey de portarlo a Linux. La versión actual es la *5.3*.

Explicar el uso de *amd* excede los objetivos de este capítulo. El mejor manual se encuentra en las fuentes: un fichero *texinfo* con información muy detallada.

Capítulo 12

Administración de Taylor UUCP

12.1 Historia

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)[1], HoneyDanBer UUCP, o HDB abreviado. El nombre fue derivado de los nombres de los autores, P. Honeyman, D. A.Novitz, y B. E. Redman. HDB fue creado para eliminar algunas deficiencias de la Versión 2. Por ejemplo, se añadieron

[1]N. del T.: BNU son las siglas de Basic Network Utilities.

nuevos protocolos de transferencia, y el directorio de cola fue dividido de manera que ahora sólo haya un directorio para cada ordenador con el que mantega tráfico de UUCP.

La implementación de UUCP que se distribuye con Linux es Taylor UUCP 1.04,[2] que es la versión en la que se basa este capítulo. La versión 1.04 de Taylor UUCP esta disponible desde Febrero de 1993. Ademas de los tradicionales ficheros de configuración, Taylor UUCP tambien se puede compilar para que use ficheros de configuración de un nuevo estilo: el estilo "Taylor".

La versión 1.05 ha aparecido recientemente, y pronto se incluirá en la mayoría de las distribuciones. Las diferencias entre estas versiones afectan en su mayor parte a aspectos que usted nunca usará, así que con lo poco que contamos aquí le debería bastar para configurar Taylor UUCP 1.05 en su sistema.

Tal y como se incluye en la mayoría de las distribuciones de Linux, el Taylor UUCP viene compilado para ser compatible con BNU, con el estilo de configuración de Taylor, o ambos. Dado que el segundo es mucho más flexible, y probablemente mas fácil de entender que los usualmente oscuros ficheros de configuración de BNU, describiré aquí el estilo Taylor.

El propósito de este capítulo no es ofrecer una explicación exhaustiva de las opciones de la línea de comando para los comandos 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 "Ficheros de configuración de UUCP", 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, *uux* y *uucp*, el paquete UUCP contiene algunos comandos más, usados solamente para fines administrativos. Se usan para observar el tráfico de UUCP en su nodo, deshacerse de viejos ficheros de registro, o crear estadísticas. No describiremos ninguna de estas utilidades, porque son periféricas a las tareas principales de UUCP. Dichas utilidades están bien documentadas y son bastante fáciles de entender. Sin embargo, hay una tercera categoría, que forma los "motores" del UUCP. Éstas son *uucico* (donde "cico" significa "copy-in copy-out"), y *uuxqt*, que ejecuta trabajos enviados desde sistemas remotos.

[2]Escrito por Ian Taylor en 1993.

12.1.1 Más Información Sobre UUCP

Aquellos que no encuentren todo lo que necesiten en este capítulo pueden leer la documentación que viene con el paquete. Viene en un grupo de ficheros de `texinfo` que describen la configuración usando el estilo de configuración de Taylor. Los ficheros texinfo se pueden convertir a DVI y a ficheros "GNU info" usando *tex* y *makeinfo* respectivamente.

Si quiere usar ficheros de configuración de BNU o incluso de la Versión 2 (!), existe un libro muy recomendable, "Managing UUCP and Usenet" ([OReilly89]). Es muy útil. Otra buena fuente de información sobre UUCP en Linux es el UUCP-HOWTO de Vince Skahan, que aparece regularmente en **comp.os.linux.announce**.

También hay un grupo de noticias sobre UUCP, llamado **comp.mail.uucp**. Si usted tiene preguntas específicas sobre Taylor UUCP, puede que sea mejor que las pregunte en ese grupo, en vez de en los grupos **comp.os.linux**.

12.2 Introducción

12.2.1 Disposición de Transferecias de UUCP y Ejecución Remota

El concepto de *trabajos* es vital para entender el UUCP. Cada transferencia que un usuario inicia con *uucp* o *uux* se denomina un trabajo. Se compone de un *comando* para ser ejecutado en un sistema remoto, y una colección de *ficheros* para ser transferidos entre redes. Una de estas partes puede faltar.

Por ejemplo, el siguiente comando describe un trabajo UUCP, que conlleva copiar el fichero *netguide.ps* en el ordenador **pablo**, y ejecutar luego el comando *lpr* para imprimir el fichero.

```
$ uux -r pablo!lpr !netguide.ps
```

Generalmente UUCP no llama al sistema remoto inmediatamente para ejecutar el trabajo (se puede hacer con *kermit*). En lugar de esto, UUCP guarda la descripción del trabajo temporalmente. Esto se denomina *spooling*.[3] El arbol de directorios en el que se almacenan los trabajos se llama *directorio de cola*, y se encuentra normalmente en */var/spool/uucp*. En nuestro ejemplo, la descripción del trabajo contendría información sobre el comando remoto que hay que ejecutar (*lpr*), el usuario que ha iniciado la ejecución, y otro par de

[3] N. del T.: *spooling* puede traducirse por "poner en una cola". Spooling se refiere a la acción de poner varios trabajos en una cola para ser usados más tarde.

cosas. Ademas de la descripción del trabajo, UUCP tiene que guardar el fichero de datos *netguide.ps*.

La localización exacta y la nomenclatura de los ficheros de cola puede variar, dependiendo de las opciones de compilación. Los UUCPs que son compatibles con HDB generalmente guardan los ficheros de cola en un directorio llamado */var/spool/uucp/***`máquina`**, donde **`máquina`** es el nombre del ordenador remoto. Si fue compilado para usar ficheros de configuración de Taylor, UUCP crea subdirectorios bajo el directorio de cola específico a un ordenador para diferentes tipos de ficheros de cola.

En intervalos regulares UUCP se conecta al sistema remoto. Cuando se ha establecido una conexión, UUCP transfiere los ficheros que describen el trabajo, junto con los ficheros de datos. Los trabajos que entran en el ordenador remoto no se ejecutan de inmediato, sino despues de que la conexión haya finalizado. Esto lo hace *uuxqt*, que también se ocupa de reenviar cualquier trabajo que esté designado para otro ordenador diferente.

Para diferenciar trabajos importantes y trabajos menos importantes, UUCP asocia un *nivel* a cada trabajo. El nivel es una sola letra o dígito, de *0* a *9*, de *A* a *Z* y de *a* a *z*, con precedencia decreciente. El correo se suele poner en cola con nivel B o C, mientras que las noticias se suelen poner con nivel N. Los trabajos con un nivel más alto se transfieren más pronto. Los niveles pueden ser asignados usando la opción `-q` cuando se ejecuta *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 por defecto 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.

12.2.2 El Funcionamiento Interno de *uucico*

◇ Para comprender por qué *uucico* necesita saber ciertas cosas, hemos de revisar cómo se conecta a un sistema remoto.

Cuando usted ejecuta *uucico -s* **`sistema`** desde la línea de comandos, 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*.

A continuación viene la *fase de negociación*[4]: El maestro envía su nombre, ademas 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 telefono 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.

No vamos a profundizar en más detalle: para esto, diríjase a las fuentes o a cualquier buen libro sobre UUCP. Tambien existe un artículo muy antiguo en la red, escrito por David A. Novitz, que da una descripción detallada del protocolo UUCP. La FAQ[5] de Taylor UUCP también trata algunos detalles de cómo UUCP esta implementado. Se puede encontrar regularmente en **comp.mail.uucp**.

[4] N. del. T.: Literalmente, del inglés *handshake* o "apretón de manos". En este contexto significa "negociación de protocolos", fase en la que los programas *uucico* de cada extremo de comunicación deciden qué protocolo común usar para enviar los ficheros.

[5] N. del T.: Del inglés *Frecuently Asked Questions* o Lista de Preguntas Comunes.

12.2.3 Opciones de la línea de comandos de *uucico*

En esta sección de describen las opciones más importantes de la línea de comandos del programa *uucico*. Para obtener una lista completa, consulte la página del manual de *uucico(1)*.

`-s` *sistema* — Llamar al mencionado *sistema* si no está prohibido por restricción de la hora de llamada.

`-S` *sistema* — Llamar al *sistema* incondicionalmente.

`-r1` — Comenzar *uucico* en modo master. Este es el valor por defecto cuando se usa `-s` o `-S`. Por sí sola, la opción `-r1` hace que *uucico* intente llamar todos los sistemas en *sys*, a no ser que esté prohibido por la hora de llamada o el número permitido de reintentos.

`-r0` — Comenzar *uucico* en modo esclavo. Este es el valor por defecto cuando no se usan `-s` ni `-S`. En modo esclavo, las entrada y salida estándar se suponen conectadas a un puerto serie, o al puerto de TCP especificado por la opción `-p` si se usa ésta.

`-x` *tipo*, `-X` *tipo*

Activar la información para resolver problemas del tipo especificado. Se pueden especificar varios tipos en una lista separada por comas. Los siguientes son tipos válidos: *abnormal*, *chat*, *handshake*, *uucp-proto*, *proto*, *port*, *config*, *spooldir*, *execute*, *incoming* y *outgoing*. Usando *all*, todas las opciones se activan. Por razones de compatibilidad con otras implementaciones de UUCP, se puede especificar un número en vez del tipo, lo cual activa las *n* primeras opciones de la anterior lista.

Los mensajes generados se registran en el fichero *Debug* bajo */var/spool/uucp*.

12.3 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.

12.3.1 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 mas 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 */usr/lib/uucp* o un subdirectorio de éste. Algunas distribuciones de Linux contienen programas de UUCP que tienen soporte para ambos ficheros de HDB y Taylor, y usan diferentes subdirectorios para cada grupo de ficheros de configuración. Normalmente hay 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. [6]

El fichero central de configuración es */usr/lib/uucp/config*, y se usa para establecer los parámetros generales. El más importante de ellos (y por ahora, el único), es el nombre de su ordenador anfitrión de UUCP. En la Cervecera Virtual, se usa **vstout** como su ordenador de conexión a UUCP.

```
# /usr/lib/uucp/config - Fichero principal de configuracion de UUCP
hostname        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
```

[6]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.

```
system          pablo
time            Any
phone           123-456
port            serial1
speed           38400
chat            ogin: neruda ssword: lorca
```

port especifica el puerto a usar, y *time* especifica las horas a las que se puede llamar a ese ordenador. *chat* describe la macro del diálogo de entrada – la secuencia de caracteres que hay que intercambiar para permitir a *uucico* que conecte con **pablo**. Volveremos a las macros más tarde. El comando *port* no usa un nombre de fichero de un dispositivo como */dev/cua1*, sino que usa el nombre de una entrada en el fichero *port*. Se pueden asignar estos nombres como se desee siempre y cuando hagan 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/cua1* (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*.

```
# /usr/lib/uucp/port - puertos de UUCP
# /dev/cua1 (COM2)
port            serial1
type            modem
device          /dev/cua1
speed           38400
dialer          nakwell
```

La información que afecta al propio marcador se mantiene en otro fichero, llamado *dial*. Para cada tipo de marcador, contiene básicamente la secuencia de comandos necesarios 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:

```
# /usr/lib/uucp/dial -  informacion por marcador.
# NakWell modems
dialer          nakwell
chat            "" ATZ OK ATDT\T CONNECT
```

La línea que empieza con *chat* especifica el diálogo del módem, que no es sino la secuencia de comandos enviados y recibidos del módem para inicializarlo, y para hacerle marcar el

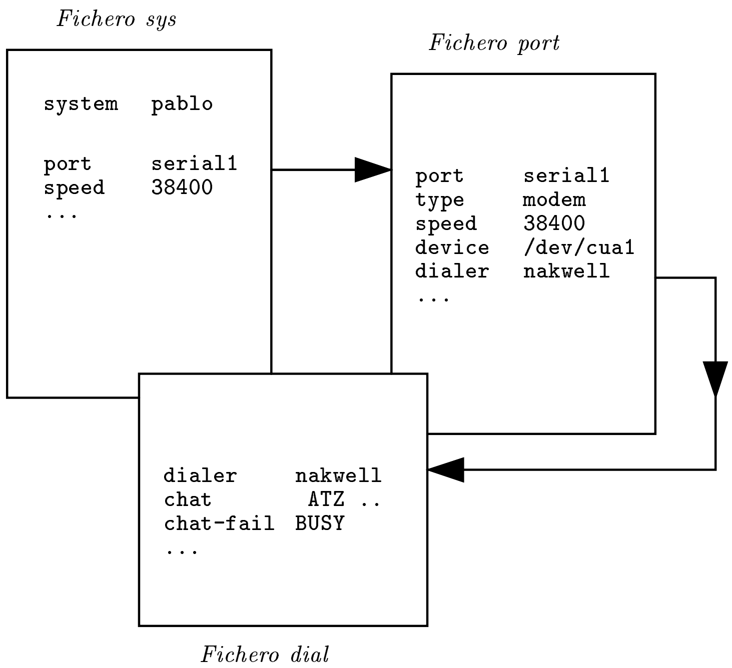

Figura 12.1: Interacciones de los Ficheros de Configuración de Taylor UUCP.

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 *uucico* utiliza estos ficheros de configuración, suponga que utiliza el comando

```
$ uucico -s pablo
```

en la línea de comandos. Lo primero que *uucico* hace es buscar **pablo** en el fichero *sys*. A partir de la entrada en el fichero *sys* para **pablo**, el programa ve que debería usar el puerto *serial1* para establecer la conexión. El fichero *port* le dice que ese puerto es un puerto de módem, y que tiene un módem NakWell conectado.

uucico busca ahora en *dial* la entrada que describe el módem NakWell, y al encontrar una, abre el puerto serie */dev/cua1* y ejecuta el diálogo de marcación. Esto es, envía "`ATZ`", espera a que el módem responda con "`OK`", etc. Cuando se encuentre los caracteres "\\T", sustituye el número de teléfono (123–456) obtenido del fichero *sys*.

Cuando el módem devuelve `CONNECT`, la conexión se ha establecido, y el diálogo de marcación se ha completado. *uucico* ahora vuelve al fichero *sys* y ejecuta el diálogo de entrada. En nuestro ejemplo, esperaría la pregunta "`login:`", enviaría su nombre de usuario (`neruda`), esperaría la pregunta "`password:`", y enviaría la clave, "`lorca`".

Tras completar la autorización, se supone que el sistema remoto ejecuta su propio *uucico*. Los dos entrarán entonces en la fase de negociación de protocolos descrita en la sección anterior.

La relación de dependencia de los ficheros de configuración se puede ver en la figura 12.1.

12.3.2 Lo que UUCP necesita saber

Antes de empezar a escribir los ficheros de configuración, debe conseguir cierta información que UUCP necesita.

Primero tiene que saber en qué dispositivo serie está su módem. Normalmente, los puertos (de DOS) COM1 hasta COM4 se corresponden con los ficheros de dispositivos */dev/cua0* hasta */dev/cua3*. La mayoría de las distribuciones, como Slackware, crean un enlace simbólico */dev/modem* al fichero de dispositivo donde esta el módem, y configuran *kermit*, *seyon*, etc, para que usen este fichero. En este caso, usted también debería usar usar */dev/modem* en la configuración de UUCP.

La razón para esto es que todos los programas, para llamar por teléfono, usan unos *ficheros cerrojo* 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..cua1*. 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 obtener más detalles sobre cómo configurar sus puertos serie, lea el capítulo 4.

A continuación tiene que averiguar a qué velocidad se comunicarán su módem y 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 2400bps (bits por segundo). Usando protocolos de compresión como V.42bis, la velocidad real de transferencia puede alcanzar los 9600bps.

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.[7]

[7]Si solo 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**.

También necesitará saber *exactamente* cómo entrar en el sistema. Por ejemplo, ¿tiene que pulsar la tecla BREAK 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 *script* 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.

12.3.3 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 a ordenadores que usted llama directamente, o en una red local, el nombre no tiene que ajustarse a ninguna regla.[8]

Sin embargo, si usa UUCP para tener una conexión a correo y noticias, se debería pensar en registrar el nombre con el proyecto de Mapa de UUCP, que se describe en el capítulo 13. Incluso si usted participa en un dominio, podría considerar el tener un nombre oficial de UUCP para su ordenador.

Frecuentemente la gente elige su nombre de UUCP para que corresponda con el primer componente de su nombre de dominio completamente cualificado. Suponga que la dirección de su dominio es **swim.twobirds.com**, entonces su nombre de UUCP podría ser **swim**. Piense en los nodos de UUCP como si se conociesen entre ellos por el nombre propio. Por supuesto, también puede usar un nombre de UUCP completamente desvinculado de su nombre de dominio, y por consiguiente, un nombre no cualificado.

Sin embargo, asegúrese de no usar el nombre no cualificado en direcciones de correo a no ser que lo haya registrado como su nombre de UUCP oficial.[9] Lo mejor que puede pasar es que el correo dirigido a un nombre de UUCP no registrado se pierda en algún *agujero negro digital.* Si utiliza un nombre que alguien ya está usando, este correo será dirigido a ese sitio, y le causará al administrador del correo un sinfín de dolores de cabeza.

Los programas de UUCP usan el nombre devuelto por *hostname* como el nombre de UUCP por defecto. Este nombre se encuentra normalmente en la macro */etc/rc.local*. Si su nombre de UUCP es diferente del que le dió a su ordenador, tiene que usar la opción *hostname* en el fichero *config* para indicar a *uucico* su nombre de UUCP. Esto se describe a continuación.

[8] La única limitación es que no puede ser más largo que 7 caracteres, para no confundir a algunos nodos con sistemas de ficheros que imponen un estrecho límite en los nombres de ficheros.

[9] El Proyecto de Mapa de UUCP registra todos los nodos de UUCP de todo el mundo y comprueba que sean únicos. Para registrar su nombre de UUCP, pregúntele a los responsables del nodo que gestiona su correo; ellos podrán ayudarle.

12.3.4 Ficheros de configuración Taylor

Volvemos ahora a los ficheros de configuración. Taylor UUCP obtiene su información de los siguientes ficheros:

config Este es el fichero principal de configuración. Aquí puede definir el nombre de UUCP de su ordenador.

sys Este fichero describe todos los nodos conocidos por el sistema. Por cada nodo, especifica su nombre, a qué horas llamarlo, que número marcar, que tipo de dispositivo usar, y cómo entrar.

port Contiene entradas que describen cada puerto disponible, junto con la velocidad de línea soportada y las instrucciones de marcación.

dial Describe las instrucciones de marcación usados para establecer una conexíon telefónica.

dialcode Contiene expansiones simbólicas para códigos de marcación.

call Contiene el nombre de usuario y la clave a usar cuando llame a un sistema. No se suele usar.

passwd Contiene nombres de usuario y claves que los sistemas pueden usar cuando entren en su ordenador. Este fichero se usa sólo cuando *uucico* hace su propia comprobación de claves.

Los ficheros de configuración de Taylor se componen generalmente de líneas que contienen pares palabra - valor. Una almohadilla (#) indica un comentario que ocupa toda una línea. Para usar una almohadilla por sí misma, puede poner una barra invertida delante de la almohadilla.

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 comandos completa en los documentos Texinfo.

Cuando crea que ha configurado su sistema de UUCP completamente, puede comprobarlo usando la utilidad *uuchk* (que se encuentra en */usr/lib/uucp*). *uuchk* lee sus ficheros de configuración, e imprime un informe detallado de los valores de configuración usados para cada sistema.

12.3.5 Opciones Generales de Configuración – el Fichero *config*

Normalmente no usará este fichero para otra cosa que describir su nombre de nodo UUCP. Por defecto, UUCP usará el nombre establecido con el comando *hostname*, pero generalmente es una buena idea configurar el nombre de UUCP explícitamente. A continuación mostramos un fichero de ejemplo:

```
# /usr/lib/uucp/config - fichero principal de configuracion de UUCP
hostname        vstout
```

Por supuesto, también existen otros parámetros configurables aquí, como los referentes al nombre del directorio de colas, o los derechos de acceso para el UUCP anónimo. Esto último se describirá en una sección posterior.

12.3.6 Cómo informar a UUCP sobre otros sistemas – el fichero *sys*

El fichero *sys* describe los sistemas que su ordenador conoce. Una entrada comienza con la palabra *system*; las líneas siguientes hasta la siguiente *system* proporcionan detalles sobre los parámetros específicos sobre ese sistema o nodo. Comúnmente, una entrada de un sistema definirá parametros tales como el número de teléfono y el diálogo de entrada.

Los parámetros antes de la primera línea con *system* determinan los valores por defecto usados para todos los sistemas. Lo normal es que los parámetros de protocolos y similares se incluyan en la sección por defecto.

A continuación se tratan los campos más importantes con cierto detalle.

Nombre del sistema

El comando *system* especifica el nombre del sistema remoto. Tiene que especificar el nombre correcto del sistema remoto, no un alias que usted se invente, porque *uucico* lo verificará con la información que reciba del otro sistema una vez se conecte.[10]

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 *alternavias*, que se describen más adelante.

[10] Algunas de las versiones 2 de UUCP antiguas no envían su nombre cuando son llamadas; sin embargo, las implementaciones más recientes sí lo hacen, y Taylor UUCP también.

Número de teléfono

Si para conectar con el sistema remoto hace falta una línea de teléfono, el campo *phone* especifica el número que tiene que marcar el módem. Puede incluir varios separadores que son interpretados por el proceso de marcación efectuado por *uucico*. Un signo igual (=) significa esperar un tono secundario, y un guión genera una pausa de un segundo. Por ejemplo, algunas instalaciones de teléfono se atascan si no deja una pausa entre un prefijo de una compañía y el número de teléfono.

Cualquier lista de caracteres se puede usar para esconder información que depende de cada nodo, como el prefijo de provincia. Estos caracteres se traducen en un código de marcación usando el fichero *dialcode*. Suponga que tiene el siguiente fichero *dialcode*:

```
# /usr/lib/uucp/dialcode - traduccion de codigos de marcacion
Bogoham         024881
Coxton          035119
```

Con estas traducciones, se puede usar un número de teléfono como *Bogoham7732* en el fichero *sys*, lo cual hace las cosas un poco más legibles.

Puerto y Velocidad

Las opciones *port* y *speed* se usan para elegir el dispositivo usado para llamar al sistema remoto, y la velocidad máxima del dispositivo.[11] Una entrada *system* puede usar cualquiera de las dos opciones solas, o ambas. Cuando se busca un dispositivo apropiado en el fichero *port*, sólo se eligen aquellos puertos cuyo nombre y/o velocidad coinciden.

Normalmente es suficiente con usar la opción *speed*. Si solo tiene un dispositivo serie definido en *port*, *uucico*, de cualquier modo, siempre escogerá el correcto, así que sólo tiene que especificar la velocidad deseada. Si tiene varios modems conectados a su sistema, tampoco es una buena idea nombrar un puerto en particular, porque si *uucico* encuentra que hay varios puertos con el mismo nombre, trata de usarlos todos hasta que encuentra uno que no está en uso.

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

[11] La velocidad en baudios del terminal (tty) tiene que ser por lo menos igual o mayor que la velocidad máxima de transmisión.

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 (`ogin:` y `ssword:`) no contienen 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:`.

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 razon, 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 (*timeout*). 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 return antes de que aparezca la pregunta de entrada.

```
"" \n\r\d\r\n\c ogin:-BREAK-ogin: vstout ssword: catch22
```

Hay un par de tiras 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 tira vacía. Le dice a *uucico* que no espere nada, sino que siga con la siguiente tira de enviado immediatamente.

\t
: Un carácter de tabulador.

\r
: Un carácter de retorno de línea.

\s
: El carácter de espacio. Lo necesitamos para incluir espacios en un diálogo.

\n
: Carácter de nueva línea.

\\
: Carácter de barra invertida.

En tiras de caracteres de envío se pueden incluir, ademas de los mencionados anteriormente, los siguientes caracteres:

EOT Carácter de fin de la trasmisión (`^D`).

BREAK Carácter Break.

c Suprime el envío del carácter nueva línea al final de cada tira de caracteres.

d Retrasar el envío 1 segundo.

E Activar 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 defecto.

e Desactivar la comprobación de eco.

K Lo mismo que *BREAK*.

p Pausa de una fracción de un segundo.

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 que 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.

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 horas de llamada es con la entrada *time*, seguida de una tira de caracteres que consta de dos campos, día y hora. El día puede ser cualquiera de los siguientes: *Mo, Tu, We, Th, Fr, Sa, Su* (que corresponden a Lunes, Martes, Miércoles, Jueves, Viernes, Sábado y Domingo, respectivamente) combinados, *Any* (cualquiera), *Never* (nunca), o *Wk* para los días laborables. La hora consiste en dos números de un reloj de 24 horas, separados por un guión. Especifican el grupo de horas durante las que se pueden efectuar llamadas. La combinación de los símbolos se escribe sin ningún espacio en blanco entre ellos. Se pueden especificar varios grupos de día-hora separados por comas. Por ejemplo,

```
time            MoWe0300-0730,Fr1805-2000
```

permite llamadas en Lunes y Miercoles, 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.

El comando *time* tiene un segundo argumento 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. Por defecto, *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.

El comando *timegrade* le permite añadir un rango máximo de cola a un calendario. Por ejemplo, asumiendo que usted tiene los siguientes comandos *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 (*news*) (normalmente con rango N) serán transferidas sólo durante la noche y los fines de semana.

Al igual que *time*, el comando *timegrade* acepta un intervalo de reintento en minutos como un tercer argumento opcional.

Sin embargo, hay que hacer una observación: la opción *timegrade* sólo se aplica a lo que *su* sistema envía; el sistema remoto puede transferir todo lo que le plazca. Usted puede usar la opción *call-timegrade* para forzar explícitamente que envíe solamente trabajos sobre cierto rango de cola; pero no hay ninguna garantía de que obedecerá esta petición.[12]

Igualmente, el campo *timegrade* no se comprueba cuando un sistema remoto hace una llamada a éste, de manera que cualquier trabajo puesto en la cola para el sistema que llama al nuestro será enviado. Sin embargo, el sistema remoto puede pedir explícitamente a nuestro *uucico* que se mantenga a sí mismo en un cierto rango de cola.

12.3.7 Qué dispositivos hay – el fichero *port*

El fichero *port* indica a *uucico* qué puertos tiene disponibles. Pueden ser puertos de módem, pero cualquier otro tipo, como líneas serie directas o sockets de TCP también se pueden usar.

Al igual que el fichero *sys*, el fichero *port* consta de entradas separadas que empiezan con la palabra *port*, seguida del nombre del puerto. Este nombre puede ser usado por la palabra *port* en el fichero *sys*. El nombre no tiene por qué ser único; si hay varios puertos con el mismo nombre, *uucico* intentará cada uno de los puertos hasta que encuentre uno que no está siendo utilizado.

El comando *port* tiene que estar seguido por el comando *type* que especifica qué tipo de puerto se está describiendo. Tipos válidos son *modem*, *direct* para comunicaciones directas, y *tcp* para sockets de TCP. Por defecto, cuando el comando *port* no se incluye en el fichero, el tipo de puerto asumido será *modem*.

En esta sección sólo hablaremos de puertos de módem; los puertos de TCP y las líneas directas serán tratados en una sección posterior.

[12]Si el sistema remoto esta ejecutando Taylor UUCP, obedecerá.

Para puertos directos y de módem, tiene que especificar el dispositivo para llamar usando la directiva *device*. Usualmente es el nombre de un fichero de dispositivo en el directorio */dev*, como por ejemplo */dev/cua1*.[13]

En el caso de un dispositivo de módem, la directiva *port* también determina qué tipo de módem está conectado al puerto. Cada tipo de módem tiene que configurarse de manera diferente. Incluso los modems que dicen ser compatibles con Hayes no tienen por qué ser realmente compatibles entre sí mismos. Por lo tanto, tiene que decirle a *uucico* cómo inicializar el módem y cómo hacer que marque 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 usando el comando *dialer*.

Es posible que usted quiera usar el módem de maneras diferentes, dependiendo del sistema al que está llamando. Por ejemplo, algunos modems antiguos no entienden cuándo un módem rápido trata de conectar a 14400bps; simplemente desconectan la línea en vez de negociar la conexión a 9600bps por ejemplo. Si sabe que el ordenador **plasta** usa un módem tan tonto, usted tiene que configurar su módem de manera diferente cuando llame a ese ordenador. Para hacer esto, necesita una entrada adicional del comando *port* en el fichero *port* que especifica un marcador diferente. Ahora puede dar un nombre diferente al nuevo puerto, como por ejemplo *serial1-lento*, y usar la directiva *port* en la entrada del sistema **plasta** en el fichero *sys*.

Otra manera de distinguir los puertos es por la velocidad que usan. Por ejemplo, las dos entradas *port* de la situación anterior pueden ser así:

```
# NakWell modem; conexion a velocidad alta.
port            serial1         # nombre del puerto
type            modem           # puerto modem
device          /dev/cua1       # esto es COM2
speed           38400           # velocidad soportada
dialer          nakwell         # marcador normal

# NakWell modem; conexion lenta
port            serial1         # nombre del puerto
type            modem           # puerto modem
device          /dev/cua1       # esto es COM2
speed           9600            # velocidad soportada
dialer          nakwell-slow    # no intentar alta velocidad
```

La entrada de sistema para el ordenador **plasta** usaría ahora *serial1* como el nombre del puerto, pero pediría usar la velocidad de 9600bps solamente. *uucico* usará automáticamente

[13] Hay quien usa los dispositivos *ttyS**, que son solamente para aceptar llamadas.

la segunda entrada de *port*. Todos los otros ordenadores con velocidad de 38400bps en la entrada de sistema serán llamados usando la primera entrada de *port*.

12.3.8 Cómo marcar un número – el fichero *dial*

El fichero *dial* describe cómo se usan los distintos marcadores. Tradicionalmente, UUCP habla de "marcadores" en vez de modems, porque en los viejos tiempos era normal que un dispositivo de marcación automático (que era caro) sirviese un banco entero de modems. Hoy, la mayoría de los modems tienen soporte para marcar incluido, así que la distinción tiende a desaparecer.

De cualquier modo, cada marcador o módem puede necesitar una configuración diferente. Se puede describir cada uno de ellos en el fichero *dial*. Las entradas en *dial* empiezan con el comando *dialer* que indica el nombre del marcador.

La entrada más importante, aparte de ésta, es el diálogo del módem, especificado por el comando *chat*. Similar al diálogo de entrada (login), consta de una secuencia de caracteres que *uucico* envía al marcador y de la secuencia que espera recibir como respuesta. Se usa normalmente para reiniciar el módem a un estado conocido, y marcar el número. El siguiente ejemplo de una entrada de un marcador muestra un diálogo típico para un módem compatible con Hayes:

```
# NakWell modem; conexion a alta velocidad
dialer          nakwell         # nombre del marcador
chat            "" ATZ OK\r ATH1E0Q0 OK\r ATDT\T CONNECT
chat-fail       BUSY
chat-fail       ERROR
chat-fail       NO\sCARRIER
dtr-toggle      true
```

El diálogo del módem comienza con `""`, es decir, que espera una cadena vacía. *uucico* entonces envía el primer comando (`ATZ`) de inmediato. `ATZ` es el comando de Hayes para reiniciar el módem. Entonces espera hasta que el módem envíe `OK`, y a continuación envía el siguiente comando que desactiva el eco local, y cosas parecidas. Después de que el módem envíe `OK` otra vez, *uucico* envía el comando de marcación (`ATDT`). La secuencia de escape `\T` en esta cadena es reemplazada con el número de teléfono obtenido de la entrada de sistema en el fichero *sys*. *uucico* espera a que el módem devuelva la cadena `CONNECT`, que indica que se ha establecido una conexión exitosa con el módem remoto.

A menudo el módem no puede conectarse con el sistema remoto, por ejemplo si el otro sistema esta conectado con otro ordenador y la línea está ocupada. En este caso, el módem devuelve algún mensaje de error indicando la razón. Los diálogos de módem no pueden detectar estos mensajes; *uucico* seguirá esperando la cadena esperada hasta que un temporizador se agote. El fichero de recopilación de información (log) de UUCP mostrará solamente el mensaje "timed out in chat script" (tiempo agotado en la macro de diálogo) en vez de la razón real.

Sin embargo, Taylor UUCP le permite informar a *uucico* sobre estos mensajes de error usando el comando *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 log de UUCP.

El último comando en el ejemplo anterior indica a UUCP que cambie la línea DTR antes de empezar el diálogo de módem. La mayoría de los modems se pueden configurar para conectarse cuando detectan un cambio en la línea DTR, y entrar en modo de comando.[14]

12.3.9 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 el fichero *sys*, hay que especificar al sistema a llamar con TCP de la siguiente forma:

```
system          gmu
address         news.groucho.edu
time            Any
port            tcp-conn
chat            ogin: vstout word: clouseau
```

El comando *address* da la dirección de internet (IP) del ordenador, o su nombre de dominio completo (FQDN). La entrada correspondiente en el fichero *port* sería así:

[14]También se pueden configurar algunos modems para que se reinicien a sí mismos cuando detecten una transición en DTR. Sin embargo, a algunos modems no parece gustarles esto y en ocasiones se bloquean.

```
port            tcp-conn
type            tcp
service         540
```

Esta entrada indica que hay que usar una conexión de TCP cuando una entrada en el fichero *sys* hace referencia a *tcp-conn*, y que el programa *uucico* deberá tratar de conectarse al puerto TCP 540 en el sistema remoto. Éste es el puerto por defecto del servicio UUCP. En vez del número de puerto, también se puede especificar un nombre de puerto simbólico. El numero de puerto correspondiente será buscado en el fichero */etc/services*.

12.3.10 Uso de una conexión directa

Supongamos que usted usa una línea directa que conecta su sistema **vstout** con el ordenador **tiny**. Al igual que en el caso del módem, tiene que escribir una entrada de sistema en el fichero *sys*. El comando *port* identifica el puerto serie en el que **tiny** está conectado.

```
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. La entrada *dialer* no hace falta porque no hay que marcar ningún número.

```
port            direct1
type            direct
speed           38400
```

12.4 Los síes y noes de UUCP – Ajuste de Permisos

12.4.1 Ejecución de comandos

La tarea de UUCP es copiar ficheros de un sistema a otro, y pedir la ejecución de ciertos comandos en sistemas remotos. Por supuesto, usted como administrador querrá control sobre los derechos que concede a otros sistemas – permitirles que ejecuten cualquier comando en su sistema no es una buena idea en absoluto.

Los únicos comandos que Taylor UUCP permite a otros sistemas ejecutar en su ordenador son *rmail* y *rnews*, que se usan comúnmente para intercambiar correo y noticias de Usenet sobre UUCP. El directorio en el que *uuxqt* busca es una opción que se elige al compilar el programa, pero normalmente incluye */bin*, */usr/bin*, y */usr/local/bin*. Para cambiar el conjunto de comandos para un sistema en particular, se puede usar la palabra *commands* en el fichero *sys*. Igualmente, el directorio de búsqueda se puede cambiar con el comando *command-path*. Por ejemplo, usted puede querer dar acceso al sistema **pablo** para que ejecute el comando *rsmtp* además de *mail* y *rnews*:[15]

```
system          pablo
...
commands        rmail rnews rsmtp
```

12.4.2 Transferencias de Ficheros

Taylor UUCP también le permite ajustar, en gran medida, las tranferencias de ficheros. Por un lado, usted puede desactivar las transferencias hacia y desde un sistema determinado. Simplemente necesita dar el valor *no* al comando *request*, y el sistema remoto no será capaz de obtener ficheros de su sistema ni de poner otros ficheros. De igual modo, puede prohibir que sus usuarios transfieran ficheros desde o hacia otro sistema poniendo la palabra *no* en el campo *transfer*. Por defecto, los usuarios del sistema local y el remoto pueden enviar y obtener ficheros.

Además, usted puede configurar los directorios de y a los que quiere que se puedan copiar ficheros. Usualmente se prohibe el acceso de sistemas remotos a una sola estructura de directorios, pero aun así se permite a los usuarios locales que envíen ficheros de sus directorios. Comúnmente, a los usuarios remotos se les permite que reciban ficheros sólo del directorio público de UUCP, */var/spool/uucppublic*. Este es el lugar tradicional para poner los ficheros disponibles públicamente; muy parecido a un servidor de FTP en Internet. Normalmente se refiere a este directorio con el carácter tilde.

Por lo tanto, Taylor UUCP provee cuatro comandos diferentes para configurar los directorios para enviar y recibir ficheros. Éstos son *local-send*, que especifica la lista de directorios desde los que un usuario puede pedir a UUCP que envíe ficheros; *local-receive*, que da la lista de directorios donde un usuario puede pedir que se reciban los ficheros; y *remote-send* y *remote-receive*, que hacen lo correspondiente para las peticiones que vienen de un sistema remoto. Consideremos el siguiente ejemplo:

```
system          pablo
...
local-send      /home ~
```

[15]El programa *rsmtp* se usa para manejar el correo con SMTP por lotes. Esto se explica en los capítulos sobre correo.

```
local-receive   /home ~/recibir
remote-send     ~ !~/entrada !~/recibir
remote-receive  ~/entrada
```

El comando *local-send* permite a los usuarios de su sistema que envíen cualquier fichero bajo */home* y en el directorio público de UUCP al sistema **pablo**. El comando *local-receive* les permite recibir ficheros bien en el directorio *recibir* con permiso de escritura universal en *uucppublic*, o en cualquier directorio que tenga permiso de escritura universal bajo */home*. La directiva *remote-send* permite que el sistema **pablo** obtenga ficheros de */var/spool/uucppublic*, excepto los ficheros bajo los directorios *entrada* y *recibir*. Esto se indica a *uucico* poniendo un signo de exclamación delante de los nombres de los directorios. Finalmente, la última línea permite que **pablo** ponga ficheros en *entrada*

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.

12.4.3 Reenvío

UUCP provee un mecanismo para que otros sistemas ejecuten transferencias de ficheros por usted. Por ejemplo, esto le permite que el sistema **seci** obtenga un fichero de **uchile** por usted, y lo envíe a su sistema. El siguiente comando haría esto:

```
$ uucp -r seci!uchile!~/find-ls.gz ~/uchile.files.gz
```

Esta técnica de pasar un trabajo a través de varios sistemas se llama *forwarding* (reenvío). En el ejemplo anterior, la razón para usar el reenvío puede ser que **seci** tiene acceso por UUCP a **uchile**, pero su sistema no lo tiene. Sin embargo, si usted tiene un sistema de UUCP, es deseable limitar el servicio de reenvío a unos pocos sistemas en que usted confía, para que no se le acumule una factura telefónica horrenda cuando alguien use su sistema para obtener la última versión de X11R6.

Por defecto, Taylor UUCP no permite el reenvío. Para permitirlo para un sistema en particular, puede usar el comando *forward*. Este comando especifica una lista de ordenadores desde y hacia los cuales el sistema remoto puede pedirle que reenvíe trabajos. Por ejemplo, el administrador de UUCP del sistema **seci** tendría que añadir las siguientes líneas al fichero *sys* para permitir que **pablo** obtenga ficheros de **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 sea en efecto pasado a **pablo**. De otro modo, UUCP se desharía del fichero. Esta entrada usa una variación del comando *forward* que permite que **uchile** sólo envíe ficheros a **pablo** a través de **seci**, no al revés.

Para permitir el reenvío a cualquier sistema, use el comando especial *ANY* (tiene que estar en mayúsculas).

12.5 Configuración de su sistema para ser llamado.

Si quiere configurar su sistema para que otros se conecten a éste llamándole, tiene que permitir conexiones en su puerto serie, y modificar ciertos ficheros del sistema para proveer cuentas de UUCP. Éste es el tema de esta sección.

12.5.1 Configuración de *getty*

Si quiere usar una línea serie como un puerto de entrada, tiene que activar un proceso *getty* en ese puerto. Sin embargo, algunas implementaciones de *getty* no son válidas para esto porque normalmente se desea usar un puerto para entrada y para salida. Por lo tanto tiene que asegurarse de usar un *getty* que es capaz de compartir la línea con otros programas como *uucico*, o *minicom*. Un programa que se comporta así es *uugetty* del paquete *getty_ps*. La mayoría de las distribuciones de Linux lo tienen; busque *uugetty* en el directorio */sbin*. Otro programa que existe es *mgetty*, de Gert Doering, que además hace recepción de facsímiles. Tambien puede obtener la última versión de estos programas en **sunsite.unc.edu**, tanto en binario como en código fuente.

La explicación de las diferencias de cómo *uugetty* y *mgetty* manejan la entrada al sistema está más allá del alcance de esta pequeña sección; para más información, vea el HOWTO Serial de Greg Hankins[16], así como la documentación que viene con *getty_ps* y *mgetty*.

[16] También disponible en Castellano (Serial-COMO), en LuCAS

12.5.2 Proveer Cuentas de 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**.

Para los sistemas que se conectan con el suyo a través del puerto serie, usualmente tiene que añadir estas cuentas al fichero de claves del sistema, */etc/passwd*. Es buena idea poner todos los usuarios de UUCP en un grupo especial como **uuguest**. El directorio raíz de cada cuenta de UUCP tiene que ser el directorio público */var/spool/uucppublic*; el shell de entrada tiene que ser *uucico*.

Si tiene el paquete de claves ocultadas (*shadow password*) instalado, podremos hacer esto con el comando *useradd*:

```
# useradd -d /var/spool/uucppublic -G uuguest -s /usr/lib/uucp/uucico Upablo
```

Si no utiliza la aplicación de claves ocultas, probablemente tendrá que editar */etc/passwd* a mano, añadiendo una línea como la siguiente, donde 5000 y 150 son el número de identificación de usuario (uid) y el número de grupo asignado al usuario **Upablo** y al grupo **uuguest**, respectivamente.

```
Upablo:*:5000:150:Cuenta de UUCP:/var/spool/uucppublic:/usr/lib/uucp/uucico
```

Una vez creada la cuenta, tiene que activarla asignándole una clave con el comando *passwd*.

Para servir a sistemas de UUCP que se conectan a su sistema por TCP, tiene que configurar *inetd* para que reconozca correctamente conexiones en el puerto *uucp*. Esto se consigue añadiendo la siguiente línea al fichero */etc/inetd.conf*:[17]

```
uucp   stream  tcp   nowait  root  /usr/sbin/tcpd  /usr/lib/uucp/uucico -l
```

La opción `-l` hace que *uucico* haga su propia autorización de entrada. Pedirá un nombre de usuario y una clave, igual que el programa estándar *login*, pero usará su propia base de datos de claves, en vez de */etc/passwd*. Este fichero privado de claves se llama */usr/lib/uucp/passwd* y contiene pares de nombres de entrada y claves:

[17] Normalmente, *tcpd* tiene modo 700, así que tiene que invocarlo como usuario **root**, no como usuario **uucp**, como haría normalmente.

```
Upablo  IslaNegra
Ulorca  cordoba
```

Por supuesto, este fichero tiene que pertenecer al usuario **uucp** y tener permiso 600.

Si esta base de datos parece una idea tan buena que le gustaría usarla en verificación normal de entrada (login) por serie también, se desilusionará al saber que esto no es posible por el momento de manera sencilla. Para empezar, necesita Taylor UUCP 1.05 para hacer esto, porque permite a *getty* que pase el nombre del usuario que llama al programa *uucico* usando la opción `-u`.[18] Luego tiene que engañar al programa *getty* que esté usando para que llame a *uucico* en vez del usual *login*. Con *getty_ps*, esto se puede hacer poniendo la opción *LOGIN* en el fichero de configuración. Sin embargo, esto desactiva los logins interactivos por completo. *mgetty*, por otro lado, tiene una característica atractiva que le permite invocar diferentes comandos de entrada (login) según el nombre de usuario suministrado. Por ejemplo, puede decirle a *mgetty* que use *uucico* para todos los usuarios cuyo nombre de usuario comienze con una U mayúscula, pero dejar que todos los demas usen el comando estándar *login*.

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 comandos *called-login* a cada entrada de sistema en el fichero *sys*. Esto se describe en la sección siguiente.

12.5.3 Protección contra 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.

A menos que usted sepa que puede confiar en todos los sistemas que llaman al suyo, usted *tiene* que protegerse de esta clase de impostores. La cura de esta enfermedad es requerir que cada sistema use un nombre de entrada particular, poniendo un comando *called-login* en el fichero *sys*. Un ejemplo de esto podría ser así:

```
system          pablo
... opciones usuales ...
called-login    Upablo
```

[18] La opción `-u` también existe en 1.04, pero no hace nada.

La ventaja de este método es que cuando un sistema entra y pretende ser **pablo**, el programa *uucico* comprobará que haya entrado como usuario **Upablo**. Si no es así, el sistema que nos llama será desconectado. Debería acostumbrarse a incluir el comando *called-login* en todas las entradas de sistema que añada a su fichero *sys*. Es importante que haga esto para *todos* los sistemas, independientemente de si llamarán a su sistema o no. Para aquellos sistemas que nunca le llamarán, usted puede indicar en *called-login* un mombre ficticio, como **nuncallama**.

12.5.4 Vuélvase Loco – Comprobación de Secuencia de Llamadas

Otra manera de defenderse de impostores es usando la comprobación de secuencia de llamadas. La comprobación de secuencia de llamadas le ayuda a protegerse de intrusos que de alguna manera consiguieron la clave con la que usted entra en su sistema de 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 conecta con su proveedor de correo e intenta entrar, el *uucico* remoto le rechazará, porque los números de secuencia ya no son iguales.

Si usted activa la comprobación de secuencia de llamadas, debería comprobar los ficheros históricos regularmente para buscar mensajes de error que puedan significar posibles ataques. Si su sistema rechaza el número de secuencia de llamada que el sistema remoto le ofrece, *uucico* pondrá un mensaje en el fichero histórico que dirá algo como "Out of sequence call rejected" ("Llamada fuera de secuencia rechazada"). Si su sistema es rechazado por el proveedor de correo porque los número de secuencia no están sincronizados, pondrá un mensaje en el fichero histórico que dice "Handshake failed (RBADSEQ)" ("Negociación fallida (RBADSEQ)").

Para activar la comprobación del número de secuencia, tiene que añadir el siguiente comando en la entrada de sistema:

```
# activar comprobacion de numero de secuencia
sequence        true
```

Aparte de esto, tiene que crear el fichero que contiene el número de secuencia. Taylor UUCP mantiene el número de secuencia en un fichero llamado *.Sequence* en el directorio de cola (spool) del sistema remoto. *Tiene* que pertenecer al usuario **uucp**, y debe tener permisos 600 (es decir, visible y escribible sólo por **uucp**). Lo mejor es inicializar este fichero con un valor arbitrario que ambas partes hayan acordado. De otro modo el atacante podría apañárselas para adivinar el numero probando todos los valores menores que, digamos, 60.

```
# cd /var/spool/uucp/pablo
# echo 94316 > .Sequence
# chmod 600 .Sequence
# chown uucp.uucp .Sequence
```

Por supuesto, el sistema remoto también tiene que activar la comprobación del número de secuencia, y empezar usando el mismo número que usted.

12.5.5 UUCP Anónimo

Si quiere ofrecer aceso anónimo de UUCP a su sistema, primero tiene que establecer una cuenta especial como se describió anteriormente. Es práctica común darle a esta cuenta el nombre y la clave **uucp**.

Además, tiene que especificar algunas opciones de seguridad para sistemas desconocidos. Por ejemplo, usted podría querer prohibirles que ejecuten comandos en su sistema. Sin embargo, estos parámetros no se pueden poner en una entrada del fichero *sys*, porque el comando *system* requiere el nombre del sistema, que en este caso no tenemos. Taylor UUCP resuelve este dilema con el comando *unknown*. *unknown* se puede usar en el fichero *config* para especificar cualquier comando que puede aparecer normalmente 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 limita a los sistemas desconocidos a que se bajen ficheros del directorio *pub* y que dejen ficheros en el directorio *incoming* bajo */var/spool/uucppublic*. La siguiente línea hace que *uucico* ignore cualquier petición del sistema remoto de activar la comprobación de errores (debugging) localmente. Las dos últimas líneas permiten que los sistemas desconocidos ejecuten *rmail*; pero el camino de busqueda de comandos especificado hace que *uucico* busque el comando *rmail* solamente en un directorio privado llamado *anon-bin*. Esto le permite a usted suminitrar algun programa *rmail* especial que, por ejemplo, reenvíe todo el correo al superusuario para examinarlo. Esto permite a los usuarios anónimos contactar con el administrador del sistema, pero al mismo tiempo evita que ellos manden correo a otros sistemas.

Para activar UUCP anónimo, tiene que especificar por lo menos un comando *unknown* en el fichero *config*. Si no, *uucico* rechazará cualquier sistema desconocido.

12.6 Protocolos de bajo nivel de UUCP

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 tiras 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.

12.6.1 Resumen 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 corriente o flujo (streaming) y por paquetes. La primera clase de protocolos transfiere un fichero entero, posiblemente calculando una suma de comprobación (checksum). 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 su lado, 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.

El ancho de los datos también supone una diferencia. A veces, el envío de caracteres de ocho bits sobre una conexión serie es imposible, por ejemplo si la conexión atraviesa un estúpido servidor de terminales. En este caso, los caracteres con el octavo bit igual a uno tienen que ser especialmente tratados. Cuando se envían caracteres de ocho bits sobre una conexion de siete bits, tienen que estar bajo la suposición del peor caso posible. Esto duplica la cantidad de datos a transmitir, aunque la compresión que se hace por hardware puede compensar esto. Las líneas que pueden transmitir caracteres de ocho bits se llaman preparadas para ocho bits. Éste es el caso de todas las conexiones TCP, así como la mayoría de los modems.

Existen los siguientes protocolos con Taylor UUCP 1.04:

g Éste es el protocolo más común y debería ser entendido por prácticamente todos los *uucico*'s. Hace comprobación de errores en profundidad y es, por lo tanto, apropiado para las ruidosas conexiones telefónicas. *g* require una conexión preparada para ocho bits. Es un protocolo orientado a paquetes que usa una técnica de ventana deslizante.

i Éste es un protocolo bidireccional de paquetes que puede enviar y recibir ficheros al mismo tiempo. Requiere una conexión que permita comunicación bidireccional simultánea (full-duplex) y preparada para ocho bits. Actualmente sólo es usado por Taylor UUCP.

t Éste es un protocolo diseñado para usarse sobre una conexión de TCP, u otras redes libres de errores. Usa paquetes de 1024 bytes y requiere una conexión de ocho bits.

e Este protocolo básicamente hace lo mismo que *t*. La principal diferencia es que *e* es un protocolo de flujo.

f Este protocolo está pensado para usarse sobre conexiones fiables X.25. Es un protocolo de flujo y espera una conexión de siete bits. Los caracteres de ocho bits son codificados, lo cual lo hace muy ineficiente.

G Ésta es la versión del *Unix SVR4*[19] del protocolo *g*. También se utiliza en algunas otras versiones de UUCP.

a Este protocolo es similar a ZMODEM. Requiere una conexión de ocho bits, pero codifica ciertos caracteres como XON y XOFF.

[19] N. del T.: Unix Sistema V Version 4

12.6.2 Ajuste del protocolo de transmisión

Todos los protocolos permiten alguna variación en el tamaño de los paquetes, el cronómetro y similares. Usualmente, los valores por defecto 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.[20] 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.

Taylor UUCP provee un mecanismo para satisfacer sus necesidades mediante el ajuste de estos parámetros con el comando *protocol-parameter* en el fichero *sys*. Por ejemplo, para ajustar el tamaño de los paquetes del protocolo *g* a 512 cuando se conecte con el sistema **pablo**, tiene que añadir:

```
system          pablo
...
protocol-parameter g  packet-size  512
```

Los parámetros ajustables y sus nombres varían entre protocolos. Para ver una lista completa de éstos puede consultar la documentación que acompaña al código fuente de Taylor UUCP.

12.6.3 Selección de protocolos específicos

No todas las implementaciones de *uucico* hablan y entienden cada protocolo, de modo que durante la fase de negociación de protocolos, ambos procesos tienen que ponerse de acuerdo en uno comúm. El *uucico* maestro ofrece al esclavo una lista de protocolos soportados enviando `P`*`protlist`*, de la cual el esclavo elige uno.

Según el tipo de puerto usado (módem, TCP o directo), *uucico* crea una lista por defecto de protocolos. Para módem y conexiones directas, esta lista normalmente incluye *i*, *a*, *g*, *G*, y *j*. Para conexiones TCP, la lista es *t*, *e*, *i*, *a*, *g*, *G*, *j*, y *f*. Esto se puede cambiar con el comando *protocols*, que se puede especificar en una entrada de sistema o en una de puerto. Por ejemplo, usted podría modificar la entrada del fichero *port* para su puerto de módem de esta manera:

```
port            serial1
```

[20] La mayoría de los programas incluidos en las distribuciones de Linux usan por defecto un tamaño de ventana de 7 y paquetes de 128 bytes.

```
...
protocols       igG
```

Esto requiere que cualquier conexión entrante o saliente en este puerto use *i*, *g* o *G*. Si el sistema remoto no soporta ninguno de éstos, la negociación fallará.

12.7 Solución de problemas

Esta sección describe lo que puede ir mal con su conexión de UUCP y sugiere dónde buscar el error. Sin embargo, sólo he puesto las preguntas que se me han ocurrido, por lo que pueden surgir otras muchas cosas que no diga aquí.

En cualquier caso, active la opción de encontrar errores con `-xall`, y observe el resultado en el fichero *Debug* del directorio de cola. Esto debería ayudarle rápidamente a reconocer dónde reside el problema. Siempre me ha servido de ayuda el activar el altavoz del módem cuando no se conecta. Con un módem compatible Hayes, esto se consigue añadiendo "`ATL1M1 OK`" en el diálogo del módem en el fichero *dial*.

La primera cosa a comprobar siempre debería ser si todos los permisos de los ficheros están ajustados correctamente. *uucico* debe tener identificación de usuario **uucp**, y todos los ficheros en */usr/lib/uucp*, */var/spool/uucp* y */var/spool/uucppublic* tienen que pertenecer a **uucp**. También hay algunos ficheros ocultos[21] en el directorio de cola que tienen que pertenecer a **uucp**.

***uucico* dice constantemente "Wrong time to call"**: Esto probablemente significa que en la entrada de sistema en *sys*, usted no especificó el comando *time* que determina a qué horas se puede llamar al sistema remoto, o bien especificó unas horas que en realidad prohiben llamar en este momento. Si no se especifica cuándo se puede llamar, *uucico* asume que no se puede llamar nunca a ese sistema.

***uucico* se queja de que el sistema ya está en uso**: Esto significa que *uucico* detectó un fichero cerrojo (lock) para el sistema remoto en */var/spool/uucp*. El fichero cerrojo puede pertenecer a una llamada anterior al sistema que fue interrumpida. Sin embargo, también es posible que haya otro *uucico* ejecutandose en el sistema que esté intentando llamar al sistema remoto y se atascó en una macro de diálogo, etc. Si este *uucico* no consigue conectarse al sistema remoto, mátelo con una señal de colgar (SIGHUP), y borre cualquier fichero de bloqueo que haya dejado.

Me puedo conectar al sistema remoto, pero la macro de diálogo falla: Mire el texto que recibe del sistema remoto. Si está salteado, esto puede ser un problema relacionado

[21] Es decir, ficheros cuyo nombre empieza con un punto. Estos ficheros normalmente no aparecen con un comando *ls*.

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), despues 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.

Mi módem no marca: Si su módem no indica que la línea DTR ha sido elevada cuando *uucico* hace una llamada, posiblemente no le ha dado el dispositivo correcto a *uucico*. Si su módem reconoce DTR, compruebe con un programa terminal que usted puede escribir comandos. Si esto funciona, active el eco con el comando *E* al comienzo del diálogo del módem. Si esto no produce un eco de sus comandos durante el diálogo del módem, compruebe si la velocidad de la línea es demasiado alta o demasiado baja para su módem. Si sí que ve el eco, compruebe si ha desactivado las respuestas del módem, o las ha configurado como códigos numéricos. Verifique que la macro de diálogo en sí misma es válida. Recuerde que tiene que poner dos barras invertidas para enviar una al módem.

Mi módem intenta marcar, pero la llamada no sale: Inserte un retraso en el número de teléfono. Esto es especialmente útil cuando se llama fuera de la red interna de una compañía. Para la gente en Europa, que normalmente marca con pulsos (pulse-tone), pruebe con tonos (touch-tone). En ciertos paises, los servicios telefónicos han actualizado sus redes recientemente. "touch-tone" ayuda a veces.

El fichero de registro (log) dice que tengo un ratio de paquetes perdidos extremadamente alto: Esto parece un problema de velocidad. Puede ser que la conexión entre su ordenador y su módem sea demasiado lenta (recuerde adaptarla a la mayor velocidad efectiva posible). O puede ser que su hardware sea demasiado lento para servir las interrupciones a tiempo. Con un chip NSC 16550A en su puerto serie, 38kbps puede funcionar razonablemente bien; sin embargo, sin FIFOs (como el chip 16450), el límite es 9600. También tiene que asegurarse de que la negociación hardware está incluida en la línea serie.

Otra posible causa es que la negociación hardware no esté activada en el puerto. Taylor UUCP 1.04 no tiene mecanismos para activar la negociación de RTS/CTS. Tiene que activarla explícitamente en el fichero *rc.serial* usando el siguiente comando:

```
$ stty crtscts < /dev/cua3
```

Puedo entrar en el otro sistema, pero la negociación falla: Bien, puede ser debido a muchas causas. Los mensajes en el fichero de registro deberían decirle un montón de cosas. Mire qué protocolos ofrece el sistema remoto (envía un `P`*`protlist`* durante la negociación). A lo mejor no tienen nada en comúm (¿selecionó algún protocolo en *sys* o *port*?).

Si el sistema remoto envía `RLCK`, hay un fichero de bloqueo (lock) para su sistema en el sistema remoto. Si no es porque usted ya está conectado al sistema remoto en otra línea, pida que lo borren.

Si envía `RBADSEQ`, el otro sistema tiene la comprobación de la cuenta de conversación activada para su sistema, pero los números no se corresponden. Si envía `RLOGIN`, no le fue permitido entrar con ese nombre de usuario.

12.8 Archivos de registro histórico (Log Files)

Cuando se compila el paquete de UUCP para usar ficheros de registro al estilo Taylor-UUCP, se tendrán tres ficheros históricos globales, y todos residirán en el directorio de cola. El fichero principal se llama *Log* y contiene toda la información sobre las conexiones establecidas y los ficheros transferidos. Un extracto típico podría ser como el siguiente (despues de formatearlo para que quepa 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 importante es *Stats*, que lista las estadísticas de transferencias de ficheros. La sección de *Stats* que corresponde a la transferencia anterior se muestra aquí:

```
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)
```

Como en el caso anterior, las líneas han sido partidas para que quepan en la página.

El tercer fichero es *Debug*. Éste es el sitio donde se incluye toda la información para buscar errores. Si usted usa detección de errores (debugging), tiene que asegurarse de que este fichero tenga modo de protección de 600. Dependiendo del modo de búsqueda de errores que haya elegido, este fichero puede incluir el nombre de usuario y la clave que usted usa para conectarse al sistema remoto.

Algunos programas de UUCP que incluyen algunas distribuciones de Linux han sido compilados para usar el estilo de fichero de registro histórico de HDB. HDB UUCP usa muchos ficheros de registro archivados bajo */var/spool/uucp/.Log*. Este directorio contiene tres directorios más, llamados *uucico*, *uuxqt* y *uux*. Éstos contienen el resultado de información histórica generada por cada uno de los comandos correspondientes, ordenada en diferentes ficheros para cada sistema. Por lo tanto, la salida del programa *uucico* cuando se llama a **pablo** acabará en *.Log/uucico/pablo*, mientras que el *uuxqt* correspondiente escribirá en *.Log/uuxqt/pablo*. Las líneas escritas a cada uno de estos ficheros son sin embargo iguales que en Taylor UUCP.

Cuando active la opción de búsqueda de errores con el estilo HDB, la información será escrita en el directorio *.Admin* bajo */var/spool/uucp*. Durante llamadas salientes, la información se envía al fichero *.Admin/audit.local*, mientras que la salida de *uucico* cuando alguien nos llama se graba en *.Admin/audit*.

Capítulo 13

Correo Electrónico

Uno de los usos más comunes de las redes informáticas desde sus orígenes ha sido el correo electrónico. 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 *e-mail* (correo electrónico), 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 nodos conectados a la Internet cumplen uno recogido en el RFC 822, complementado en algunos RFCs que describen un método independiente de la máquina para transferir caracteres especiales, y similares. Mucho se ha discutido recientemente sobre el "correo multi-media", que tiene que ver con incluir imágenes y sonido en los mensajes de correo. Otro estándar, X.400, ha sido definido por el CCITT.

Hay ya una gran cantidad de programas de transporte de correo para sistemas UN⋆X. Uno de los más conocidos es el *sendmail*, de la Universidad de Berkeley, que se usa en diversas plataformas. El autor original fue Eric Allman, que esta trabajando activamente en el equipo *sendmail* de nuevo. Hay dos adaptaciones para Linux del *sendmail-5.56c* disponibles, una de las cuales se describirá en el capítulo 15. La versión de *sendmail* actualmente en desarrollo es la 8.6.5.

El agente de correo de uso más común en Linux es el *smail-3.1.28*, escrito por Curt Landon Noll y Ronald S. Karr y con Copyright de los mismos autores. Este es el que se incluye en la mayoría de las distribuciones de Linux. En lo sucesivo nos referiremos a él simplemente como *smail*, aunque hay otras versiones del mismo programa que son totalmente diferentes, y que no describiremos aquí.

Comparado con *sendmail*, *smail* es bastante joven aún. Si se ocupan del correo de un nodo pequeño sin necesidades de direccionamiento complicadas, sus capacidades son muy parecidas. Para nodos grandes, sin embargo, *sendmail* siempre gana, porque su método de configuración es mucho más flexible.

Ambos *smail* y *sendmail* 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 del ordenador local), hay muchos más parámetros que pueden ajustarse. El fichero principal de configuración de *sendmail* es muy difícil de entender a la primera. Parece como si el gato se hubiese echado una siesta sobre el teclado con la tecla de mayúsculas pulsada. Los ficheros de configuración de *smail* están más estructurados y son más fáciles de entender que los del *sendmail*, pero no dan al usuario tanto poder a la hora de ajustar el comportamiento del gestor de correo. De todos modos, para nodos pequeños de UUCP o Internet, el trabajo que se necesita para poner a punto cualquiera de ellos es prácticamente el mismo.

En este capítulo trataremos sobre qué es el 'e-mail' y qué temas tendrá que abordar usted como administrador del sistema. Los capítulos 14 y 15 darán instrucciones para poner a punto *smail* y *sendmail* por primera vez. La información que se suministra debe bastar para poner en marcha pequeños nodos, 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 UN⋆X, incluyendo Linux.

Para más información sobre temas específicos de correo electrónico sobre Linux, por favor, consulte el 'Electronic Mail HOWTO' de Vince Skahan, que aparece en **comp.os.linux.announce** con regularidad. Las distribuciones fuente de *elm*, *smail* 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.

13.1 ¿Qué es un mensaje de correo?

Un mensaje de correo consta de un contenido (*body*), 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 es transmitido.

La segunda variedad 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 *encabezado del mensaje (mail header)*. Se separa del *contenido del mensaje (mail body)* por una línea en blanco. [1]

La mayoría del software para transporte de correo que se usa en el mundo UN⋆X usa un formato de encabezado 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 fácilmente adaptado a otras redes, incluyendo muchas basadas en UUCP.

Pero RFC 822 es sólo el máximo denominador común. 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 extensiones de correo multimedia (multi-media mail extension, MIME).

En todos esos estándares, el encabezado 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 (:) o un espacio. El formato y la semántica de cada campo varía dependiendo del nombre del mismo. Un campo del encabezado se puede continuar más allá de una línea, si la línea siguiente comienza con un carácter de tabulación. Los campos pueden aparecer en cualquier orden.

Un encabezado de correo típico puede ser algo así:

```
From brewhq.swb.de!ora.com!andyo Wed Apr 13 00:17:03 1994
Return-Path: <brewhq.swb.de!ora.com!andyo>
Received: from brewhq.swb.de by monad.swb.de with uucp
        (Smail3.1.28.1 #6) id m0pqqlT-00023aB; Wed, 13 Apr 94 00:17 MET DST
Received: from ora.com (ruby.ora.com) by brewhq.swb.de with smtp
        (Smail3.1.28.1 #28.6) id <m0pqoQr-0008qhC>; Tue, 12 Apr 94 21:47 MEST
Received: by ruby.ora.com (8.6.8/8.6.4) id RAA26438; Tue, 12 Apr 94 15:56 -0400
Date: Tue, 12 Apr 1994 15:56:49 -0400
Message-Id: <199404121956.PAA07787@ruby>
From: andyo@ora.com (Andy Oram)
To: okir@monad.swb.de
Subject: Re: Tu parte de RPC
```

[1] Se suele añadir una *firma (signature)* o *.sig* a un mensaje, usualmente conteniendo información sobre el autor, junto con un chiste o cita célebre. Se separa del resto del mensaje con una línea conteniendo "`-- `".

Usualmente, todos los campos del encabezado necesarios son generados por la interface con el servidor 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. Una lista de campos de encabezado comunes y su significado se da a continuación:

`From:` Contiene la dirección de correo electrónico del remitente, y posiblemente el "nombre real". Un verdadero zoológico de formatos distintos se usa aquí.

`To:` Esta es la dirección de e-mail del destinatario.

`Subject:` Describe el contenido del mensaje en pocas palabras. Al menos eso es lo que *debiera* hacer.

`Date:` La fecha en la que el mensaje fue enviado.

`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 solo en aquélla 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 es opcional.

`Message-ID:` Una cadena generada por el transporte de correo en el sistema remitente. Es única para cada mensaje.

`Received:` Cada nodo que procesa su correo (incluyendo las máquinas del remitente y el destinatario) insertan este campo en el encabezado, dando el nombre del nodo, una identificación de mensaje, hora y fecha a la que lo recibieron, de qué nodo 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 aun no han sido incluidas en un RFC, o que no lo serán nunca. Esto se usa, por ejemplo, en la lista de correo de los Activistas de Linux, donde el canal a usar se selecciona con el campo de encabezado `X-Mn-Key:`.

La única excepción a esta estructura es la primera línea. Comienza con la palabra clave `From` seguida de un espacio en blanco, en vez de dos puntos. Para distinguirlo del campo ordinario `From:` se suele denotar como `From_`. Contiene la ruta que ha seguido el mensaje, escrita al estilo ruta bang de UUCP (explicado más adelante), la hora y la fecha en que fue recibido por la última máquina que lo ha procesado, y una parte opcional que especifica desde qué máquina ha sido recibido. Como este campo es regenerado por cada sistema que procesa el mensaje, alguna veces queda incluido en los datos del sobre.

El campo `From_` continúa existiendo por compatibilidad con procesadores de correo antiguos, pero no se usa demasiado en la actualidad excepto por algunos interfaces de usuario de correo que se basan en él para marcar el comienzo de un mensaje en el buzón del usuario. Para evitar problemas potenciales con líneas del contenido del mensaje que comiencen también con “From ”, se ha convertido en práctica común distinguir este último caso precediéndolo de un “>”.

13.2 ¿Cómo se reparte el correo?

Generalmente, usted escribirá su correo usando un interface de correo como *mail* o *mailx*; u otros más sofisticados como *elm*, *mush*, 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 interface en la mayoría de los casos se lo pasará a otro programa para que lo transmita. Este programa se denomina el *agente de transporte de correo (mail transport agent)*, o MTA. En algunos sistemas hay agentes de transporte de correo distintos para envíos locales o lejanos; en otros hay sólo un MTA. El comando para envíos lejanos se denomina usualmente *rmail*, el otro se denomina *lmail* (si existe).

Un envío local de correo es, por supuesto, algo más que añadir el mensaje al buzón del destinatario. Usualmente el MTA local entenderá cómo usar alias (definir direcciones locales de destinatarios que dirigen a otras direcciones) y cómo usar redirecciones[2], es decir, dirigir el correo de un usuario a otra dirección). También, 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 usara normalmente SMTP. SMTP son las siglas de *Simple Mail Transfer Protocol*, o Protocolo Simple de Trasferencia de Correo que se define en el RFC 788 y RFC 821. SMTP usualmente conecta con la máquina del destinatario directamente, negociando la transferencia del mensaje con el demonio SMTP del otro lado.

[2] N. del T.: Del inglés *forwarding*

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.

Dado que esto se hace para cada mensaje por separado, puede producir una carga considerable de trabajo en un nodo procesador de correo grande, además de inundar las colas UUCP con cientos de pequeños mensajes que ocupan una cantidad de disco desproporcionada. [3] Por esto algunos MTAs permiten recopilar varios mensajes de un sistema remoto en un solo lote. El fichero de lotes contiene los comandos SMTP que el nodo local ejecutaría normalmente si usara una conexión SMTP directa. A esto se le llama BSMTP, o *batched* SMTP (SMTP por lotes). El lote es suministrado al programa *rsmtp* o *bsmtp* en el sistema remoto, que procesará la entrada como si una conexión SMTP normal hubiese ocurrido.

13.3 Direcciones de correo electrónico

Para el correo electrónico, una dirección consiste en, al menos, el nombre de la máquina que maneja el correo del destinatario, y una identificación de usuario reconocida por ese sistema. Puede ser el nombre de acceso del destinatario, pero puede ser también cualquier otra cosa. Otros esquemas de direcciones, como el X.400, usan un conjunto más general de "atributos" que se utilizan para buscar la máquina del destinatario en un servidor de directorio X.500.

La forma en que se interpreta un nombre de máquina, es decir, a qué nodo va a llegar finalmente nuestro mensaje, y cómo combinar este nombre con el nombre de usuario del destinatario depende enormemente de la red en la que nos encontremos.

Los nodos en la Internet siguen el estándar RFC 822, que requiere una notación **usuario@máquina.dominio**, donde **máquina.dominio** es el nombre de dominio totalmente cualificado (Fully Qualified Domain Name, o FQDN) de la máquina. El signo de *arroba* que aparece entre medias se suele denominar signo "at"[4]. Dado que esta notación no indica la ruta hasta la máquina de destino, sino que da el nombre (único) de dicha máquina, a esto se le suele llamar una dirección *absoluta*.

En el entorno UUCP original, la forma predominante era **ruta!máquina!usuario**, donde **ruta** describía una secuencia de máquinas a través de las cuales debía viajar el mensaje para llegar a **máquina**, su destino final. Esta notación se llama la *ruta bang*, porque un signo de exclamación se denomina coloquialmente "bang". Hoy en día muchas redes basadas

[3]Esto es así porque el espacio de disco se asigna usualmente en bloques de 1024 Bytes. Incluso un mensaje de como mucho 400 Bytes ocupará 1 Kb completo.

[4]N. del T. de la preposición inglesa "at", que significa "en"

en UUCP han adoptado el RFC 822, y entenderán ese tipo de dirección.

Estos dos tipos de direcciones no se mezclan muy bien. Supongamos una dirección **máquinaA!usuario@máquinaB**. No queda claro si el signo '**@**' tiene precedencia sobre la ruta o viceversa: ¿Hemos de enviar el mensaje a **máquinaB**, que lo enviará a **máquinaA!usuario**, o debe ser enviado **máquinaA**, que lo redirigirá a **usuario@máquinaB**?

Las direcciones que mezclan diferentes tipos de operadores de dirección se denominan *direcciones híbridas*. El más notorio es el ejemplo anterior. Se resuelve usualmente dándole precedencia al signo '**@**' sobre la ruta. En el ejemplo anterior, esto significa enviar el mensaje a **máquinaB** primero.

De todos modos, hay una forma de especificar rutas acorde con RFC 822: <**@máquinaA,@máquinaB:usuario@máquinaC**> denota la dirección de **usuario** en **máquinaC**, indicando que se debe llegar a **máquinaC** a través de **máquinaA** y **máquinaB** (en ese orden). Este tipo de dirección se suele llamar una *dirección route-addr* (de route, ruta y address, dirección).

Y también existe el operador de dirección '**%**': **usuario%máquinaB@máquinaA** será enviado primero a **máquinaA**, que sustituirá el signo de tanto por ciento que se encuentre más a la derecha en la expresión (en este caso el único) por un signo '**@**'. La dirección quedará ahora **usuario@máquinaB**, y el gestor de correo redirigirá alegremente el mensaje a la **máquinaB** que lo entregará a **usuario**. Este tipo de dirección se suele denominar a veces como "Ye Olde ARPANET Kludge" ("La Vieja Chapuza de ARPANET") y su uso está desaconsejado. Aun así muchos agentes de transporte de correo generan este tipo de direcciones.

Otras redes tienen más formas de expresar direcciones. Las redes basadas en el protocolo DECnet, por ejemplo, usan dos signos de dos puntos como separador, dando lugar a direcciones como *`máquina::usuario`*.[5] Finalmente, el estándar X.400 usa un esquema totalmente distinto, describiendo a un destinatario por un conjunto de pares atributo-valor, como país u organización.

En FidoNet, cada usuario se identifica por un código como **2:320/204.9**, que consiste en cuatro números que denotan la zona (2 es Europa), red (320 es París y Banlieve), nodo (el repetidor/BBS local), y punto (el PC del usuario). Las direcciones Fidonet se pueden traducir a RFC 822: la anterior se escribiría **Thomas.Quinot@p9.f204.n320.z2.fidonet.org**. ¿No he dicho antes que los nombres de dominio son fáciles de recordar?

Hay algunas implicaciones al usar esos tipos diferentes de direcciones que serán descritas a lo largo de las próximas secciones. De todos modos, en un entorno RFC 822 raramente

[5]Cuando se intente llegar a una dirección DECnet desde un entorno RFC 822 se puede usar `máquina::usuario.@relevo`, donde *`relevo`* es el nombre de una pasarela Internet-DECnet conocida.

se usará otra cosa que direcciones absolutas como `usuario@máquina.dominio`.

13.4 ¿Cómo funciona el encaminado del correo?

El proceso de dirigir un mensaje a la máquina del destinatario se denomina *encaminado*. Además de encontrar una ruta desde el nodo emisor al receptor, este proceso incluye también chequeo de errores así como optimización de velocidad y coste.

Hay mucha diferencia en la forma en la que un nodo UUCP maneja el encaminado y la forma en que lo hace un nodo Internet. En la Internet, el trabajo principal de dirigir datos al nodo del destinatario (una vez que se conoce por su dirección IP) se realiza por la capa IP de control de red, mientras que en UUCP, la ruta debe ser suministrada por el usuario, o generada por el agente de transporte de correo.

13.4.1 Encaminado de correo en la Internet

En la Internet, depende enteramente del nodo de destino el que se realice algún encaminado específico de correo. El comportamiento por defecto consiste en enviar el mensaje al nodo de destino buscando su dirección IP, y dejando el encaminado de los datos en sí a la capa IP de transporte.

Generalmente, la mayoría de los nodos querrán que todo el correo entrante se dirija a un servidor de correo fácilmente accesible que sea capaz de procesar todo ese tráfico, y que distribuirá ese correo localmente. Para anunciar ese servicio, el nodo publica el llamado campo MX para su dominio local en la base de datos DNS. MX significa *Mail Exchanger (Intercambiador de correo)* y básicamente quiere decir que el servidor va a actuar como un redistribuidor de correo para todas las máquinas de este dominio. Los campos MX también pueden usarse para manipular el tráfico dirigido a máquinas que no están ellas mismas conectadas a la Internet, como redes UUCP o redes corporativas que contienen información confidencial.

Los campos MX también tienen una *preferencia* asociada. Es un entero positivo. Si existen varios intercambiadores de correo para una máquina, el agente de transporte de correo intentará enviar el mensaje al intercambiador con menor valor de preferencia, y sólo si éste falla probará uno con mayor valor. Si el nodo local es él mismo un intercambiador de correo para la dirección de destino, no redirigirá los mensajes a cualquier máquina MX que tenga un valor de preferencia mayor que el suyo propio: ésta es una forma segura de evitar bucles de correo.

Supongamos que una organización, digamos la Sociedad ACME, quiere que todo su correo sea manipulado por su máquina llamada **mailhub**. Entonces tendrán un campo MX como el siguiente en su base de datos DNS:

```
acme.com        IN   MX      5    mailhub.acme.com
```

Esto anuncia que **mailhub.acme.com** es un intercambiador de correo para **acme.com** con un valor de preferencia de 5. Una máquina que desee enviar un mensaje a **joe@greenhouse.acme.com** buscará el registro DNS de **acme.com**, y encontrará el campo MX apuntando hacia **mailhub**. Si no hay ningún MX con un valor de preferencia menor que 5, el mensaje será enviado a **mailhub**, que lo entregará a **greenhouse**.

Lo anterior es sólo un esbozo de cómo funcionan los campos MX. Para más información sobre encaminado de correo en la Internet, por favor consulte el RFC 974.

13.4.2 Encaminado de correo en el mundo UUCP

El encaminado de correo en redes UUCP es mucho más complicado que en la Internet, porque el software de transporte no realiza ningún encaminado. Al principio, todo el correo tenía que ser dirigido usando rutas bang. Las rutas bang especifican, como hemos dicho ya, una lista de nodos a través de los cuales enviar el mensaje, separados por signos de admiración, y seguida del nombre del usuario. Para dirigir una carta a Juanita, usuaria de una máquina llamada **moria**, deberíamos usar la ruta **eel!swim!moria!juanita**. Esto enviaría el correo desde nuestro nodo a **eek**, desde allí a **swim** y finalmente a **moria**.

El inconveniente obvio de esta técnica es que obliga a que recordemos mucho sobre la topología de la red, enlaces rápidos, etc. Aun peor, cualquier cambio en la topología de la red — como enlaces que se eliminan o nodos que se quitan — puede causar que el mensaje no llegue al destino simplemente porque no se estaba al tanto del cambio. Y finalmente en caso de que nos mudemos a otro lugar, deberemos actualizar todas esas rutas.

Sin embargo, un hecho que hizo que fuese necesario el uso del encaminado manual fue la presencia de nombres de nodo ambiguos: por ejemplo, supongamos que hay dos máquinas llamadas **moria**, una en los EE.UU., y otra en Francia. ¿A cual de ellas nos referimos con **moria!juanita**? Esto puede quedar claro especificando que ruta seguir para llegar a **moria**.

El primer paso para eliminar ambigüedades en nombres de nodos fue la fundación del *Proyecto de Cartografía UUCP (The UUCP Mapping Project)*. Se encuentra en la Universidad de Rutgers, y lleva el registro de todos los nombres de nodos oficiales UUCP, junto con información de sus vecinos UUCP y su localización geográfica, asegurándose de que ningún nombre se repite. La información recogida por el Proyecto de Cartografía se publica como los *Mapas de Usenet*, que se distribuyen regularmente en Usenet.[6] Una entrada típica para un sistema en un mapa (después de eliminar los comentarios) sería algo así:

```
moria
        bert(DAILY/2),
        swim(WEEKLY)
```

Esta entrada dice que **moria** tiene un enlace con **bert**, al que llama dos veces al día (DAILY=diariamente), y con **swim**, al que llama una vez a la semana (WEEKLY). Volveremos de nuevo al formato del fichero mapa más adelante.

Usando la información sobre conectividad proporcionada en los mapas, se pueden generar automáticamente las rutas completas de nuestra máquina a cualquier nodo de destino. Esta información se almacena usualmente en el fichero *paths*, también llamado *base de datos de alias de rutas (pathalias database)*. Suponiendo que los mapas indican que se puede llegar a **bert** a través de **ernie**, entonces una entrada de alias de ruta para **moria** generada a partir del fragmento de mapa anterior quedaría tal que así:

```
moria           ernie!bert!moria!%s
```

Si ahora damos una dirección de destino de **juanita@moria.uucp**, nuestro MTA escogerá la ruta mostrada anteriormente, y enviará el mensaje a **ernie** con una dirección en el sobre de **bert!moria!juanita**.

De todos modos, construir un fichero *paths* a partir de los mapas completos de Usenet no es una buena idea. La información proporcionada en ellos normalmente está bastante distorsionada, y a veces es obsoleta. Así, sólo unos pocos nodos grandes usan los mapas mundiales completos de UUCP para construir su fichero *paths*. La mayoría de los nodos sólo mantienen información de encaminado hacia los nodos en sus cercanías, y envían cualquier correo dirigido a nodos que no encuentran en sus bases de datos a un nodo más inteligente, con una información de encaminado más completa. Este esquema se denomina *encaminado por nodo inteligente (smart-host routing)*. Las máquinas que tienen un solo enlace de correo UUCP (llamadas *nodos hoja*) no realizan ningún encaminado propio; confían enteramente en su nodo inteligente para esta tarea.

13.4.3 Mezcla de UUCP y RFC 822

La mejor cura contra los problemas vistos con el encaminado en redes UUCP es la adopción del sistema de nombres de dominio en ellas. Por supuesto, no se pueden hacer peticiones

[6] Los mapas de nodos registrados en el Proyecto de Cartografía UUCP se distribuyen en el grupo de noticias **comp.mail.maps**; otras organizaciones pueden publicar mapas separados para su red.

a un servidor de nombres usando UUCP. Aun así, muchos nodos UUCP han formado pequeños dominios que coordinan su encaminado internamente. En los mapas, estos dominios publican uno o dos nodos como sus pasarelas de correo, de modo que no tenga que haber una entrada en el mapa para cada nodo en el dominio. Las pasarelas pueden distribuir todo el correo que fluye hacia dentro y hacia fuera del dominio. El esquema de encaminado dentro del dominio es completamente invisible para el resto del mundo.

Esto funciona muy bien con el esquema de encaminado por nodo inteligente descrito anteriormente. La información de encaminado global se encuentra sólo en las pasarelas; a los nodos menores dentro de un dominio les basta con un pequeño fichero *paths* escrito a mano que contiene la lista de las rutas dentro de su dominio, y la ruta al concentrador de correo. Incluso las pasarelas de correo no tienen por qué incluir información de encaminado hacia cada una de las máquinas UUCP en el mundo. Aparte de la información completa de encaminado para el dominio al que sirven, ahora sólo necesitan tener en sus bases de datos rutas a dominios completos. Por ejemplo, la siguiente entrada alias de ruta encaminará todo el correo para nodos del dominio **sub.org** a **smurf**:

```
.sub.org        swim!smurf!%s
```

Cualquier correo dirigido a **claire@jones.sub.org** será enviado a **swim** con una dirección en el sobre tal como **smurf!jones!claire**.

La organización jerárquica del espacio de nombres de dominio permite a los servidores de correo mezclar rutas más específicas con otras menos específicas. Por ejemplo, un sistema en Francia puede tener rutas específicas hacia subdominios de **fr**, pero encaminar cualquier correo dirigido a máquinas del dominio **us** hacia algún sistema en los EE.UU. De esta forma, el encaminado basado en dominios (que es como se denomina esta técnica) reduce enormemente el tamaño de las bases de datos de encaminado, así como las tareas de administración requeridas.

El mayor beneficio de usar nombres de dominio en un entorno UUCP, de todos modos, es que al cumplir con RFC 822 se pueden fácilmente usar pasarelas entre redes UUCP y la Internet. Hoy en día, muchos dominios UUCP tienen un enlace con una pasarela a Internet que actúa como su nodo inteligente. Enviar mensajes a través de la Internet es más rápido, y la información de encaminado es mucho más fiable porque los nodos en la Internet pueden usar DNS en vez de mapas Usenet.

Para poder ser alcanzables desde la Internet, los dominios basados en UUCP usualmente hacen que su pasarela a Internet anuncie una entrada MX para ellos (las entradas MX que fueron descritas anteriormente). Por ejemplo, supongamos que **moria** pertenece al dominio **orcnet.org** y que **gcc2.groucho.edu** actúa como su pasarela a Internet. Entonces **moria** usaría **gcc2** como su nodo inteligente, de modo que todo el correo hacia dominios remotos sería enviado a través de la Internet. Por otro lado, **gcc2** anunciaría una entrada MX para **orcnet.org**, y enviaría todo el correo entrante para nodos de **orcnet** a **moria**.

El único problema que queda es que los programas de transporte UUCP no pueden manejar nombres de dominio totalmente cualificados. La mayoría de los paquetes integrados de UUCP fueron diseñados para tratar con nombres de nodos de ocho caracteres como máximo, algunos incluso menos, y usar caracteres no alfanuméricos, como puntos, está totalmente fuera de lugar para la mayoría.

Así, es imprescindible disponer de alguna forma de relacionar nombres RFC 822 y UUCP. La forma en que esto se hace es totalmente dependiente de la implementación. Una manera común de relacionar FQDNs con nombres UUCP es usar el fichero de alias de ruta para esto:

```
moria.orcnet.org  ernie!bert!moria!%s
```

Esto producirá una ruta bang al estilo UUCP puro a partir de una dirección que especifica un nombre de dominio totalmente cualificado. Algunos programas de correo suministran un fichero especial para esto; *sendmail*, por ejemplo, usa el fichero *uucpxtable*.

La transformación inversa (llamada coloquialmente *dominizar*) se necesita a veces cuando se envía correo desde una red UUCP a la Internet. Mientras el remitente use el nombre de dominio totalmente cualificado en la dirección de destino, este problema se puede evitar no eliminando el nombre del dominio de la dirección del sobre cuando se redirige el mensaje al nodo inteligente. De todos modos, siguen existiendo algunos nodos UUCP que no son parte de ningún dominio. Éstos se *dominizan* usualmente añadiendo el pseudo-dominio **uucp**.

13.5 Formatos de Fichero Mapa y Alias de Ruta

La base de datos de alias de ruta ofrece la información de encaminado principal en redes basadas en UUCP. Una entrada típica será como sigue (nombre del nodo y ruta separadas por tabuladores):

```
moria.orcnet.org  ernie!bert!moria!%s
moria             ernie!bert!moria!%s
```

Esto hace que cualquier mensaje a **moria** sea enviado vía **ernie** y **bert**. Ambos nombres de **moria** (su nombre totalmente cualificado y su nombre UUCP) deben ser suministrados si el programa no tiene una forma separada de relacionar ambos.

Si se quieren dirigir los mensajes destinados a máquinas dentro de algún dominio a su concentrador de correo, se debe también especificar un camino en la base de datos de alias de ruta, dando el nombre del dominio como objetivo, precedido de un punto. Por ejemplo, si todas las máquinas de **sub.org** pueden ser alcanzadas a través de **swim!smurf**, la entrada de alias de ruta debe ser:

```
.sub.org        swim!smurf!%s
```

Escribir un fichero de alias de ruta es aceptable sólo cuando se está manejando un nodo que no necesita hacer demasiados encaminados. Si se tienen que hacer encaminados hacia un gran número de máquinas, una manera mejor es usar el comando *pathalias* para crear el fichero a partir de los ficheros de mapa. Los mapas se pueden mantener con más facilidad, porque sólo hay que añadir o quitar un sistema editando la entrada de dicho sistema en el mapa, y volver a crear el fichero de mapa. Aunque los mapas publicados por el Proyecto de Cartografía de Usenet ya no se usan demasiado para encaminar, las redes UUCP pequeñas pueden suministrar información de encaminado en su propio conjunto de mapas.

Un fichero de mapa consiste principalmente en una lista de nodos, junto con los nodos que cada sistema registra o lo registran. El nombre del sistema comienza en la columna uno, y sigue una lista de enlaces separados por comas. La lista se puede continuar en varias líneas, comenzando cada nueva línea con un tabulador. Cada enlace consiste en el nombre del nodo enlazado, seguido del coste, entre corchetes. El coste es una expresión aritmética, consistente en números y costes simbólicos. Las lineas que comienzan con un signo hash (#) se ignoran.

Como ejemplo, consideremos **moria**, que registra a **swim.twobirds.com** dos veces al día, y a **bert.sesame.com** una vez a la semana. Además el enlace con **bert** usa sólo un módem lento de 2400bps. **moria** publicaría la siguiente entrada en los mapas:

```
moria.orcnet.org
        swim.twobirds.com(DAILY/2),
        bert.sesame.com(WEEKLY+LOW)

moria.orcnet.org = moria
```

La última línea lo haría ser conocido bajo su nombre UUCP también. Obsérvese que debe ser *DAILY/2*, porque llamar dos veces al día reduce a la mitad el coste de ese enlace.

Usando la información de dichos ficheros de mapa, *pathalias* puede calcular rutas óptimas para cualquier destino que aparezca en el fichero de rutas, y producir una base de datos de alias de ruta a partir de éste, que se puede usar para encaminar hacia esos nodos.

pathalias proporciona varias posibilidades más, como esconder nodos (es decir, hacerlos accesibles sólo a través de una pasarela), etc. Véase la página del manual sobre *pathalias* para más detalles, además de una lista completa de costes de enlace.

Los comentarios en el fichero de mapa contienen generalmente información adicional sobre los nodos descritos en él. Existe un formato rígido para especificar ésta, de manera que pueda ser recuperada de los mapas. Por ejemplo, un programa llamado *uuwho* usa una base de datos creada a partir de los ficheros de mapa para presentar esta información de una manera elegante.

Cuando se registra un nodo con una organización que distribuye ficheros de mapa a sus miembros, generalmente se debe rellenar una de esas entradas de mapa.

A continuación hay una entrada de mapa de ejemplo (de hecho, es la del nodo del autor):

```
#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 detrás de los primeros dos caracteres es un tabulador. El significado de la mayoría de los campos es bastante obvio; recibiremos una descripción detallada del dominio con el que vayamos a registrarnos. El campo L es el más divertido de averiguar: da nuestra posición geográfica en latitud/longitud y se usa para dibujar los mapas *postscript* que muestran todos los nodos de cada país, así como los del mundo entero. [7]

13.6 Configuración de *elm*

elm significa "electronic mail" (correo electrónico) y es una de las utilidades UN⋆X más razonablemente bautizadas. Proporciona una interface de pantalla completa con una buena

[7] Aparecen regularmente en **news.lists.ps-maps**. Cuidado: son ***enormes***.

utilidad de ayuda. No vamos a explicar aquí como se usa *elm*, sólo nos detendremos en sus opciones de configuración.

Teóricamente, se puede usar *elm* sin configurar, y todo funciona bien — con suerte. Pero hay algunas opciones que deben definirse, aunque sólo se necesitan en contadas ocasiones.

Cuando comienza, *elm* lee un conjunto de variables de configuración desde el fichero *elm.rc* en */usr/lib/elm*. Entonces, intentará leer el fichero *.elm/elmrc* en nuestro directorio personal. Normalmente este fichero no se genera a mano. Se crea cuando se escoge "*save options*" (grabar opciones) desde el menú de opciones de *elm*.

El conjunto de opciones para el fichero privado *elmrc* también está disponible en el fichero global *elm.rc*. La mayoría de las definiciones en el fichero privado *elmrc* sustituirán a las del fichero global.

13.6.1 Opciones Globales de *elm*

En el fichero global *elm.rc*, se deben definir las opciones que pertenecen a nuestro nombre de máquina. Por ejemplo, en la Cervecera Virtual, el fichero para **vlager** contendrá lo siguiente:

```
#
# El nombre del nodo local
hostname = vlager
#
# Nombre del dominio
hostdomain = .vbrew.com
#
# Nombre de dominio totalmente cualificado (FQDN)
hostfullname = vlager.vbrew.com
```

Estas opciones definen la idea que tiene *elm* sobre el nombre de la máquina local. Aunque esta información se usa raramente, debemos definir estas opciones. Obsérvese que estas opciones sólo tienen efecto cuando se dan en el fichero global de configuración; cuando se encuentran en nuestro *elmrc* privado, serán ignoradas.

13.6.2 Conjuntos de Caracteres Nacionales

Recientemente, han habido propuestas para corregir el estándar RFC 822 para soportar varios tipos de mensajes, como texto simple, datos binarios, ficheros Postscript, etc. El conjunto de estándares y RFCs que cubren estos aspectos se suelen conocer como MIME, o Extensiones de Correo Internet Multipropósito (Multipurpose Internet Mail Extensions). Entre otras cosas, esto permite al destinatario saber si un conjunto de caracteres distinto del estándar ASCII ha sido usado al escribir el mensaje, por ejemplo usando los acentos o diéresis del castellano. Esto tiene soporte en *elm* hasta cierto punto.

El conjunto de caracteres usado por Linux internamente para representar caracteres se suele denominar ISO-8859-1, que es el nombre del estándar que cumple. También se conoce como Latin-1. Cualquier mensaje que use caracteres de ese conjunto debe llevar la siguiente línea en su encabezado:

```
Content-Type:  text/plain; charset=iso-8859-1
```

El sistema que recibe el mensaje debe reconocer este campo y tomar las medidas apropiadas cuando muestra el mensaje. El valor por defecto para mensajes *text/plain* (texto simple) es un valor de *charset* (conjunto de caracteres) de *us-ascii*.

Para poder mostrar mensajes con conjuntos de caracteres distintos al ASCII, *elm* debe saber cómo mostrar esos caracteres. Por defecto, cuando *elm* recibe un mensaje con un campo *charset* distinto de *us-ascii* (o un tipo de contenido distinto de *text/plain*, a todos los efectos), intenta mostrar el mensaje usando un comando llamado *metamail*. Los mensajes que requieren *metamail* para ser mostrados aparecen con una '`M`' en la primera columna en la pantalla de listado de mensajes (*overview*).

Como el conjunto de caracteres nativo de Linux es ISO-8859-1, llamar a *metamail* no es necesario para mostrar mensajes que usen dicho conjunto. Si se le dice a *elm* que la pantalla entiende ISO-8859-1, no usará *metamail* sino que mostrará el mensaje directamente. Esto se puede hacer definiendo la siguiente opción en el *elm.rc* global:

```
displaycharset = iso-8859-1
```

Obsérvese que se puede definir esta opción incluso cuando nunca vayamos a enviar o recibir mensajes que realmente contengan caracteres distintos del ASCII. Esto es así porque la gente que envía esos mensajes, usualmente configura su programa de correo para que incluya el campo `Content-Type:` (tipo de contenido) adecuado en el encabezado de correo por defecto, vayan o no a enviar mensajes sólo ASCII.

De todos modos, definir esta opción en *elm.rc* no es suficiente. El problema es que cuando muestra los mensajes con el paginador incorporado, *elm* llama a una función de biblioteca por cada carácter para determinar si es mostrable o no. Por defecto, esta función sólo reconoce caracteres ASCII como mostrables, y muestra todos los demás como "`^?`". Debemos solucionar esto definiendo la variable de entorno *LC_CTYPE* como *ISO-8859-1*, que le indica a la biblioteca que acepte caracteres Latin-1 como mostrables. El soporte para esta y otras características está disponible a partir de la *libc-4.5.8*.

Cuando enviamos mensajes que contienen caracteres especiales del ISO-8859-1, debemos asegurarnos de definir dos variables más en el fichero *elm.rc*:

```
charset = iso-8859-1
textencoding = 8bit
```

Esto hace que *elm* defina el conjunto de caracteres como ISO-8859-1 en el encabezado de correo, y lo envíe como valores de 8 bit (el comportamiento por defecto es recortar todos los caracteres a 7 bit).

Por supuesto, cualquiera de estas opciones se puede definir también en el fichero *elmrc* privado en lugar de en el global.

Capítulo 14

Cómo configurar y poner en marcha *smail*

Este capítulo es una breve introducción a la forma de configurar *smail* y además dará una idea general de la funcionalidad que este programa provée. Aunque *smail* es muy similar en comportamiento a *sendmail*, sus archivos de configuración son totalmente diferentes.

El archivo de configuración principal es */usr/lib/smail/config*. Este archivo es el que se debe editar para ajustar los valores específicos al sistema que se está configurando. Si únicamente es un ordenador terminal de UUCP, serán relativamente pocas las opciones a cambiar. Hay además otros archivos que configuran las opciones de encaminamiento y transporte que se pueden modificar; se hablará brevemente sobre la forma de hacerlo.

La forma de operación normal de *smail* hace que procese y entregue todo el correo de entrada inmediatamente. Si se tiene un tráfico relativamente alto, se puede preferir que *smail* guarde todos los mensajes en una *cola*, y los procese a intervalos regulares.

Cuando se trabaja con correo dentro de una red TCP/IP, es frecuente que *smail* funcione como demonio: en el momento de arrancar la máquina, se invoca desde el archivo *rc.inet2*, y se coloca en segundo plano, desde donde espera que haya una conexión TCP que entre por el puerto SMTP (el puerto 25 es lo normal). Este esquema es muy bueno cuando se espera una gran cantidad de tráfico, pues *smail* no se lanza por separado para cada conexión que ingresa. La alternativa es usar a *inetd* como el administrador del puerto SMTP, y lanzar una copia de *smail* cada vez que haya una conexión en este puerto.

El programa *smail* tiene muchas opciones que se usan para controlar su comportamiento; describirlas una por una en detalle no es de gran utilidad. Afortunadamente *smail* tiene varios modos estándar de operación que se habilitan cuando es invocado con un nombre específico tal como *rmail* o *smtpd*. Es común que estos nombres específicos sean enlaces simbólicos al binario de *smail*. Se verán más de éstos cuando se discutan algunas otras características de *smail*.

Hay dos enlaces a *smail* que deben existir siempre: */usr/bin/rmail* y */usr/sbin/sendmail*.[1]

Cuando se crea y se envía un mensaje de correo con un agente de usuario tal como *elm*, el mensaje se pasará a *rmail* para su entrega, con la lista de destinatarios dada en la línea de comandos. Lo mismo sucede con el correo que entra vía UUCP. Algunas versiones de *rmail*, sin embargo, invocan a */usr/sbin/sendmail* en vez de a *rmail*, por lo que son necesarios ambos enlaces. Por ejemplo, si *smail* está en */usr/local/bin*, se debe escribir lo siguiente en la línea de comandos:

```
# ln -s /usr/local/bin/smail /usr/bin/rmail
# ln -s /usr/local/bin/smail /usr/sbin/sendmail
```

Si se quiere investigar más sobre los detalles de configuración de *smail*, se debe buscar en las páginas del manual *smail(1)* y *smail(5)*. Si no estuviesen incluidas en su distribución preferida del Linux, se pueden obtener junto con el código fuente de *smail*.

14.1 Configuración de UUCP

Para usar *smail* en un entorno que sólo tiene UUCP, la instalación básica es muy sencilla. Primero se debe asegurar de que estén los dos enlaces simbólicos a *rmail* y *sendmail* mencionados anteriormente. Si se espera recibir conexiones de SMTP de otros sitios, también se debe hacer un enlace de *rsmtp* a *smail*.

En la distribución de *smail* de Vince Skahan, se encuentra un archivo muestra de configuración. Su nombre es *config.sample* y está en */usr/lib/smail*. Se debe copiar a *config* y editarlo para ajustar los valores específicos de su sistema.

Suponiendo que su máquina se llama *swim.twobirds.com*, y está registrado en los mapas UUCP como *swim* y su relevo UUCP es *ulysses*, entonces el archivo *config* podría ser como el siguiente:

```
#
# Los nombres de nuestros dominios
visible_domain=two.birds:uucp
#
```

[1] Ésta es la nueva ubicación estándar de *sendmail* de acuerdo con el Estándar del Sistema de Archivos de Linux. Otra ubicación común es */usr/lib*.

```
# Nuestro nombre en los mensajes que viajan al exterior
visible_name=swim.twobirds.com
#
# Tambien se usa este nombre uucp
uucp_name=swim.twobirds.com
#
# Nuestro relevo UUCP
smart_host=ulysses
```

La primera instrucción le indica a *smail* los dominios a los que su sistema pertenece. Se deben insertar sus nombres aquí, separados con signos de punto y coma. Si el nombre de su sistema está registrado en los mapas de UUCP, será necesario agregar además la palabra *uucp*. Cuando se manipula un mensaje de correo, *smail* determina el nombre de su nodo usando una llamada de sistema *hostname(2)* y revisa la dirección del destinatario con respecto al nombre del nodo, revisando cada uno de los nombres de la lista. Si la dirección coincide con cualquiera de estos nombres, o el nombre del sistema no está calificado, el receptor se considera local y *smail* intenta entregar el mensaje a un usuario o alias dentro del sistema local. En cualquier otro caso, el receptor se considera remoto y se intenta entregar al nodo adecuado.

La palabra clave *visible_name* debe contener un solo nombre de dominio totalmente calificado de la máquina que se desea utilizar para los mensajes que se envían hacia afuera. Este nombre se usa cuando se genera la dirección de quien envía el correo en todos los mensajes de salida. Es importante asegurarse de que el nombre que se use sea reconocido por *smail* como una referencia al sistema local (i.e. el nombre del ordenador con uno de los dominios listados en el atributo *visible_domain*). Si no se hiciese de esta forma, las respuestas a los mensajes enviados rebotarán hacia afuera del nodo local.

La última instrucción pone la ruta utilizada para el encaminamiento del relevo UUCP (descrito en la sección 13.4). Con este cambio mostrado, *smail* enviará cualquier correo dirigido hacia direcciones remotas al relevo. Como los mensajes serán entregados a través de UUCP, el atributo debe especificar un sistema conocido para los programas UUCP que corran en su sistema. Consulte el capítulo 12 sobre el tema de cómo hacer que su nodo sea conocido por UUCP.

Hay una opción que se utiliza en el archivo anterior que aún no ha sido explicada; ésta es *uucp_name*. La razón para utilizar esta opción es la siguiente: normalmente *smail* utiliza el valor que devuelve *hostname(2)* para cosas que hace el UUCP tales como poner en el encabezado *From_* el camino de regreso del correo. Si el nombre del nodo *no* está registrado en el mapa de UUCP, es necesario indicar a *smail* que en vez de éste utilice el nombre de dominio completamente calificado.[2] Esto se puede hacer agregando la opción *uucp_name* al

[2]La razón de esto es: Suponga que el nombre de su sistema es *monad* y que no está registrado en los

archivo de configuración *config*.

Hay otro archivo en */usr/lib/smail*, que se llama *paths.sample*. Este es un ejemplo de la forma que tiene un archivo de caminos, *paths*. Sin embargo, este archivo no es necesario a menos que se tengan enlaces de correo a más de un lugar. Si fuese necesario hacerlo, se debe escribir uno nuevo o generar uno partiendo de los mapas de Usenet. El archivo *paths* se describirá más adelante, en este mismo capítulo.

14.2 Configuración para una red local

Si está funcionando una instalación con dos o más nodos conectados por medio de una red local, es necesario designar a uno de ellos para que maneje la conexión UUCP con el mundo exterior. Entre las máquinas de la red local, es muy probable que se quiera intercambiar correo con SMTP sobre TCP/IP. Suponga que se tiene nuevamente el ejemplo de la Cervecera Virtual, y **vstout** se configura como una pasarela UUCP.

En un entorno de red, es preferible mantener todos los archivos con el correo de los usuarios en un solo sistema de archivos, que puede ser montado con NFS desde todas las demás máquinas. Esto permite a los usuarios desplazarse de máquina en máquina sin tener que mover su correo por todos lados (o peor, revisar tres o cuatro ordenadores para ver su correo recién recibido cada mañana). Así mismo, es deseable hacer que las direcciones de los usuarios sean independientes del ordenador en la cual el correo se almacena. Es una práctica común utilizar el nombre del dominio como la dirección de quien envía el correo en vez de utilizar el nombre de la máquina servidora del correo. El usuario Janet, por ejemplo, podría especificar su dirección como **janet@vbrew.com** en vez de **janet@vale.vbrew.com**. A continuación se explicará como hacer que el servidor reconozca el nombre del dominio como un nombre válido para su instalación.

Otra forma de mantener todos los apartados postales en un anfitrión central es utilizar POP o IMAP. POP quiere decir, por sus siglas en inglés *Post Office Protocol*, es decir, *Protocolo de Oficina Postal* y permite a los usuarios tener acceso a sus archivos de correo a través de una conexión TCP/IP. IMAP, o *Protocolo de Acceso Interactivo al Correo* por sus siglas en inglés de *Interactive Mail Access Protocol*, es similar a POP, excepto que es más general. Ambos clientes y servidores para IMAP y POP han sido portados a Linux, y están disponibles en **sunsite.unc.edu** bajo el directorio */pub/Linux/system/Network*.

---

mapas y además hay un lugar registrado en los mapas que se llama *monad*. Cada correo que se dirija a *monad!root*, aún cuando haya sido enviado desde un vecino directo UUCP, viajará hasta el otro *monad*. Esto es una molestia para todos.

14.2.1 Cómo escribir los archivos de configuración

La configuración para la Cervecera funciona de la siguiente forma: todos los nodos, con excepción del servidor de correo **vstout**, encaminan todo el correo que va hacia el exterior hacia este servidor, utilizando la técnica de encaminamiento al relevo de correo. **vstout** encamina todo el correo que va hacia el exterior al verdadero nodo de relevo que, a su vez, envía todo el correo de la Cervecera; este último nodo se llama **moria**.

El archivo estándar *config* para todas las máquinas con la excepción de **vstout** es como sigue:

```
#
# Nuestro dominio:
visible_domain=vbrew.com
#
# El nombre que usamos:
visible_name=vbrew.com
#
# Encaminamiento al relevo: via SMTP hacia vstout
smart_path=vstout
smart_transport=smtp
```

Esto es muy parecido a lo que se ha hecho para configurar un sistema que sólo funciona con UUCP. La diferencia principal es que el medio de transporte utilizado para enviar el correo al nodo de relevo es SMTP. El atributo *visible_domain* hace que *smail* utilice el nombre del dominio en vez de utilizar el nombre del sistema local en todo el correo de salida.

En la pasarela de correo UUCP **vstout** el archivo *config* es ligeramente distinto:

```
#
# Los nombres de nuestros sistemas:
hostnames=vbrew.com:vstout.vbrew.com:vstout
#
# La forma en que nos llamamos a nosotros mismos:
visible_name=vbrew.com
#
# En el mundo uucp, somos conocidos como vbrew.com
uucp_name=vbrew.com
#
# Transporte inteligente: via uucp hacia moria
smart_path=moria
smart_transport=uux
#
```

```
# somos la autoridad para nuestro dominio
auth_domains=vbrew.com
```

Este archivo de configuración, *config*, utiliza un esquema diferente para indicar a *smail* cómo se llama el sistema local. En vez de dar una lista de dominios y permitir que busque el nombre del nodo con una llamada al sistema, se especifica una lista explícitamente. La lista de arriba contiene tanto el dominio completamente calificado como el del sistema no calificado, y el nombre del dominio completo en sí mismo. Esto hace que *smail* reconozca a **janet@vbrew.com** como una dirección local, y entregue el mensaje a **janet**.

La variable *auth_domains* indica el nombre de los dominios para los cuales **vstout** es considerado como autoridad. Esto es, si *smail* recibe cualquier correo con una dirección hacia ***host*.vbrew.com** en donde ***host*** no corresponde a ninguna máquina existente, se rechaza el mensaje y se devuelve al remitente del mismo. Si esta línea no está, cualquier mensaje rechazado será enviado nuevamente al relevo de correo, quien lo mandará a **vstout**, y así sucesivamente hasta que se descarte por exceder la cuenta máxima de saltos.

14.2.2 Cómo ejecutar *smail*

La primera cosa que se debe hacer es decidir si se ejecutará *smail* como un demonio independiente, o si se permitirá que *inetd* administre el puerto SMTP e invoque a *smail* cuando un cliente solicite una conexión SMTP. Normalmente es preferible la operación como un demonio independiente en el servidor de correo, debido a que esto carga la computadora menos que lanzar una copia nueva de *smail* cada vez que se solicite una conexión individual. Cuando un servidor de correo entrega casi todo el correo que recibe directamente a los usuarios, es preferible optar por la operación con *inetd*.

Independientemente del modo de operación que se haya elegido para cada anfitrión individual, es importante asegurarse que se tiene la siguiente línea en el archivo */etc/services*:

```
smtp            25/tcp          # Simple Mail Transfer Protocol
```

Esto define el número del puerto TCP que *smail* utilizará para las conexiones SMTP. 25 es el puerto estándar definido por el RFC de Números de Puerto Asignados.

Cuando se ejecuta como demonio, *smail* se coloca a sí mismo en segundo plano, y esperará a que ocurra una conexión en el puerto SMTP. Cuando haya una conexión, lanza un proceso y conduce una conversación SMTP en dicho puerto. El demonio *smail* se lanza normalmente invocándolo desde el *script rc.inet2* con la siguiente instrucción:

```
/usr/local/bin/smail -bd -q15m
```

El modificador `-bd` indica que se funcionará como demonio, y `-q15m` hace que se procesen los mensajes acumulados en la cola cada 15 minutos.

Si en cambio, se quiere utilizar *inetd*, el archivo */etc/inetd.conf* deberá contener una línea como la siguiente:

```
smtp    stream  tcp nowait  root  /usr/sbin/smtpd smtpd
```

smtpd debe ser un enlace simbólico al binario de *smail*. Recuerde que tiene que forzar a que *inetd* relea *inetd.conf* enviándole una señal *HUP* después de hacer estos cambios.

El modo demonio y el modo *inetd* son mutuamente excluyentes. Si se ejecuta *smail* como demonio, asegúrese de que esté comentada cualquier línea en *inetd.conf* para el servicio *smtp*. De manera similar, cuando se tenga a *inetd* como administrador de *smail*, asegúrese de que *rc.inet2* no lanza al demonio *smail*.

14.3 Si no logra pasar...

Si algo va mal con la instalación, hay algunas herramientas que pueden ayudar a encontrar cuál es la raíz del problema. El primer lugar que se debe revisar es el conjundo de archivos de registro de *smail*. Están en */var/spool/smail/log*, y se llaman *logfile* y *paniclog*, respectivamente. El primero lista todas las transacciones, mientras que el último sólo se usa cuando haya mensajes de error relacionados con errores en la configuración y similares.

Un ejemplo típico de una línea en el *logfile* es el siguiente:

```
04/24/94 07:12:04: [m0puwU8-00023UB] received
|            from: root
|         program: sendmail
|            size: 1468 bytes
04/24/94 07:12:04: [m0puwU8-00023UB] delivered
|             via: vstout.vbrew.com
|              to: root@vstout.vbrew.com
|         orig-to: root@vstout.vbrew.com
|          router: smart_host
|       transport: smtp
```

Esto muestra que un mensaje de **root** a **root@vstout.vbrew.com** ha sido correctamente entregado al sistema **vstout** a través de SMTP.

Los mensajes que *smail* no pudo entregar generan una línea similar en el archivo de registro, pero con el mensaje de error en vez de la parte que dice entregado (`delivered`):

```
04/24/94 07:12:04: [m0puwU8-00023UB] received
|           from: root
|        program: sendmail
|           size: 1468 bytes
04/24/94 07:12:04: [m0puwU8-00023UB] root@vstout.vbrew.com ... deferred
 (ERR_148) transport smtp: connect: Connection refused
```

El error de arriba es típico para una situación en la cual *smail* reconoce correctamente que el mensaje debería ser entregado a **vstout** pero que no fue posible establecer la conexión al servicio SMTP en **vstout**. Si esto sucede, es posible que tenga un problema de configuración o bien que el soporte TCP esté ausente de los binarios del *smail*.

Este problema no es tan raro de encontrar. Hay varios binarios de *smail* que vienen con distribuciones de Linux y que no tienen soporte de red TCP/IP. Si este es su caso, debe recompilar el programa *smail*. Una vez instalado *smail*, se debe revisar si se tiene soporte de red TCP haciendo un telnet al puerto SMTP de su máquina. Una conexión exitosa al servidor SMTP se muestra a continuación (la entrada por teclado se marca con ***este tipo de letra***):

```
$ telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 monad.swb.de Smail3.1.28.1 #6 ready at Sun, 23 Jan 94 19:26 MET
QUIT
221 monad.swb.de closing connection
```

Si esta prueba no produce el mensaje de SMTP (la línea que comienza con el código 220), debe asegurarse de que su configuración es *verdaderamente* correcta antes de recompilar *smail*, como se describirá a continuación.

Si hay algún problema con *smail* que no se pueda localizar con el mensaje de error que *smail* genera, se pueden activar los mensajes de depuración. Para hacer esto, se debe utilizar el modificador `-d`, seguido de un número opcional que especifique el nivel de detalle de la información (no se debe dejar ningún espacio entre el modificador y el argumento numérico). Entonces, *smail* mostrará un informe de su operación en la pantalla que dará más pistas acerca de lo que puede estar mal.

14.3.1 Cómo compilar *smail*

Si está seguro de que su *smail* carece de soporte de red TCP, es necesario obtener el código fuente. Es posible que ya esté incluído en su distribución si la obtuvo en CD-ROM, si no fuese así, se puede conseguir en la red vía FTP.[3]

La mejor forma de compilar *smail*, es comenzar con el conjunto de archivos de configuración de la distribución de Vince Skahan. Para incluir el controlador de TCP dentro de la compilación, se debe poner la macro *DRIVER_CONFIGURATION* en el archivo *conf/EDITME* con el parámetro *bsd-network* o *arpa-network*. El primero se utiliza para las instalaciones de red local, pero cuando se está en Internet es necesario usar *arpa-network*. La diferencia entre estas dos es que la segunda tiene un manejador especial para el servicio BIND que permite reconocer registros MX, lo cual la primera no puede hacer.

14.4 Modos de entrega de correo

Como se mencionó anteriormente, *smail* es capaz de entregar los mensajes inmediatamente o encolarlos para un proceso posterior. Si se decide encolar los mensajes, *smail* guardará todo el correo en el directorio *messages* debajo de */var/spool/smail*. No se procesarán hasta que se le indique explícitamente que lo haga (a este proceso se le conoce como "ejecutar la cola").

Se puede selecionar uno de tres modos de entrega definiendo el atributo *delivery_mode* en el archivo *config* para que esté como *foreground*, *background*, o *queued*. Es decir, proceso normal, proceso en segundo plano, o proceso en cola. Estas opciones seleccionan la entrega normal (procesamiento inmediato de los mensajes de entrada), en segundo plano (los mensajes son entregados por medio de un hijo del proceso receptor: el proceso padre muere inmediatamente después de la creación del hijo), y el encolado. El correo de entrada siempre será encolado independientemente de esta opción si la variable booleana *queue_only* está puesta en el archivo *config*.

Si se activa el modo de cola, se debe asegurar de que las colas se revisen regularmente; probablemente cada 10 o 15 minutos. Si se ejecuta *smail* como demonio, se debe agregar la opción `-q10m` en la línea de comandos para procesar la cola cada 10 minutos. De forma alternativa, se puede invocar *runq* desde el *cron* en esos intervalos de tiempo. *runq* deberá ser un enlace a *smail*.

Se puede revisar la cola del correo al invocar *smail* con la opción `-bp`. De manera equivalente, se puede hacer que *mailq* sea un enlace a *smail*, e invocar *mailq*:

```
$ mailq -v
m0pvB1r-00023UB From: root  (in /var/spool/smail/input)
                Date: Sun, 24 Apr 94 07:12 MET DST
```

[3]Si compró una distribución Linux a un proveedor comercial, se puede solicitar el código fuente con "un cargo de envío nominal" (que sólo cubra los gastos), de acuerdo con las condiciones de copia de *smail*.

```
                 Args: -oem -oMP sendmail root@vstout.vbrew.com
    Log of transactions:
     Xdefer: <root@vstout.vbrew.com> reason: (ERR_148) transport smtp:
     connect: Connection refused
```

Esto muestra un solo mensaje que está esperando en la cola de mensajes. El registro de transacciones (que sólo se mostrará si se da a *mailq* la opción `-v`) puede dar una explicación adicional de por qué el mensaje está esperando para su entrega. Si aún no se ha intentado entregar el mensaje, no se mostrará la información del registro.

Aún cuando no se utilice el modo de cola, *smail* pondrá de forma ocasional los mensajes en la cola cuando falle la entrega inmediata por una razón transitoria. Para las conexiones SMTP, esto puede ser debido a que el nodo siguiente sea un inalcanzable; pero los mensajes pueden también ser pospuestos cuando el sistema de archivos del receptor esté lleno. En cualquier caso, debe poner una cola que se revise, por ejemplo, cada hora (utilizando *runq*), porque si no, cualquier mensaje pospuesto se quedará encolado indefinidamente.

14.5 Otras opciones del fichero *config*

Hay otras muchas opciones en el archivo *config*, algunas poco usadas en sistemas sencillos. Sin embargo, mencionaremos algunas que sí que serán útiles con frecuencia:

error_copy_postmaster
: Si esta variable boleana se pone, cualquier error generará un mensaje al administrador de correo. Normalmente esto sólo se hace para los errores que se deben a una configuración incorrecta. La variable puede activarse poniéndola en el archivo *config*, precedida por un signo de suma (+).

max_hop_count
: Si la cuenta de saltos para un mensaje (i.e. el número de nodos que se han atravesado) es igual o excede a este número, los intentos de entrega producirán un mensaje de error que será enviado a quien generó el mensaje. Esto se utiliza para prevenir que los mensajes entren en un ciclo infinito. La cuenta de saltos se calcula generalmente a partir del número de campos *Received:* que se encuentran en el encabezado del correo. Además, esta cuenta también puede ser ajustada de forma manual utilizando la opción `-h` en la línea de comandos.

  Esta variable tiene como valor por defecto 20.

postmaster
: La dirección del administrador de correo. Si la dirección **Postmaster** no puede ser resuelta como dirección local válida, entonces ésta se utiliza como

último recurso. El valor por defecto es **root**.

14.6 Encaminamiento de mensajes y entrega

smail divide la entrega del correo en tres partes, la ruta, el módulo de entrega local y el módulo de transporte.

El módulo de encaminamiento resuelve todas las direcciones remotas, determinando el nodo al que el mensaje será enviado y el transporte que será utilizado. Dependiendo de la naturaleza del enlace, se utilizarán transportes diferentes tales como UUCP o SMTP.

Las direcciones locales se dan al módulo de entrega local que resuelve cualquier reenvío o alias. Por ejemplo, la dirección podría ser un alias o una lista de correo, o el usuario podría querer reenviar su correo a otra dirección. Si la dirección resultante es remota, se maneja de nuevo en el módulo de encaminamiento, de otra forma se asigna a un transporte para su entrega local. Normalmente, la acción a realizar será entregar a un archivo de correo, pero los mensajes también pueden ser pasados a la entrada de un comando (por ejemplo, un filtro de correo que el usuario quiera establecer) o agregados a un archivo arbitrario cualquiera.

El módulo de transporte, finalmente, es el responsable de la entrega, independientemente del método que se haya escogido. Intenta entregar el mensaje y en caso de problemas, puede devolver un mensaje al remitente o posponer la entrega para intentarlo de nuevo más tarde.

Con *smail* se tiene mucha flexibilidad para configurar estas tareas. Para cada una de ellas, hay varios controladores[4] disponibles, de los cuales se puede elegir el más adecuado. Se debe indicar a *smail* la elección a través de los siguientes archivos: *routers*, *directors* y *transports*, que se encuentran en */usr/lib/smail*. Si estos archivos no existiesen, se toman valores por defecto razonables que funcionan en la mayor parte de los sistemas que utilizan SMTP o UUCP como transporte. Si se quiere cambiar la política de encaminamiento de *smail*, o modificar un transporte, es conveniente obtener los archivos ejemplo que vienen con la distribución de los programas fuente de *smail*[5], copiar los archivos ejemplo a */usr/lib/smail*, y modificarlos de acuerdo con sus necesidades. Los archivos de ejemplos de configuración están también en el Apéndice B.

[4] Aquí, conocemos por *controladores* a los distintos módulos internos de *smail* capaces de utilizar un método de entrega de mensajes u otro. Así, tenemos controladores para UUCP o para SMTP

[5] Los archivos de configuración por defecto se encuentran en *samples/generic* bajo el subdirectorio de los programas fuente.

14.7 Mensajes de encaminamiento

Cuando se le da un mensaje, *smail* revisa primero si el destino está en el sistema local o en un nodo remoto. Si la dirección del ordenador destino corresponde a uno de los nodos locales configurados en el archivo *config*, el mensaje es tratado por el módulo de entrega local. Si no fuese así, *smail* transmite la dirección del destino a varios controladores de encaminado para encontrar a qué máquina se debe transmitir el mensaje. Los controladores se pueden indicar en el archivo *routers*; si este archivo no existe, se utiliza un conjunto de encaminadores por defecto.

El nodo destino se pasa a todos los encaminadores por turno, y aquél que encuentra la ruta más específica es seleccionado. Por ejemplo, suponga que hay un mensaje dirigido a **joe@foo.bar.com** y que un encaminador conoce una ruta para todos los nodos que pertenecen al dominio **bar.com**, mientras que otro tiene la información sobre el camino directo al sistema **foo.bar.com**. Como el segundo es más específico, es elegido sobre el primero. Si hubiese dos encaminadores que proveen una solución correcta e igual de específica, se elige al primero que esté en el archivo *routers*.

A continuación, el encaminador elegido especifica qué transporte utilizará, por ejemplo UUCP, y genera así una nueva dirección destino. La nueva dirección se pasa al transporte junto con el nombre del sistema a quien se le debe pasar el mensaje. En el ejemplo anterior, *smail* podría encontrar que **foo.bar.com** se puede encontrar vía UUCP utilizando la trayectoria **ernie!bert**. Así generará un nuevo destino **bert!foo.bar.com!user**, y utilizará esta dirección, a través del transporte UUCP, como la que será transmitida a **ernie**.

Cuando se utilice la configuración por defecto, los siguientes encaminadores estarán disponibles:

- Si la dirección del nodo destino se puede resolver utilizando las llamadas de biblioteca *gethostbyname(3)* o *gethostbyaddr(3)*, el mensaje será entregado vía SMTP. La única excepción es si la dirección que se encuentra se refiere al sistema local, en cuyo caso será enviado al módulo de entrega local.

  smail también reconoce las direcciones IP escritas como cuarteto de puntos como nombre legal de máquina, siempre y cuando pueda ser resuelto a través de una llamada a *gethostbyaddr(3)*. Por ejemplo **scrooge@[149.76.12.4]** podría ser válida aunque muy rara como dirección para **scrooge** en **quark.physics.groucho.edu**.

  Si su máquina está en el Internet, estos encaminadores no son lo que usted necesita, debido a que no soportan registros MX. Vea más adelante lo que se debe hacer en este caso.

- Si la base de datos de alias de trayectorias, */usr/lib/smail/paths* existe, *smail* tratará de buscar en el archivo al nodo destino (restándole la extensión **.uucp** si la hubiera). El correo a una dirección que coincida con este encaminador será entregado utilizando UUCP, a través de la trayectoria que se haya encontrado en la base de datos.

- La dirección del nodo (restándole la extensión **.uucp** si la hubiera) se compara con la salida de la instrucción *uuname* para revisar si el sistema destino es un vecino UUCP. Si éste es el caso, el mensaje será entregado utilizando el transporte UUCP.
- Si la dirección no coincide en ninguno de los encaminadores citados anteriormente, será entregado utilizando un relevo de correo. La trayectoria al nodo de relevo así como el medio de transporte que será utilizado se ponen en el archivo *config*.

Los valores por defecto funcionan para la mayor parte de las instalaciones sencillas, pero no son útiles si las necesidades de encaminamiento son algo más complejas. Si se enfrenta con uno de los problemas que se discutirán a continuación, es necesario instalar su propio archivo *routers* para cambiar los valores por defecto. Un archivo ejemplo *routers* con el que se puede empezar está en el apéndice B. Algunas distribuciones de Linux traen además, un conjunto de archivos de configuración hechos a la medida para solventar esas dificultades.

Es probable que el peor de los problemas surja cuando su máquina viva en un universo dual con enlaces de marcado telefónico vía IP y UUCP. Entonces se tendrán nombres de nodos en el archivo *hosts* con los cuales sólo se comunica ocasionalmente a través de un enlace SLIP, y *smail* intentará entregar cualquier correo por medio de estos sistemas usando SMTP. Este comportamiento no es deseable normalmente debido a que, si el enlace SLIP se activa de forma regular, SMTP es mucho más lento que mandar el correo con UUCP. Con los valores por defecto, no se puede evitar que *smail* se porte mal.

Este problema se puede evitar revisando con *smail* el archivo *paths* antes de preguntar por el sistema de resolución, y poner a todos los nodos a los que se quiera forzar la entrega vía UUCP en el archivo *paths*. Si *nunca* se quiere enviar ningún mensaje sobre SMTP, se pueden eliminar poniendo como comentarios todos los encaminadores que están basados en el sistema de resolución.

Otro problema es que las opciones por defecto no proporcionan encaminado de correo Internet verdadero, debido a que un encaminador basado en un DNS no evalúa los registros MX. Para habilitar el soporte completo para el encaminamiento de correo Internet, es necesario eliminar al encaminador poniéndolo como comentario, y quitar el comentario de aquél que utiliza BIND. Sin embargo, algunas distribuciones de Linux incluyen binarios de *smail* que no tienen el soporte para BIND incluido y, si se habilita BIND, se obtendrá un mensaje en el archivo *paniclog* que dice "`router inet_hosts: driver bind not found`", es decir, "no se encuentra el controlador de bind", por lo que será necesario obtener el código fuente y recompilar *smail* (vea la sección 14.2 más arriba).

Para concluir, generalmente no es buena idea utilizar el controlador *uuname*. Por una parte, generará un error de configuración cuando no se tenga UUCP instalado, debido a que no encontrará al programa *uuname*. Por la otra, es que se tienen más sistemas listados en su archivo *Systems* de UUCP que aquéllos con los que se mantiene correo. Éstos pueden ser nodos con los cuales únicamente se intercambian noticias, o sistemas de los cuales se bajan archivos ocasionalmente vía UUCP anónimo, pero no se tiene más tráfico que éste.

Para resolver el primer problema, se puede sustituir el programa *uuname* con un *script* que haga un simple *exit 0*. La solución más general es, sin embargo, editar el archivo *routers* y borrar todo el driver.

14.7.1 La base de datos de trayectorias *paths*

El programa *smail* espera encontrar una base de datos de alias de trayectorias en el archivo *paths* en el subdirectorio */usr/lib/smail*. Este archivo es opcional, por lo que si no se quiere hacer ningún encaminamiento por medio de alias de trayectorias, simplemente se borra el archivo *paths*.

El archivo *paths* debe ser un archivo ASCII ordenado que contiene líneas que mapeen los nombres de los nodos destino a trayectorias UUCP con signos de admiración. El archivo tiene que estar ordenado debido a que *smail* utiliza búsqueda binaria para encontrar un sitio. No se permiten comentarios en este archivo, y el nombre del sitio debe estar separado de la trayectoria utilizando un carácter de tabulación. Las bases de datos de alias de trayectorias se discuten con más detalle en el capítulo 13.

Si se genera este archivo a mano, es importante asegurarse de incluir todos los nombres válidos para un sistema. Por ejemplo, si a una máquina se le conoce por un nombre simple UUCP y un nombre de dominio totalmente calificado, se debe añadir una línea para cada uno de ellos. El archivo debe estar ordenado (para ello, enviarlo al comando *sort(1)*).

Si su nodo es simplemente terminal, no será necesario tener un archivo *paths*: basta con ajustar los atributos de nodo de relevo en su archivo *config*, y dejarle todo el trabajo de encaminamiento que su correo genere.

14.8 Cómo entregar mensajes a las direcciones locales

Es común que una dirección local de correo sea sólo el nombre del usuario, en cuyo caso el mensaje se entrega en su archivo de correo, */var/spool/mail/*`usuario`. En otros casos se incluyen alias y nombres de la lista de correo, y correo redirigido por el usuario. En estos casos, la dirección local se expande a una nueva lista de direcciones que pueden ser locales o remotas.

Independientemente de estas direcciones “normales”, *smail* puede manejar otros tipos de destino para los mensajes locales tales como nombres de archivos o comandos (que reciben el mensaje por su entrada estándar). Éstas no son propiamente direcciones, de tal forma que no se puede mandar correo, por ejemplo a **/etc/passwd@vbrew.com**; sólo son válidas si se han tomado de un archivo alias o de redireccionamiento.

Un *nombre de archivo* es cualquier cosa que comience con una diagonal (/) o un tilde (˜). El segundo se refiere al directorio inicial del usuario, y es posible sólo si el nombre del archivo ha sido tomado de *.forward* o una línea de redirección del archivo de correo (ver más abajo). Cuando *smail* manda el mensaje a un archivo, lo añade al final del archivo y, de ser necesario, lo puede también crear.

Una *instrucción por tubería* puede ser cualquier comando UN⋆X precedido por el símbolo (|). Esto hace que *smail* envíe el comando al shell junto con sus argumentos, pero sin el ‘|’ que lo encabeza; pasando el mensaje a la entrada estándar del comando.

Por ejemplo, para meter una lista de correo en un grupo de noticias local, se podría utilizar un *script* llamado *gateit* y configurar un alias local que entregue todos los mensajes de esta lista de correo al *script*, utilizando “*—gateit*”.

Si la invocación contiene un espacio en blanco, se debe encerrar entre comillas dobles. Debido a los problemas de seguridad que pueden ser ocasionados aquí, es importante cuidar que no se ejecute el comando si la dirección ha sido obtenida de alguna forma dudosa (por ejemplo, si el archivo de alias del cual la dirección se ha obtenido puede ser escrito por cualquiera).

14.8.1 Usuarios locales

El caso más común para una dirección local es mostrar el archivo de correo del usuario. Este apartado postal está en */var/spool/mail* y tiene el nombre del usuario. También es propiedad de éste, con grupo **mail** y tiene el modo 660. Si no existe, *smail* lo crea.

Observe que aunque */var/spool/mail* es el lugar estándar para poner los archivos de correo, algunas aplicaciones tienen diferentes trayectorias compiladas en ellos, por ejemplo */usr/spool/mail*. Si la entrega a los usuarios en su sistema falla constantemente, se puede intentar hacer un enlace simbólico a */var/spool/mail* para ver si esta situación mejora.

Hay dos direcciones que *smail* necesita para funcionar: **MAILER-DAEMON** y **Postmaster**. Cuando se devuelve un mensaje de informe debido a un correo que no pudo ser entregado, se envía una copia a la cuenta del administrador postal (el **postmaster**) para su revisión (en el caso de que este mensaje pudiera ser debido a un problema de configuración). El usuario **MAILER-DAEMON** se utiliza como la dirección del remitente del mensaje devuelto.

Si estas direcciones no tienen nombres de cuentas válidas en su sistema, *smail* mapea implícitamente **MAILER-DAEMON** a **postmaster**, y **postmaster** a **root**. Es conveniente cambiar esto dándole un alias **postmaster** al responsable del mantenimiento de los programas de correo.

14.8.2 Reenvío

Un usuario puede redirigir su correo a una dirección alternativa utilizando uno de los dos métodos que soporta *smail*. Una opción es poner

```
Forward to receptor,...
```

en la primera línea de su archivo de correo. Esto enviará todo el correo que se reciba a la lista de receptores especificada allí. La otra es crear un archivo *.forward* en el directorio principal del usuario, que contenga una lista de los receptores separados por comas. Con este sistema de redireccionamiento, todas las líneas del archivo son leídas e interpretadas.

Observe que cualquier tipo de dirección puede ser utilizada. Así, un ejemplo práctico del archivo *.forward* para cuando se tome unas vacaciones puede ser

```
janet, "|vacation"
```

La primera dirección entrega el mensaje que llega al archivo de correo de **janet**, mientras que la instrucción *vacation* provoca la devolución de un mensaje que informa al remitente que janet está de vacaciones.

14.8.3 Archivos de alias

El programa *smail* entiende los archivos de *alias* compatibles con los del *sendmail* de Berkeley. Las líneas en el archivo de alias pueden ser de la forma

alias: *receptores*

receptores es una lista de direcciones separadas por comas que será sustituida por el alias. La lista de receptores puede continuar a través de varias líneas si la siguiente línea comienza con un carácter de tabulación.

Hay una característica especial que permite que *smail* maneje listas de correo desde un archivo de alias: si se especifica "`:include:`*`nombrearchivo`*" como receptor, *smail* leerá el archivo especificado, y sustituirá su contenido con una lista de receptores.

El archivo de alias principal es */usr/lib/aliases*. Si se decide hacerlo escribible por todo el mundo, *smail* no entregará ningún mensaje a los comandos de shell que pudiese contener el archivo. Un archivo de ejemplo se muestra a continuación:

```
# vbrew.com archivo /usr/lib/aliases
hostmaster: janet
postmaster: janet
usenet: phil
# La lista de correo de desarrollo de programas.
development: joe, sue, mark, biff
        /var/mail/log/development
owner-development: joe
# Los anuncios de interes general seran enviados
# a todo el personal (lista staff)
announce: :include: /usr/lib/smail/staff,
        /var/mail/log/announce
owner-announce: root
# pasarela a la lista de correos foobar a un grupo de noticias local
ppp-list: "|/usr/local/lib/gateit local.lists.ppp"
```

Si hay un error cuando se entrega a una dirección generada por el archivo *aliases*, *smail* intentará enviar una copia del mensaje de error al "dueño del alias". Por ejemplo, si la entrega a **biff** no se logra cuando se envió un mensaje a la lista de correo **development**, se enviará una copia del mensaje de error al remitente, así como también al **postmaster** y a **owner-development**. Si la dirección del dueño no existe, no se generará el mensaje de error adicional.

Cuando se entrega a un archivo o cuando se invocan programas en el archivo *aliases*, *smail* se convierte en el usuario **nobody** para evitar problemas de seguridad[6]. En especial cuando se entrega a un archivo esto constituye una verdadera molestia. En el archivo de ejemplo que se dio anteriormente, los archivos de registro *.log* deben ser propiedad y ser escribibles por el usuario **nobody**, o la entrega hacia ellos fallará.

[6] N. del T.: *nobody* significa *nadie*, y es un usuario que se utiliza cuando no se identifica al dueño de un proceso, y si bien se desea su ejecución, también es cierto que no deseamos crear un agujero en nuestros mecanismos de seguridad, por lo cual **nobody** es un usuario con los privilegios reducidos al mínimo.

14.8.4 Listas de correo

En vez de utilizar el archivo *aliases*, las listas de correo también pueden ser administradas por medio de archivos en el directorio */usr/lib/smail/lists*. Una lista de correo llamada *nag-bugs* se debe describir en el archivo *lists/nag-bugs*, el cual deberá contener las direcciones de los miembros separadas por comas. La lista puede estar en varias líneas, con líneas de comentarios que comienzan con el símbolo #.

Para cada lista de correo, un usuario (o alias) llamado **owner-*nombredelista*** debe existir; cualquier error que ocurra cuando se resuelva una dirección será enviado a este usuario. Esta dirección se usa también como la dirección del remitente en todos los mensajes de salida en el campo de encabezado `Sender:`.

14.9 Transportes basados en UUCP

Hay varios transportes compilados en *smail* que utilizan el conjunto de programas UUCP. En un entorno UUCP, los mensajes se pasan normalmente al invocar *rmail* en el siguiente nodo, dándole el mensaje en la entrada estándar y la dirección a quien va dirigido en la línea de argumentos. En el sistema, *rmail* deberá ser un enlace al programa *smail*.

Cuando se maneja un mensaje con el transporte UUCP, *smail* convierte la dirección destino a una trayectoria UUCP con símbolos de admiración. Por ejemplo, **user@host** se transformará en **host!user**. Cualquier ocurrencia del operador de direcciones '**%**' será conservada, de tal forma que **user%host@gateway** se convertirá en **gateway!user%host**. Sin embargo, *mail* nunca generará esa dirección por sí mismo.

De manera alternativa, *smail* puede enviar y recibir lotes de BSMPT vía UUCP. Con BSMTP, uno o más mensajes son empaquetados en un solo lote que contiene las instrucciones para que el controlador del correo local funcione como si se hubiera establecido una conexión SMTP real. BSMTP se utiliza frecuentemente en redes de guardar-y-enviar (por ejemplo las basadas en UUCP) para ahorrar espacio en disco. El archivo de ejemplo *transports* del apéndice B contiene un transporte doblado *bsmtp* que genera lotes parciales BSMTP en un directorio de colas. Luego, deben ser combinados en los lotes finales utilizando un script de shell que agrega las instrucciones apropiadas *HELO* y *QUIT*.

Para habilitar el transporte *bsmtp* para enlaces UUCP específicos se deben utilizar los archivos llamados *método* (revise la página del manual *smail(5)* para más detalles). Si se tiene únicamente un enlace UUCP, y se utiliza un encaminado a relevo, se puede habilitar el envío de lotes SMTP poniendo la variable de configuración *smart_transport* a *bsmtp* en vez de *uux*.

Para recibir lotes SMTP sobre UUCP, se debe asegurar que se tiene el mismo programa de decodificación de lotes que el sistema remoto que envía los lotes. Si el nodo remoto utiliza *smail* también, es necesario hacer un enlace llamado *rsmtp* a *smail*. Si el sistema remoto corre *sendmail*, se debe además instalar un script llamado */usr/bin/bsmtp* que haga un simple "*exec rsmtp*" (una enlace simbólico no funcionará).

14.10 Transportes basados en SMTP

El *smail* soporta actualmente un controlador de SMTP para entregar el correo sobre conexiones TCP.[7] Es capaz de entregar un mensaje a cualquier número de direcciones de una máquina, con el nombre de la misma especificado como nombre de dominio totalmente calificado que puede ser resuelto por el software de red, o con la notación de cuarteto de puntos encerrados entre corchetes. En general, las direcciones se resuelven con los controladores de encaminamiento del BIND, *gethostbyname(3)*, o *gethostbyaddr(3)* que lo entregarán al transporte SMTP.

El manejador de SMTP intentará conectarse al sistema remoto inmediatamente a través del puerto *smtp* como está listado en */etc/services*. Si no puede ser alcanzado, o expira el tiempo máximo de espera, la entrega del correo se reintentará posteriormente.

La entrega en Internet requiere que las rutas al nodo destino estén especificadas en el formato *route-addr* descrito en el capítulo 13, en vez de utilizar una trayectoria de signos de admiración.[8] *smail* transformará **user%host@gateway**, en donde **gateway** se alcanza vía **host1!host2!host3**, en la dirección de la ruta-fuente `<@host2,@host3:user%host@gateway>` la cual será enviada como la dirección del remitente a **host1**. Para habilitar dicha transformación (utilizando el controlador incluido de BIND), se debe editar la línea del controlador *smtp* en el archivo *transports*. Un archivo de muestra *transports* se da en el Apéndice B.

14.11 Calificación de nombre de anfitrión

Algunas veces se desean capturar los nombres de sistema no calificados (i.e. aquellos que no tienen un nombre de dominio) escritos en la dirección del remitente o del receptor, por ejemplo cuando se pasa a través de dos redes, en donde una requiere de nombres de dominio totalmente calificados. En un relevo Internet-UUCP, los nombres de nodo no calificados

[7] Los autores llaman a este soporte "simple". Para una versión futura de *smail*, han anunciado un mecanismo completo que menejará esto de manera más eficiente.

[8] Sin embargo, el uso de rutas en Internet se desaconseja totalmente. En cambio, se deben utilizar nombres de dominio totalmente calificados.

deben ser mapeados al dominio **uucp** por defecto. Cualquier otro cambio de dirección distinto a los anteriores son cuestionables.

El archivo */usr/lib/smail/qualify* indica a *smail* qué nombres de dominios debe cambiar a qué nombres de nodo. Las líneas del archivo *qualify* consisten en el nombre del sistema comenzando en la columna uno, seguidos del nombre del dominio. Las líneas conteniendo un símbolo # como su primer caracter no blanco se consideran comentarios. Las líneas se buscan en el orden en el que aparecen.

Si no existe el archivo *qualify*, no se hace ninguna calificación de nombres de nodos.

Un nombre de anfitrión especial (*) indica que todos son nombres de nodos. Así, se puede habilitar un mapeo a todos los sistemas no mencionados antes en un dominio por defecto. Debe ser utilizado sólo en la última línea.

En la Cervecera Virtual, todos los sistemas han sido configurados para utilizar nombres de dominio totalmente calificados en las direcciones de los remitentes. Las direcciones de los receptores no calificadas se considera que están en el dominio **uucp**, de tal forma que sólo una línea en el archivo *qualify* es necesaria.

```
# /usr/lib/smail/qualify, cambiado por janet el 12 Feb 1994
#
*          uucp
```

Capítulo 15

Sendmail+IDA

15.1 Acerca del autor

Vince Skahan (**vince@victrola.wa.com**) ha estado administrando un gran número de sistemas unix desde 1987, y actualmente hace funcionar sendmail+IDA en aproximadamente 300 estaciones de trabajo unix para unos 2000 usuarios.

Admite haber perdido considerablemente el sueño editando unos cuantos ficheros *sendmail.cf* "por la fuerza bruta" antes de descubrir sendmail+IDA en 1990. Admite asimismo que aguarda ansiosamente la llegada de la primera versión en Perl de sendmail, para todavía mayor disfrute.

15.2 Reconocimientos

Gracias a Neil Rickert y Paul Pomes por la gran cantidad de ayuda proporcionada a lo largo de los años en lo que se refiere al cuidado y mantenimiento de sendmail+IDA y a Rich Braun por hacer el porte inicial a Linux. Las mayores gracias son de lejos para mi mujer Susan, por todo el apoyo en este y otros proyectos.

15.3 Introducción a Sendmail+IDA

Se dice que no se es un *verdadero* administrador de sistemas Unix hasta que se haya editado el archivo *sendmail.cf*. Se dice asimismo que se está loco si se intenta hacer 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 de referencia definitiva ocupa 792 páginas es suficiente para espantar justificadamente a cualquiera. (*Sendmail*, editado por O'Reilly and Associates)

Con sendmail+IDA es distinto. Se elimina la necesidad de editar el siempre críptico archivo *sendmail.cf*, permitiendo al administrador definir la configuración de las rutas y direcciones particulares de una máquina específica, por medio de archivos de apoyo relativamente sencillos de entender, llamados *tables*. Cambiar a sendmail+IDA puede ahorrarle muchas horas de trabajo y estrés.

En comparación con los demás agentes principales de transporte de correo, es probable que no haya nada que no se pueda hacer más rápida y fácilmente que con sendmail+IDA. Las actividades más comunes, necesarias para hacer funcionar sistemas Internet o UUCP usuales, pasan a ser tareas fáciles de llevar a cabo.

Configuraciones que normalmente serían extremadamente difíciles, son ahora simples de crear y mantener.

Cuando se escribió este manual, la versión actual de *sendmail5.67b+IDA1.5* estaba disponible por FTP anónimo en **vixen.cso.uiuc.edu**. Esta versión compila sin parches ni modificaciones bajo Linux.

Todos los archivos de configuración necesarios para poder compilar, instalar y hacer funcionar los fuentes de sendmail+IDA bajo Linux se hallan en el archivo *newspak-2.2.tar.gz*, disponible por FTP anónimo en **sunsite.unc.edu**, en el directorio */pub/Linux/system/Mail*.

15.4 Archivos de configuración — Preliminares

El sendmail tradicional se configura a través de un archivo de configuración de sistema (típicamente */etc/sendmail.cf* o */usr/lib/sendmail.cf*), que no se asemeja ni de lejos a cualquier otro lenguaje que haya podido ver antes. Editar el archivo *sendmail.cf* para proporcionar un comportamiento personalizado puede ser una experiencia humillante.

Sendmail+IDA hace que este suplicio sea algo del pasado, siendo todas las opciones de configuración controladas por ficheros con formato de listados (tables), con una sintaxis bastante fácil de comprender. Estas opciones son configuradas mediante el procesado de ciertos archivos de información, que son proporcionados con los fuentes, vía "Makefiles" que invocan a m4 (analizador de macros) o dbm (procesador de bases de datos).

El archivo *sendmail.cf* define únicamente el comportamiento por omisión del sistema. Virtualmente, todos los ajustes especiales se hacen a través de un número de tablas opcionales en vez de editar directamente el archivo *sendmail.cf*. La figura 15.1 muestra todas las tablas que utiliza *sendmail*.

mailertable	define un comportamiento especial para nodos o dominios remotos.
uucpxtable	fuerza a UUCP a entregar el correo a los nodos que están en formato DNS.
pathtable	define rutas de rebote UUCP a nodos o dominios remotos.
uucprelays	cortocircuita el camino *pathalias* a nodos remotos bien conocidos.
genericfrom	convierte direcciones internas a genéricas visibles para el mundo exterior.
xaliases	convierte direcciones genéricas de/a direcciones internas válidas.
decnetxtable	convierte direcciones RFC-822 a direcciones de tipo DECnet.

Figura 15.1: Archivos de apoyo de *sendmail*.

15.5 El archivo *sendmail.cf*

El archivo *sendmail.cf* que utiliza sendmail+IDA no se edita directamente, sino que se genera desde un archivo de configuración *m4* proporcionado por el administrador del sistema local. De aquí en adelante, siempre nos referiremos a él simplemente como *sendmail.m4*.

Este archivo contiene algunas definiciones y en otros casos simplemente apunta a las tablas en donde se lleva realmente a cabo el trabajo. En general, sólo es necesario especificar:

- las trayectorias y nombres de archivos utilizados en el sistema local.
- el o los nombres de los sistemas conocidos para propósitos e-mail.
- cuál será el gestor de correo por defecto deseado (y quizá también algún nodo inteligente de reenvío [1] de correo)

Hay una gran variedad de parámetros que pueden ser definidos para establecer el comportamiento del sistema local o para ir más allá del comportamiento precompilado. Estas opciones de configuración se identifican en el archivo *ida/cf/OPTIONS* del directorio fuente.

[1] N. del T.: estos nodos se conocen también como *relevos*, del inglés *relay*

Un archivo *sendmail.m4* para una configuración mínima (UUCP o SMTP confiando todo el correo externo a anfitriones inteligentes conectados directamente) puede ser tan escueto como 10 o 15 líneas de texto excluyendo los comentarios.

15.5.1 Un ejemplo del archivo *sendmail.m4*

A continuación se muestra un ejemplo del archivo *sendmail.m4* para **vstout** en la Cervecera Virtual. **vstout** utiliza SMTP para hablar con todos los anfitriones de la red local de la Cervecera, y envía todo el correo para otros destinos a **moria**, su nodo de reenvío de Internet, vía UUCP.

15.5.2 Parámetros de uso común en *sendmail.m4*

Algunas partes del archivo *sendmail.m4* son necesarias siempre; otras pueden ser ignoradas si se acepta la configuración por defecto. Las siguientes secciones describirán cada una de las partes del archivo ejemplo *sendmail.m4* con más detalle.

Partes que definen los directorios

```
dnl #define(LIBDIR,/usr/local/lib/mail)dnl  # el directorio en donde estan
    # los archivos de soporte
```

LIBDIR define el directorio en donde sendmail+IDA espera encontrar los archivos de configuración, las diversas tablas dbm, y definiciones especiales de índole local. En una típica distribución ejecutable, esto está ya compilado en el ejecutable de sendmail y no es necesario ponerlo explícitamente en el archivo sendmail.m4.

El ejemplo anterior tiene una línea inicial *dnl* que significa que esta línea es única y esencialmente un comentario informativo.

Para modificar la localización de los archivos de soporte a un lugar distinto, elimine el *dnl* inicial de la línea superior, y ajuste el directorio deseado, luego recompile y reinstale el archivo *sendmail.cf*.

Cómo definir un sistema de correo local (mailer)

```
define(LOCAL_MAILER_DEF, mailers.linux)dnl  # gestor de correo para entrega local
```

```
dnl #------------------ EJEMPLO DE UN ARCHIVO SENDMAIL.M4 ---------
dnl # (la cadena 'dnl' es la forma de escribir un comentario en m4)
dnl # en general usted no debera ignorar LIBDIR de las trayectorias compiladas
dnl #define(LIBDIR,/usr/local/lib/mail)dnl     # lugar de los arch. de soporte
define(LOCAL_MAILER_DEF, mailers.linux)dnl     # gestor de correo para la
       # entrega local

define(POSTMASTERBOUNCE)dnl                    # el gestor de correo obtiene los rebotes
define(PSEUDODOMAINS, BITNET UUCP)dnl          # no intente usar DNS
       # en estos casos
dnl #-----------------------------------------------------------------
dnl #
define(PSEUDONYMS, vstout.vbrew.com  vstout.UUCP vbrew.com)
dnl                                            # los nombres seran conocidos por
define(DEFAULT_HOST, vstout.vbrew.com)dnl      # nuestro nombre primario,
       # 'nombre' para el correo

define(UUCPNAME, vstout)dnl                    # nuestro nombre uucp
dnl #
dnl #-----------------------------------------------------------------
dnl #
define(UUCPNODES, |uuname|sort|uniq)dnl        # nuestros vecinos uucp
define(BANGIMPLIESUUCP)dnl                     # aseguran que el correo
define(BANGONLYUUCP)dnl                        # uucp sea tratado correctamente
define(RELAY_HOST, moria)dnl                   # nuestro sistema de
       # relevo inteligente

define(RELAY_MAILER, UUCP-A)dnl                # alcanzamos moria via uucp
dnl #
dnl #------------------------------------------------------------------------
dnl #
dnl # varias tablas de busqueda dbm
dnl #
define(ALIASES, LIBDIR/aliases)dnl            # alias del sistema
define(DOMAINTABLE, LIBDIR/domaintable)dnl    # distribucion de dominios
      # entre nodos

define(PATHTABLE, LIBDIR/pathtable)dnl        # base de datos de trayectorias
define(GENERICFROM, LIBDIR/generics)dnl       # directorio generico
      # de direcciones

define(MAILERTABLE, LIBDIR/mailertable)dnl    # gestores de correo por
      # nodo o dominio

define(UUCPXTABLE, LIBDIR/uucpxtable)dnl      # trayectorias a los nodos
      # que alimentamos

define(UUCPRELAYS, LIBDIR/uucprelays)dnl      # trayectorias de cortocircuito

dnl #
dnl #------------------------------------------------------------------------
dnl #
dnl # incluye el codigo 'real' que hace que todo funcione
dnl # (provisto con el codigo fuente)
dnl #
include(Sendmail.mc)dnl                        # LINEA INDISPENSABLE !!!
dnl #
dnl #------------ FIN DEL ARCHIVO EJEMPLO DE SENDMAIL.M4 -------
```

Figura 15.2: Un archivo muestra de *sendmail.m4* para **vstout**.

La mayor parte de los sistemas operativos tienen un programa encargado de la gestión de correo local. Los programas más comunes para la mayor parte de las variantes de Unix están ya compiladas en el ejecutable de sendmail.

En Linux, es necesario definir explícitamente el gestor local de correo correspondiente, ya que, en algunas distribuciones, puede no estar incluido. Esto se lleva a cabo especificando *LOCAL_MAILER_DEF* en el fichero *sendmail.m4*

Por ejemplo, para que el popular programa *deliver*[2] gestione este servicio, se debe especificar en *LOCAL_MAILER_DEF mailers.linux*.

El siguiente archivo deberá ser instalado como *mailers.linux* en el directorio al que apunta *LIBDIR*. Esto define explícitamente el programa *deliver*, como gestor de correo interno *Mlocal*; por lo que con los parámetros adecuados, *sendmail* se encargará de entregar correctamente el correo cuyo destino es el sistema local. A menos que se sea un experto de sendmail, es probable que no se desee modificar el siguiente ejemplo.

```
# -- /usr/local/lib/mail/mailers.linux --
#     (gestores de correo locales para su uso en Linux)
Mlocal, P=/usr/bin/deliver, F=SlsmFDMP, S=10, R=25/10, A=deliver $u
Mprog,  P=/bin/sh,       F=lsDFMeuP,   S=10, R=10, A=sh -c $u
```

Hay también una opción compilada por defecto para *deliver* en el archivo *sendmail.mc* incluida en el archivo *sendmail.cf*. Si se opta por ella, se debe evitar el uso del archivo mailers.linux y en cambio definir lo siguiente en el archivo *sendmail.m4*:

```
dnl --- (en sendmail.m4) ---
define(LOCAL_MAILER_DEF, DELIVER)dnl       # gestor de correo para entrega local
```

Desafortunadamente, *Sendmail.mc* asume que el programa deliver está instalado en */bin*, lo cual no es el caso con Slackware 1.1.1 (que lo instala en */usr/bin*). En este caso es necesario, ya sea engañarlo con un enlace simbólico o recompilar *deliver* a partir del código fuente para que resida en */bin*.

Gestión de correo rechazado

```
define(POSTMASTERBOUNCE)dnl   #  el correo rechazado ira dirigido
      #  al postmaster o administrador de correo.
```

[2]*deliver* fue escrito por Chip Salzenberg (**chip%tct@ateng.com**). Es parte de varias distribuciones de Linux y se puede encontrar en los sistemas de FTP anónimo más comunes como **ftp.uu.net**.

Muchos sistemas consideran importante asegurar que el correo que se envía y se recibe tenga un 100% de fiabilidad. Aún cuando es útil que se examinen los ficheros de registro *syslogd(8)*, en general, el administrador del correo necesitará ver las cabeceras del correo rechazado, de tal forma que pueda determinar si el correo no fue entregado debido a un error del usuario, o a un error de configuración en alguno de los sistemas involucrados.

La definición de *POSTMASTERBOUNCE* hace que se envíe una copia de cada mensaje rechazado a la persona que ha sido definida como **Postmaster** para el sistema.

Desafortunadamente, al definir este parámetro, también se incluirá el *texto* en el mensaje enviado al Postmaster, lo cual en potencia, podría inquietar a los usuarios de correo del sistema en cuanto a su intimidad se refiere.

Es conveniente que los postmasters de sistema se autodisciplinen (o lo hagan por la vía de medios técnicos a través de programitas del shell que borren el texto de los mensajes rechazados que ellos reciben) a no leer el correo que no está dirigido a ellos.

Asuntos relacionados con el servidor de nombres o Domain Name Service

```
define(PSEUDODOMAINS, BITNET UUCP)dnl       # no intente usar DNS aqui
```

Hay varias redes bien conocidas que son punto de referencia común en las direcciones de correo por razones históricas, pero que no son válidas a efectos DNS. El definir *PSEUDODOMAINS* evita intentos de búsqueda infructuosos por parte del DNS, que siempre resultarán fallidos.

Cómo definir los nombres por los que se conoce al sistema local

```
define(PSEUDONYMS, vstout.vbrew.com  vstout.UUCP vbrew.com)
dnl                                     # nombres por los cuales se nos conoce
define(DEFAULT_HOST, vstout.vbrew.com)dnl   # nuestro 'nombre' primario para el correo
```

Frecuentemente, los sistemas quieren ocultar su verdadera identidad, o servir como pasarelas de correo, o recibir y procesar correo dirigido a los nombres anteriores por los cuales se les conocían.

PSEUDONYMS especifica la lista de todos los nombres de sistema para los cuales el sistema local aceptará el correo.

DEFAULT_HOST especifica la dirección de sistema que aparecerá en los mensajes que se originan en el nodo local. Es importante que este parámetro sea ajustado a un valor válido o todo el correo de retorno no podrá ser entregado.

Temas relacionados con UUCP

```
define(UUCPNAME, vstout)dnl                 # nuestro nombre uucp
define(UUCPNODES, |uuname|sort|uniq)dnl     # nuestros vecinos uucp
define(BANGIMPLIESUUCP)dnl                  # asegurandonos que el correo
define(BANGONLYUUCP)dnl                     #  uucp sea tratado correctamente
```

Con frecuencia, los sistemas son conocidos por un nombre a efectos DNS y otro para propósitos de UUCP. *UUCPNAME* permite definir que aparezca un nombre de sistema distinto en los encabezados del correo enviado a través de UUCP.

UUCPNODES define las instrucciones que proporcionan como resultado una lista con las direcciones de sistemas con los cuales se está conectado directamente a través de conexiones UUCP.

BANGIMPLIESUUCP y *BANGONLYUUCP* aseguran que el correo direccionado con la sintaxis “bang” de UUCP sea tratado de acuerdo con el comportamiento de UUCP en vez de utilizar el DNS, más común hoy en día en Internet.

Sistemas de relevo y de correo

```
define(RELAY_HOST, moria)dnl                # nuestro sistema de relevo inteligente
define(RELAY_MAILER, UUCP-A)dnl             # alcanzamos moria a traves de UUCP
```

Muchos administradores de sistema no quieren molestarse en llevar a cabo todo el trabajo necesario para asegurar que su sistema sea capaz de encontrar todas las redes (y por supuesto otros sistemas) existentes en el mundo. En lugar de hacer esto, prefieren confiar todo el correo saliente a otro sistema reconocido como “inteligente”.

RELAY_HOST define el nombre UUCP del sistema vecino inteligente.

RELAY_MAILER define el gestor de correo utilizado para enviar los mensajes hacia dicho sistema.

Es importante hacer notar que el ajuste de esos parámetros redunda en que todo el correo de salida será redirigido a ese sistema remoto, lo cual afectará la carga de ese sistema. Es necesario asegurarse de obtener el consentimiento explícito del Administrador de correo del sistema remoto antes de configurar el nuestro para que utilice a otro como nodo de reenvío de correo a efectos generales.

Tablas de configuración variadas

```
define(ALIASES, LIBDIR/aliases)dnl          # alias del sistema
```

```
define(DOMAINTABLE, LIBDIR/domaintable)dnl  # maquinas bajo el dominio
define(PATHTABLE, LIBDIR/pathtable)dnl      # base de datos de los caminos
define(GENERICFROM, LIBDIR/generics)dnl     # direccion generica del remitente
define(MAILERTABLE, LIBDIR/mailertable)dnl  # gestores de correo por nodo o dominio
define(UUCPXTABLE, LIBDIR/uucpxtable)dnl    # caminos a los maquinas que surtimos
define(UUCPRELAYS, LIBDIR/uucprelays)dnl    # caminos de cortocircuito
```

Con estas macros, se puede cambiar la localización donde sendmail+IDA busca las diversas tablas dbm que definen el comportamiento "real" del sistema. Es aconsejable depositarlos en *LIBDIR*.

El archivo maestro *Sendmail.mc*

```
include(Sendmail.mc)dnl                     # LINEA INDISPENSABLE!!!
```

Los autores de sendmail+IDA proporcionan el archivo *Sendmail.mc* que contiene las verdaderas "tripas", que serán convertidas al archivo sendmail.cf. Periódicamente, se sacan nuevas versiones que corrigen errores en el código, o agregan funcionalidad sin necesidad de una nueva versión y la recompilación general de sendmail.

Es importante *no* editar este archivo.

Bueno, ¿entonces cuáles son las líneas indispensables?

Cuando no se están utilizando ninguna de las tablas dbm opcionales, sendmail+IDA entrega el correo vía el gestor de correo por omisión *DEFAULT_MAILER* (y posiblemente el sistema de reenvío *RELAY_HOST* y el gestor de correo de reenvío *RELAY_MAILER*) definidos en el archivo *sendmail.m4* utilizado para generar *sendmail.cf*. Es posible modificar fácilmente este comportamiento cambiando ciertas líneas en los archivos *domaintable* o *uucpxtable*.

Un sistema genérico que esté en Internet y se comunique por DNS, o uno que sea sólo UUCP y envíe todo su correo vía UUCP a través de un *RELAY_HOST* inteligente, probablemente no necesite de ninguna modificación en una "table" específica.

Virtualmente todos los sistemas deberían configurar el *DEFAULT_HOST* y las macros *PSEUDONYMS*, de tal modo que definan el nombre canónico de sistema, y alias, por los que es conocido, y su *DEFAULT_MAILER*. Si todo lo que se tiene es un nodo de reenvío y un gestor de correo de reenvío, no es necesario ajustar estos valores por defecto ya que trabajará automáticamente.

Los nodos UUCP probablemente necesiten ajustar su *UUCPNAME* a su nombre oficial UUCP. También es probable que ajusten su *RELAY_MAILER* y su *RELAY_HOST* los cuales habilitarán el encaminado de nodo inteligente a través de un reenvío de correo. El transporte del correo a emplear se define en *RELAY_MAILER* y normalmente es *UUCP-A* para sistemas UUCP.

Si su sistema es sólo SMTP y emplea 'Domain Name Service' o DNS, se podría cambiar el *DEFAULT_MAILER* a *TCP-A* y probablemente borrar las líneas *RELAY_MAILER* y *RELAY_HOST*.

15.6 Un viaje por las tablas de Sendmail+IDA

Sendmail+IDA proporciona varias tablas que permiten modificar el comportamiento por defecto de sendmail (especificado en el archivo *sendmail.m4*) y definir un comportamiento especial para situaciones singulares, sistemas remotos y redes. Estas tablas son luego procesadas con *dbm*, utilizando un Makefile que es parte de la distribución.

Muchos sistemas necesitarán algunas de estas tablas, otros ninguna. Si su sistema no precisa estas tablas, lo más sencillo es, probablemente, crearlas con longitud cero (con la instrucción *touch*) y utilizar el archivo Makefile por defecto localizado en *LIBDIR* en lugar de editar el Makefile en sí mismo.

15.6.1 *mailertable*

El archivo *mailertable* define un tratamiento especial para máquinas específicas o dominios que están basados en el nodo o nombre de la red remota. Se utiliza de forma frecuente en los sistemas Internet para seleccionar un nodo de reenvío de correo intermedio, o una *pasarela* a través de la cual alcanzar una red remota, y para especificar el protocolo en particular (UUCP o SMTP) que se utilizará. Los sistemas UUCP por lo general no necesitan este archivo.

El orden es importante: sendmail lee el archivo desde el principio hacia el fin, y procesa el mensaje de acuerdo con la primera regla que encuentra. Por tanto, lo lógico normalmente es poner las reglas más explícitas al comienzo del archivo y las más genéricas al final.

Supongamos que se quiere redirigir todo el correo para el departamento de Ciencias de la Computación de la Universidad Groucho Marx vía UUCP a un sistema de relevo, **ada**. Para hacer eso, se debe agregar una línea en *mailertable* como la siguiente:

```
# (in mailertable)
#
# redirige todo el correo para el dominio .cs.groucho.edu via UUCP a ada
UUCP-A,ada         .cs.groucho.edu
```

Suponga que se quiere redirigir todo el correo al dominio más grande **groucho.edu** para que vaya a otro sistema de relevo, **bighub**, para la resolución de sus direcciones y posterior entrega. La expansión de las líneas que van en el archivo *mailertable* sería la siguiente:

```
# (en mailertable)
#
# redirige todo el correo para el dominio cs.groucho.edu via UUCP a ada
UUCP-A,ada         .cs.groucho.edu
#
# redirige todo el correo para el dominio groucho.edu via UUCP a bighub
UUCP-A,bighub      .groucho.edu
```

Como se mencionó anteriormente, el orden es importante. Invertir el orden de las dos reglas mostradas anteriormente tendría como consecuencia que todo el correo dirigido a **.cs.groucho.edu** fuese a través del camino más genérico **bighub** en vez de utilizar la trayectoria explícita **ada** que es la que se quiere.

```
# (en mailertable)
#
# redirige todo el correo para el dominio .groucho.edu via UUCP a bighub
UUCP-A,bighub     .groucho.edu
#
# (es imposible alcanzar la siguiente linea porque
#    la norma que esta arriba sera cumplida primero)
UUCP-A,ada        .cs.groucho.edu
#
```

En los ejemplos de *mailertable* anteriores, el gestor de correo *UUCP-A* hace que *sendmail* utilice UUCP como medio de entrega con cabeceras "dominizadas".

La coma entre el gestor de correo y el sistema remoto indica que el mensaje se debe redirigir a **ada** para la resolución de su dirección y posterior entrega.

Las líneas que van en *mailertable* tienen el siguiente formato:

mailer delimitador sistema_de_relevo   *maquina_o_dominio*

Existen diferentes gestores de correo posibles. Las diferencias radican generalmente en cómo tratarán las direcciones. Los gestores de correo típicos son: *TCP-A* (TCP/IP con direcciones estilo Internet), *TCP-U* (TCP/IP con direcciones estilo UUCP), y *UUCP-A* (UUCP con direcciones estilo Internet).

El carácter que separa al gestor de correo de la porción del nodo en la parte izquierda de la línea de mailertable define cómo será modificada la dirección por la *mailertable*. Lo importante aquí es que únicamente se reescribe el "sobre" (para obtener el correo en el sistema remoto). Reescribir cualquier otra cosa más que el sobre es desaconsejable debido a la alta probabilidad de arruinar la configuración del correo.

! Un signo final de exclamación elimina el nombre del nodo receptor antes de redirigirlo al gestor de correo. Esto se puede usar cuando lo que se desea, esencialmente, es forzar al correo a entrar en un sistema remoto mal configurado.

, Una coma no cambia la dirección en modo alguno. El mensaje simplemente es redirigido vía el gestor de correo especificado al nodo o sistema de reenvío especificado.

: Dos puntos eliminan el nombre del sistema receptor si hay sistemas intermedios entre usted y el destino. Así, **foo!bar!joe** eliminará **foo**, mientras que **xyzzy!janet** permanecerá sin cambios.

15.6.2 *uucpxtable*

Es común que el correo dirigido a sistemas con nombres de dominio plenamente cualificados se entregue vía formato Internet (SMTP) utilizando un servidor de DNS, o mediante un sistema de reenvío. El archivo *uucpxtable* fuerza la entrega mediante encaminamiento UUCP, al convertir el nombre de formato dominio a un nombre de nodo con estilo UUCP sin formato de dominio.

Esto se utiliza frecuentemente cuando se es un "repetidor"[3] de correo para un sistema o dominio, o cuando se desea enviar el correo a través de un enlace UUCP directo y seguro en lugar de arriesgarse a pasar potencialmente por muchos "saltos"[4] si hacemos uso del gestor de correo por omisión y cualesquiera de los sistemas intermedios y redes.

Los sistemas UUCP que se comunican con sus vecinos UUCP, que utilizan cabeceras de correo dominizadas, podrían utilizar este archivo para forzar la entrega del correo, a través del enlace directo UUCP punto a punto entre los dos sistemas, en lugar de emplear la ruta menos directa a través del *RELAY_MAILER* y el *RELAY_HOST* o a través del *DEFAULT_MAILER*.

Los sistemas Internet que no empleen UUCP probablemente no utilicen el archivo *uucpxtable*.

[3]Un análogo de lo que se entiende por "repetidor" en telecomunicaciones.

[4]N. del T. Como *hop* o salto se entiende cada vez que atravesamos un sistema intermedio.

Supongamos que usted proporciona servicio de reenvío de correo a un sistema llamado **sesame.com** en DNS y **sesame** en los mapas UUCP. Necesitaría la siguiente línea en *uucpxtable* para forzar el direccionamiento del correo para ellos a través de nuestra conexión UUCP directa.

```
#============== /usr/local/lib/mail/uucpxtable ============
# El correo enviado a joe@sesame.com se reescribe a
#  sesame!joe y luego se entrega via UUCP
#
sesame          sesame.com
#
#----------------------------------------------------------
```

15.6.3 *pathtable*

El archivo *pathtable* se utiliza para definir el encaminamiento explícito a sistemas o redes remotas. El archivo *pathtable* debe escribirse en orden alfabético con una sintaxis similar al estilo de *pathalias*. Los dos campos de cada línea deben estar separados por un TAB real; si no es así *dbm* podría "protestar".

La mayor parte de los sistemas no precisarán ninguna línea en *pathtable*.

```
#=============== /usr/local/lib/mail/pathtable ================
#
# Este archivo tiene el estilo de pathalias en cuanto a trayectorias,
# y permite encauzar el correo dirigido a los vecinos UUCP a traves de un camino
# directo, de tal forma que no se tenga que hacer un rodeo hasta el
# nodo inteligente, que se encarga de otro trafico.
#
# Es deseable que se utilicen espacios de tabulacion reales en cada linea o
#  m4 podria quejarse.
#
# Se debe encaminar el correo a traves de uno o mas sistemas intermedios
# a un sistema remoto utilizando el estilo de direcciones UUCP.
#
sesame!ernie!%s           ernie
#
# reenviado a un sistema UUCP vecino de un sistema Internet
# alcanzable.
#
swim!%s@gcc.groucho.edu    swim
#
# Lo que sigue manda todo el correo para dos redes a traves de
```

```
# distintos gateways (observe el '.' que comienza la linea).
# En este ejemplo, "uugate" y "byte" son sistemas especificos que son
# utilizados como gateways de correo a los pseudo dominios .UUCP y
# .BITNET respectivamente.
#
%s@uugate.groucho.edu          .UUCP
byte!%s@mail.shift.com         .BITNET
#
#=================== fin de pathtable ======================
```

15.6.4 *domaintable*

El archivo *domaintable* se utiliza generalmente para forzar cierto comportamiento tras una búsqueda DNS. Permite al administrador hacer disponible una lista de abreviaturas de los nombres disponibles para sistemas o dominios a los que hagamos referencia con asiduidad, reemplazando la abreviatura con el nombre apropiado automáticamente. También puede ser utilizado para sustituir los nombres de un nodo o dominio incorrectos con la información correcta.

La mayor parte de los sistemas no necesitan líneas en *domaintable.*

El siguiente ejemplo muestra cómo reemplazar una dirección personal errónea, intentándose enviar con la correcta:

```
#============= /usr/local/lib/mail/domaintable =================
#
#
maquina_mal_configurada.dominio.correcto   maquina_mal_c.dominio.erroneo #
#
#=================== fin de domaintable ========================
```

15.6.5 *alias*

Los alias posibilitan lo siguiente:

- Permiten que una abreviatura o término fácil de recordar actúe como una dirección de correo, que remite lo recibido a una o varias personas.
- Invocan a un programa que tomará como entrada el mensaje.
- Envían correo a un archivo.

Todos los sistemas precisan alias para el **Postmaster** y el **MAILER-DAEMON** a fin de cumplir con el RFC.

Se debe ser extremadamente cuidadoso con respecto a la seguridad cuando se definan alias que invoquen a programas o escriban a programas ya que el sendmail generalmente se ejecuta con los permisos setuid-root.

Los cambios al archivo de *aliases* no tienen efecto hasta que el comando

```
# /usr/lib/sendmail -bi
```

se ejecuta para construir las tablas dbm necesarias. Esto también puede hacerse ejecutando el comando *newaliases*, normalmente mediante el comando *cron*.

Para más detalles concernientes a los alias de correo, se puede encontrar más información en la página man *aliases(5)*.

```
#-------------------- /usr/local/lib/mail/aliases ------------------
#
# muestra de tipos de alias comunes
#
usenet:         janet                    # alias para una persona
admin:          joe,janet                # alias para varias personas
newspak-users:  :include:/usr/lib/lists/newspak
  # lee los receptores de un archivo
changefeed:     | /usr/local/lib/gup     # alias que invoca un programa
complaints:     /var/log/complaints      # alias que escribe el
  # correo recibido a un archivo
#
# Los siguientes dos alias deben estar presentes para cumplir con el RFC.
# Es importante tenerlos para asignar a una persona que lea el correo
# rutinariamente.
#
postmaster:     root                     # linea indispensable
MAILER-DAEMON:  postmaster               # linea indispensable
#
#------------------------------------------------------------------
```

15.6.6 Tablas utilizadas en raras ocasiones

Las siguientes tablas están disponibles, pero se utilizan muy rara vez. Consulte la documentación que viene con el código fuente de sendmail+IDA para más detalles.

uucprelays El archivo *uucprelays* se utiliza para "corto-circuitar" la trayectoria del UUCP a sistemas especialmente bien conocidos en vez de utilizar una trayectoria multi-salto o insegura generada por el procesamiento de los mapas UUCP con *pathalias*.

genericfrom y *xaliases*
El archivo *genericfrom* oculta los nombres y direcciones de los usuarios locales del mundo exterior convirtiendo automáticamente los nombres de usuarios locales a direcciones genéricas de envío no coincidentes con los nombres internos de usuarios.

La utilidad asociada *xalparse* automatiza la generación de genericfrom y el archivo *aliases* de tal forma que las traducciones de los nombres del usuario de entrada y salida tengan lugar desde el archivo maestro *xaliases*.

decnetxtable El archivo *decnetxtable* reescribe las direcciones con formato dominio a direcciones estilo DECnet muy similares al archivo domaintable, que se utiliza para reescribir direcciones sin dominizar a direcciones estilo SMTP con formato dominizado.

15.7 Instalación de *sendmail*

En esta sección se verá cómo instalar una distribución ejecutable típica de sendmail+IDA y un recorrido por lo necesario para personalizarla y hacerla funcionar.

La distribución binaria actual de sendmail+IDA para Linux puede obtenerse de **sunsite.unc.edu** en */pub/Linux/system/Mail*. Incluso si se tiene una versión anterior de *sendmail* es muy recomendable utilizar la versión *sendmail5.67b+IDA1.5* ya que todos los parches específicos para Linux están en fuentes poco revisados, y varios e importantes agujeros de seguridad han sido enmendados (algunos de ellos datan del primero de diciembre de 1993).

Si se está compilando *sendmail* desde el código fuente, se deben seguir las instrucciones que están en los archivos *README* que están incluidos en la distribución de los fuentes. El código fuente actual de sendmail+IDA está disponible en **vixen.cso.uiuc.edu**. Para construir sendmail+IDA en Linux, también se necesitan los archivos de configuración especiales para Linux *newspak-2.2.tar.gz* que están en **sunsite.unc.edu** en el directorio */pub/Linux/system/Mail*.

Si tenía instalado anteriormente *smail* u otro gestor de entrega de correo, probablemente quiera borrar o renombrar todos los ficheros pertenecientes a smail para mayor seguridad.

15.7.1 Desempaquetado de la distribución ejecutable

Lo primero es desempaquetar el archivo comprimido en algún lugar seguro:

```
$ gunzip -c sendmail5.65b+IDA1.5+mailx5.3b.tgz | tar xvf -
```

Si se tiene un *tar* "moderno", por ejemplo de una distribuición de Slackware reciente, probablemente baste con un `tar -zxvf` *`fichero`*`.tgz` y se obtendrán los mismos resultados.

Al despempaquetar el archivo se genera un directorio llamado *sendmail5.65b+IDA1.5+mailx5.3b*. En este directorio encontrará la instalación completa de sendmail+IDA más un programa binario del agente para usuario *mailx*. Todos los directorios donde se encuentran los archivos reflejan la ubicación donde deben ser instalados éstos, así que es más seguro utilizar la aplicación *tar* para moverlos a otra parte:

```
# cd sendmail5.65b+IDA1.5+mailx5.3b
# tar cf - . | (cd /; tar xvvpoof -)
```

15.7.2 Elaboración del fichero *sendmail.cf*

Para elaborar un fichero *sendmail.cf* personalizado para su sistema, se ha de escribir un fichero *sendmail.m4*, y procesarlo posteriormente con *m4*.

En */usr/local/lib/mail/CF* puede encontrar un archivo de ejemplo llamado *sample.m4*. Cópielo a *`nombredesusistema`*.*m4*, y edítelo a fin de que refleje la situación de su sistema.

El fichero de ejemplo está configurado para un sistema sólo UUCP con cabeceras dominizadas y que se comunica con un sistema inteligente. Los sistemas como éste precisan de pocas variaciones.

En esta sección se señalaran las macros a cambiar. Si quiere tener una descripción completa de lo que hacen, diríjase a la sección anterior, "Discusión del fichero *sendmail.m4*".

LOCAL_MAILER_DEF
: Define el fichero que especifica los agentes de correo para gestión local. Vea la sección previa "Definición del gestor local de correo" para saber de qué va.

PSEUDONYMS
: Especifica todos los nombres por los que es conocido su sistema.

DEFAULT_HOST
: Escriba su nomenclatura de dominio plenamente cualificado. Este nombre aparecerá como su nombre de sistema en todo el correo saliente.

UUCPNAME
: Ponga su nombre de sistema sin cualificar.

RELAY_HOST y *RELAY_MAILER*
: Si se comunica mediante UUCP con un sistema inteligente, defina *RELAY_HOST* como el nombre UUCP del "repetidor inteligente" de su vecino UUCP. Haga uso del gestor de correo UUCP-A si desea que las cabeceras de sus mensajes contengan su dominio.

DEFAULT_MAILER
: Si está conectado a Internet y se comunica mediante DNS, debería definir esto como *TCP-A*. Esto le dice a sendmail que emplee *TCP-A* como gestor de correo, que entregará el correo via SMTP haciendo uso del estilo RFC normal en las direcciones de los receptores de correo. Los sistemas conectados permanentemente a Internet probablemente no precisen definir *RELAY_HOST* o RELAY_MAILER.

Para crear el fichero *sendmail.cf*, ejecutar la orden

```
# make nombredesusistema.cf
```

Esto procesa el fichero `nombredesusistema` y crea el fichero `nombredesusistema`*.cf* a partir de él.

Lo próximo será comprobar si el fichero que acaba de crear hace lo que se espera de él o no. Esto se explica en las próximas dos secciones.

Una vez se está contento con su comportamiento, cópielo en su sitio con el comando:

```
# cp nombredesusistema.cf /etc/sendmail.cf
```

Llegados a este punto, su sistema sendmail está listo para funcionar. Escriba la siguiente línea en el fichero de arranque adecuado (generalmente */etc/rc.inet2*). Puede ejecutarlo "a mano" para que empiece a funcionar en este momento.

```
# /usr/lib/sendmail -bd -q1h
```

15.7.3 Comprobando el fichero *sendmail.cf*

Para hacer que sendmail funcione en modo 'test', ha de ejecutarlo con la opción `-bt`. La configuración por defecto es el fichero sendmail.cf que esté instalado en el sistema. Puede probar un fichero de configuración alternativo mediante la opción `-Cfichero_alternativo`.

En los siguientes ejemplos, probamos *vstout.cf*, el fichero de configuración generado a partir del fichero *vstout.m4* que puede ser examinado en la figura 15.2.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
>
```

Las siguientes comprobaciones aseguran que *sendmail* es capaz de gestionar el correo de todos los usuarios del sistema. En todos los casos, el resultado de la comprobación deberá ser el mismo, y apuntar al nombre del sistema local como el gestor de correo en *LOCAL*.

Comprobemos primero cómo se gestionaría el envío a un usuario local:

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 me
rewrite: ruleset  3   input: me
rewrite: ruleset  7   input: me
rewrite: ruleset  9   input: me
rewrite: ruleset  9 returns: < me >
rewrite: ruleset  7 returns: < > , me
rewrite: ruleset  3 returns: < > , me
rewrite: ruleset  0   input: < > , me
rewrite: ruleset  8   input: < > , me
rewrite: ruleset 20   input: < > , me
rewrite: ruleset 20 returns: < > , @ vstout . vbrew . com , me
rewrite: ruleset  8 returns: < > , @ vstout . vbrew . com , me
rewrite: ruleset 26   input: < > , @ vstout . vbrew . com , me
rewrite: ruleset 26 returns: $# LOCAL $@ vstout . vbrew . com $: me
rewrite: ruleset  0 returns: $# LOCAL $@ vstout . vbrew . com $: me
```

El resultado muestra cómo *sendmail* procesa las direcciones internamente. Esto es llevado a cabo por varias *rulesets* que las analizan, llaman a otras involucradas, y descomponen la dirección en sus componentes.

En nuestro ejemplo, le pasamos la dirección **me** a las rulesets 3 y 0 (esto es lo que significa el término `3,0` introducido antes de la dirección).

La última línea muestra la dirección interpretada tal y como la devuelve la ruleset 0, que contiene el gestor de correo al que se le encomendaría el mensaje, y la máquina y usuario proporcionados al mismo.

A continuación, comprobaremos el envío de correo a un usuario de nuestro sistema con sintaxis UUCP.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 vstout!me
rewrite: ruleset  3   input: vstout ! me
[...]
rewrite: ruleset  0 returns: $# LOCAL $@ vstout . vbrew . com  $: me
>
```

A continuación, comprobamos el correo dirigido a un usuario de nuestro sistema con sintaxis Internet, a nuestro nombre de sistema plenamente cualificado (FQDN)

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 me@vstout.vbrew.com
rewrite: ruleset  3   input: me @ vstout . vbrew . com
[...]
rewrite: ruleset  0 returns: $# LOCAL $@ vstout . vbrew . com $: me
>
```

Debería repetir los anteriores dos pasos con cada uno de los nombres especificados como parámetros *PSEUDONYMS* y *DEFAULT_NAME* del fichero *sendmail.m4*.

Por último, comprobar que puede enviar correo a su nodo de reenvío.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
```

```
[Note: No initial ruleset 3 call]
> 3,0 fred@moria.com
rewrite: ruleset  3   input: fred @ moria . com
rewrite: ruleset  7   input: fred @ moria . com
rewrite: ruleset  9   input: fred @ moria . com
rewrite: ruleset  9 returns: < fred > @ moria . com
rewrite: ruleset  7 returns: < @ moria . com > , fred
rewrite: ruleset  3 returns: < @ moria . com > , fred
rewrite: ruleset  0   input: < @ moria . com > , fred
rewrite: ruleset  8   input: < @ moria . com > , fred
rewrite: ruleset  8 returns: < @ moria . com > , fred
rewrite: ruleset 29   input: < @ moria . com > , fred
rewrite: ruleset 29 returns: < @ moria . com > , fred
rewrite: ruleset 26   input: < @ moria . com > , fred
rewrite: ruleset 25   input: < @ moria . com > , fred
rewrite: ruleset 25 returns: < @ moria . com > , fred
rewrite: ruleset  4   input: < @ moria . com > , fred
rewrite: ruleset  4 returns: fred @ moria . com
rewrite: ruleset 26 returns: < @ moria . com > , fred
rewrite: ruleset  0 returns: $# UUCP-A $@ moria $: < @ moria . com > , fred
>
```

15.7.4 Integración global - Prueba de integración del fichero *sendmail.cf* y las tablas.

Llegados a este punto, ya ha verificado que el sistema de correo tendrá el comportamiento por defecto deseado, y que será capaz tanto de enviar como de recibir correo con dirección válida. Para terminar la instalación, puede ser necesario crear las tablas dbm apropiadas para conseguir finalmente los resultados deseados.

Tras crear las tablas necesarias para su sistema, deberá procesarlas a través de *dbm* mediante la ejecución de la orden *make* en el directorio que contenga las tablas.

Si su sistema es sólo UUCP, *no* necesita crear ninguna de las tablas mencionadas en el fichero *README.linux*. Sólo tendrá que modificar los ficheros de tal modo que funcione el *Makefile*.

Si su sistema es sólo UUCP y "habla" con más sistemas además de su nodo de reenvío inteligente, necesitará añadir entradas *uucpxtable* para cada uno (o el correo destinado a ellos se encaminará también a través del nodo inteligente) y ejecutar *dbm* sobre el recién modificado fichero *uucpxtable*.

Para empezar, necesita asegurarse de que el correo que ha de pasar por su *RELAY_HOST* se envía mediante el *RELAY_MAILER*.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 fred@sesame.com
rewrite: ruleset  3   input: fred @ sesame . com
rewrite: ruleset  7   input: fred @ sesame . com
rewrite: ruleset  9   input: fred @ sesame . com
rewrite: ruleset  9 returns: < fred > @ sesame . com
rewrite: ruleset  7 returns: < @ sesame . com > , fred
rewrite: ruleset  3 returns: < @ sesame . com > , fred
rewrite: ruleset  0   input: < @ sesame . com > , fred
rewrite: ruleset  8   input: < @ sesame . com > , fred
rewrite: ruleset  8 returns: < @ sesame . com > , fred
rewrite: ruleset 29   input: < @ sesame . com > , fred
rewrite: ruleset 29 returns: < @ sesame . com > , fred
rewrite: ruleset 26   input: < @ sesame . com > , fred
rewrite: ruleset 25   input: < @ sesame . com > , fred
rewrite: ruleset 25 returns: < @ sesame . com > , fred
rewrite: ruleset  4   input: < @ sesame . com > , fred
rewrite: ruleset  4 returns: fred @ sesame . com
rewrite: ruleset 26 returns: < @ sesame . com > , fred
rewrite: ruleset  0 returns: $# UUCP-A $@ moria $: < @ sesame . com > , fred
>
```

Si tiene más vecinos UUCP, además de su *RELAY_HOST*, necesita asegurarse de que el correo para ellos experimenta un procesamiento adecuado. El correo con direcciones de sintaxis tipo UUCP dirigido a otro sistema con el que se comunique también mediante UUCP, irá a ellos directamente (a menos de que lo impida explícitamente mediante una entrada *domaintable*). Asumimos que el sistema **swim** es un vecino UUCP directo para nosotros. Pasar a *sendmail* un mensaje **swim!fred** deberá producir el siguiente resultado:

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 swim!fred
rewrite: ruleset  3   input: swim ! fred
[...lines omitted...]
rewrite: ruleset  0 returns: $# UUCP $@ swim $: < > , fred
```

```
>
```

Si tiene entradas *uucpxtable* para forzar la gestión de correo UUCP a ciertos vecinos UUCP que envían su correo con cabeceras dominizadas tipo Internet, tambíen tiene que verificarlo.

```
# /usr/lib/sendmail -bt -Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 dude@swim.2birds.com
rewrite: ruleset  3   input: dude @ swim . 2birds . com
[...lines omitted...]
rewrite: ruleset  0 returns: $# UUCP $@ swim . 2birds $: < > , dude
>
```

15.8 Trucos y trivialidades sobre administración de correo

Ahora que ya se ha discutido la teoría sobre configuración, instalación y comprobación de un sistema sendmail+IDA, dediquemos unos instantes al análisis de las cosas que *suceden* rutinariamente en la vida de un administrador de correo.

Los sistemas remotos fallan a veces. Los modems o líneas telefónicas fallan o las definiciones DNS son elaboradas incorrectamente debido a un error humano. En estos casos, los administradores de correo han de saber cómo reaccionar de forma rápida, efectiva y *segura* para mantener el tráfico del correo a través de rutas alternativas hasta que los sistemas remotos o los proveedores de acceso puedan restablecer sus servicios habituales.

El resto de este capítulo pretende proporcionarle soluciones para las "emergencias con el correo electrónico" más frecuentes.

15.8.1 Reenvío de correo a un sistema inteligente

Para redirigir el correo para un sistema o dominio particular hacia el sistema de reenvío inteligente designado, se empleará normalmente el fichero *mailertable*.

Por ejemplo, para redirigir el correo para **backwood.org** a su sistema de pasarela UUCP **backdoor**, tendrá que poner la siguiente entrada en *mailertable*:

```
UUCP-A,backdoor   backwood.org
```

15.8.2 Envío de correo a Sistemas Remotos mal configurados

Los sistemas Internet tendrán frecuentemente problemas a la hora de hacer entrar el correo en sistemas mal configurados. Existen varios casos, pero el síntoma general es que el correo es devuelto por el sistema remoto o que nunca lo alcanza.

Estos problemas pueden colocar al administrador local del sistema en una situación crítica, ya que sus usuarios generalmente no tienen en cuenta que usted no administra todos los sistemas a lo largo y ancho del mundo (o que usted no sepa cómo hacer que el administrador remoto solucione el problema). Ellos tan sólo sabrán que su correo no llegó al destinatario deseado en el otro extremo, y usted será la persona más cercana a la que pedir responsabilidades.

La configuración de un sistema remoto es problema de sus administradores, no de usted. En cualquier caso, asegúrese de no estropear la configuración de su sistema a fin de comunicarse con un sistema remoto mal configurado. Si no puede ponerse en contacto con el administrador (*Postmaster*) del sistema remoto a fin de que arreglen su configuración lo antes posible, tiene dos opciones:

- Generalmente es posible forzar el correo hacia el interior del sistema remoto con éxito, aunque el sistema remoto esté mal configurado; las respuestas provenientes del otro extremo posiblemente no funcionen... pero ese es problema del administrador remoto.

  Puede corregir las cabeceras erróneas de sus destinatarios de correo saliente simplemente usando una entrada *domaintable* para su sistema/dominio, lo que redundará en que la información no válida sea corregida en el correo originado desde su sistema:

  ```
  descerebrado.dominio.correcto.com       descerebrado.dominio.erroneo.com
  ```

- Los sistemas mal configurados devuelven con frecuencia el correo al sistema que lo originó, argumentando que "este correo no es para este sistema" ya que no tienen debidamente configurado su *PSEUDONYMNS* o equivalente. Es posible quitar toda información relativa al nombre y dominio del sistema en los destinatarios de correo saliente de nuestro sistema hacia ellos.

  El *!* de la siguiente *mailertable* gestiona el correo hacia su sistema remoto

  ```
  TCP!descerebrados.dominio.correcto.com   descerebrados.dominio.erroneo.com
  ```

No obstante, y aunque se consiga que el correo entre en su sistema, no hay garantías de que ellos puedan responder a nuestros mensajes (su sistema está mal configurado, recuérdelo...) pero para entonces sus usuarios estarán quejándose a sus administradores, que es mejor que los suyos se enfaden con usted.

15.8.3 Envío Forzado de correo a través de UUCP

En un mundo ideal (desde la perspectiva Internet), todas las máquinas tendrán registro en el Servicio de Nombres de Dominio (DNS) y envían su correo con nombres de dominio plenamente cualificados.

Si se da la circunstancia de que se comunica vía UUCP con un sistema de estas características, puede forzar el correo a ser enviado directamente a través de la conexión punto-a-punto UUCP en lugar de hacerlo a través de su gestor de correo habitual, esencialmente "desdominizando" su nombre de sistema mediante el fichero *uucpxtable*.

Para forzar el envío de correo a la máquina **sesame.com**, deberá poner lo siguiente en el fichero *uucpxtable*:

```
# desdominizamos sesame.com para forzar el envio UUCP
sesame    sesame.com
```

El resultado es que sendmail determinará entonces (a través de *UUCPNODES* del fichero *sendmail.m4*) que se está conectado directamente al sistema remoto, y encolará el correo saliente para ser enviado vía UUCP.

15.8.4 Prevención de que el correo sea enviado vía UUCP

La condición contraria también se da. Con frecuencia, los sistemas tienen cierto número de conexiones UUCP que rara vez se emplean o que no siempre son tan fiables, o que no están tan disponibles como el gestor de correo por defecto o el sistema de reenvío.

Por ejemplo, en el área de Seattle hay varios sistemas que intercambian las distintas distribuciones Linux vía UUCP anónimo conforme se van liberando las distribuciones. Estos sistemas se comunican mediante UUCP sólo cuando es necesario, por lo que es generalmente más rápido y fiable enviar el correo a través de múltiples saltos muy fiables y nodos de reenvío (que siempre están disponibles).

Se puede evitar fácilmente el envío directo de correo a una máquina a la que se está directamente conectado. Si el sistema remoto posee un nombre de dominio plenamente cualificado, se puede añadir una entrada como ésta en el fichero *domaintable*:

```
# Evitamos que se envie el correo via UUCP a un sistema vecino
 snorkel.com       snorkel
```

Esto reemplazará cualquier aparición del nombre UUCP con el nombre FQDN, impidiendo por tanto cualquier concordancia con la línea *UUCPNODES* del fichero *sendmail.m4*. El resultado es, generalmente, que el correo se enviará vía *RELAY_MAILER* y *RELAY_HOST* (o *DEFAULT_MAILER*).

15.8.5 Procesado de la cola de correo a voluntad

Para procesar los mensajes de la cola de correo saliente inmediatamente, no hay más que teclear '/usr/lib/runq'[5]. Esto llamará a sendmail con las opciones apropiadas para hacer que procese inmediatamente la cola de procesos pendientes en lugar de esperar al próximo procesamiento programado.

15.8.6 Informe sobre las estadísticas de correo

Muchos administradores de sistema (y las personas para las que trabajan) están interesados en el volumen de correo que es enviado, recibido o que pasa a través de nuestro sistema. Hay varios métodos de cuantificar el tráfico de correo.

- El paquete sendmail incorpora una utilidad llamada *mailstats* que lee un fichero llamado */usr/local/lib/mail/sendmail.st* [6]e informa sobre el número de mensajes y bytes transferidos por cada uno de los gestores de correo que se empleen y que aparezcan en el fichero *sendmail.st*. Este fichero debe ser creado manualmente por el administrador local para que el registro tenga lugar por parte de sendmail. Los totales se reinicializan borrando y volviendo a crear el fichero *sendmail.st*. Un método para hacer esto es el siguiente;

```
# cp /dev/null /usr/lib/local/mail/sendmail.st
```

- Probablemente la mejor forma de obtener informes de calidad acerca de quién usa el correo y la cantidad de volumen que pasa hacia, por, y a través del sistema local sea activar el depurado de correo (*debugging*) mediante el uso de *syslogd(8)*. Esto generalmente conlleva el tener que arrancar el demonio *syslogd* desde su fichero de inicialización del sistema (de todos modos lo debería estar haciendo), y añadir una línea al fichero */etc/syslog.conf(5)* que tiene el siguiente aspecto:

```
mail.debug                          /var/log/syslog.mail
```

[5]La llamada a sendmail con el parámetro '-q' tiene idénticos efectos ('sendmail -q')

[6]N. del T.: En ciertas distribuciones actuales, como por ejemplo *RedHat*, la localización es */var/log/sendmail.st*; esto dependerá de la filosofía de la distribución que emplee; el *LFS* (Linux Filesystem Standards, anterior FSSTND) al ser un fichero de *log* o de registro, recomienda el directorio */var/log*

Si emplea *mail.debug*, y recibe un volumen de correo medio/alto, el resultado proporcionado por syslog puede hacerse bastante grande. Los ficheros de registro generados por *syslogd* necesitan generalmente ser purgados rutinariamente por *crond(8)*.

Existen cierto número de utilidades disponibles comúnmente que pueden resumir el resultado del registro de correo procedente de syslogd. Una de las más conocidas es *syslog-stat.pl*, un script en *Perl* que se distribuye con los fuentes de sendmail+IDA.

15.9 Integración y puesta a punto de Distribuciones Ejecutables

A pesar de que el Estándar de Sistema de Ficheros de Linux está en desarrollo, todavía no está ni terminado ni aceptado universalmente. Mi intención aquí es mostrar que todavía no somos[7] un estándar, y proporcionar una idea de cuáles son los lugares donde aparecen problemas con mayor frecuencia

No existe ninguna configuración auténticamente estándar del transporte de correo electrónico y sus agentes, así como no hay una "única estructura de directorios."

De acuerdo con esto, es necesario asegurarse de que todas las distintas partes del sistema (USENET news, mail, TCP/IP) están de acuerdo con la localización del gestor de correo local (*lmail*, *deliver*, etc.), el gestor de correo remoto (*rmail*), y el programa de transporte de correo (*sendmail* o *smail*). Estas suposiciones generalmente no están documentadas; no obstante, el uso del comando *strings* puede ayudarnos a determinar qué ficheros y directorios son los esperados. A continuación vienen algunos problemas que hemos observado en el pasado con algunas de las distribuciones ejecutables y fuentes disponibles comúnmente para Linux.

- Algunas versiones de la distribución de NET-2 de TCP/IP tienen servicios definidos para un programa llamado *umail* en lugar de para *sendmail*.
- Existen varios portes de *elm* y *mailx* que buscan al gestor de correo (envío) */usr/bin/smail* en lugar de a sendmail.
- Sendmail+IDA tiene un gestor de correo local interno para *deliver*, pero espera que esté en */bin* en lugar de la localización más típica en Linux */usr/bin*.

En lugar de pasar por la trabajosa tarea de compilar todos los clientes de correo a partir de sus fuentes, generalmente los engañaremos con los enlaces simbólicos apropiados.

[7]N. del T.: Esto ha cambiado desde que esta guía fue escrita, el *LFS* (Linux Filesystem Standards) o anterior *FSSTND* está en vías de ser aceptado, si es que no lo está ya.

15.10 Dónde obtener más información

Existen muchos lugares donde buscar más información sobre *sendmail*. Si se quiere un listado completo, vea el "*Linux MAIL Howto*", que se envía regularmente a **comp.answers**. También está disponible por FTP en **rtfm.mit.edu**. De todos modos, el lugar definitivo son los fuentes de sendmail+IDA. Busque en el directorio *ida/cf* que cuelga del directorio de los fuentes, los ficheros *DBM-GUIDE*, *OPTIONS*, y *Sendmail.mc*.

Capítulo 16

Netnews

16.1 Historia de Usenet

La idea de las noticias en red nació en 1979, cuando dos estudiantes de graduado, Tom Truscott y Jim Ellis, pensaron en usar UUCP para conectar ordenadores con el propósito de intercambiar información entre usuarios de UN⋆X. Instalaron una pequeña red de tres ordenadores en Carolina del Norte.

Inicialmente el tráfico de información era manejado por cierto número de *shell scripts* (más tarde reescritos en C), pero que nunca fueron hechos públicos. Fueron rápidamente reemplazados por "A" news, la primera edición pública de programas para *news*.

"A" news no estaba diseñado para manejar más que unos pocos artículos por grupo y día. Cuando el volumen de información continuó creciendo, fue reescrito por Mark Horton y Matt Glickman, quienes lo denominaron la versión "B" (también conocido como Bnews). El primer lanzamiento público de Bnews fue la versión 2.1, en 1982. Se fue expandiendo continuamente, conforme se le añadían nuevas prestaciones. La versión actual es Bnews 2.11. Poco a poco se va quedando obsoleta, habiéndose pasado a INN su último mantenedor oficial.

Geoff Collyer y Henry Spencer reescribieron y lanzaron en 1987 otra nueva versión, conocida como versión "C" o Cnews. En el tiempo transcurrido desde entonces ha habido algunos parches para Cnews, siendo el más notable de ellos la Cnews Performance Release. En sistemas que transportan un gran número de grupos, el consumo de recursos producido al ejecutar frecuentemente *relaynews* (el programa encargado de procesar los artículos) es bastante significativo. La Performance Release añade una opción que permite ejecutar *relaynews* en *modo daemon*, es decir, ejecutándose como tarea de fondo.

La Performance Release es la versión de Cnews que se incluye en la mayoría de las distribuciones de Linux actuales.

Todas las versiones hasta la "C" están principalmente diseñadas para utilizarse en redes UUCP, aunque igualmente pueden utilizarse en otros entornos. La transferencia eficiente de noticias sobre redes tipo TCP/IP, DECNet o similares, requiere otro planteamiento. Esta es la razón por la que en 1986 se introdujo el "Network News Transfer Protocol" (NNTP) o *Protocolo de Transferencia de Noticias a través de la Red.* Este protocolo está basado en conexiones de red, y especifica cierto número de comandos para transferir los artículos de forma interactiva.

Hay bastantes aplicaciones basadas en el NNTP disponibles en la Red. Una de ellas es el paquete *nntpd*, de Brian Barber y Phil Lapsley, que puede usarse, entre otras cosas, para proporcionar un servicio de lectura de noticias a distintos nodos de una red local. *nntpd* fue diseñado para complementarse con Bnews o Cnews y darles prestaciones NNTP.

Otra aplicación NNTP diferente es INN, o Internet News. No es simplemente un interfaz, sino un sistema de noticias por derecho propio. Consta de un sofisticado demonio de noticias que es capaz de mantener varias conexiones NNTP simultáneas, y es por lo tanto, el software elegido por muchos servidores en Internet.

16.2 ¿Qué *es*, en definitiva, Usenet?

Una de las cosas más asombrosas de Usenet es que no forma parte de ninguna organización, ni tiene ninguna clase de autoridad central. De hecho, parte del saber popular de Usenet consiste en que excepto por una descripción técnica, no se puede definir *qué* es, tan sólo qué no es. Si tiene Vd. a mano el excelente "Zen and the Art of the Internet" (disponible en Internet o a través de Prentice-Hall, ver [Kehoe92]), de Brendan Kehoe, encontrará una sorprendente lista de *impropiedades* de Usenet.

A riesgo de sonar tonto, podría definirse Usenet como la colaboración de servidores separados que intercambian noticias de Usenet. Para ser un servidor en Usenet, todo lo que hay que hacer es encontrar otro servidor Usenet y llegar a un acuerdo con sus propietarios y administradores para intercambiar noticias con ellos. Proporcionar artículos a otro servidor se denomina también *alimentación* o *feeding*, de dónde se origina otro axioma común de Usenet: "Consigue alguien que te pase las noticias, y ya eres parte de Usenet".

La unidad fundamental de las noticias de Usenet es el artículo. Es un mensaje que un usuario escribe y "publica" en la red. Para posibilitar que los sistemas de noticias lo manejen, está precedido de información administrativa, conocida como cabecera del artículo. Es muy similar a la cabecera utilizada para el correo que se describe en el estándar RFC 822, y como ésta, consiste en varias líneas de texto, cada una de las cuales comienza con el nombre de un campo terminado en dos puntos, siguiendo después el valor de dicho campo.[1]

[1]El formato de los mensajes de noticias de Usenet se especifica en la RFC 1036, "Standard for interchange

Los artículos son enviados a uno o más *grupos de noticias*. Podrían considerarse a los grupos como foros para artículos relativos a una misma temática. Todos los grupos están organizados en una jerarquía, en la cual el nombre de cada grupo indica su lugar en la misma. Esto a menudo hace más fácil ver sobre qué versa un grupo de noticias. Por ejemplo, todo el mundo puede deducir por el nombre que **comp.os.linux.announce** se usa para anuncios relativos a un sistema operativo para computadoras llamado Linux.

Estos artículos son intercambiados entre todos los servidores de Usenet a los que interese tener noticias de este grupo. Cuando dos servidores acuerdan intercambiar noticias, son libres de intercambiar cualquier grupo que deseen, y pueden incluso añadir sus propias jerarquías locales. Por ejemplo, **groucho.edu** puede tener un enlace de noticias con **barnyard.edu**, un gran servidor de noticias, y varios enlaces con servidores menores a los que *alimenta* con noticias. El Colegio Barnyard puede recibir todos los grupos de Usenet, mientras que la UGM sólo quiere algunas jerarquías mayores como **sci**, **comp**, **rec**, etc. Algunos servidores situados más abajo en esta cadena, digamos un servidor UUCP llamado **brewhq**, querrán incluso menos grupos, ya que no tendrán los suficientes recursos de hardware o de red. Por otro lado, **brewhq** puede querer recibir grupos de la jerarquía **fj** que la UGM no tiene. Por lo tanto, mantiene otro enlace con **gargleblaster.com**, quien tiene todos los grupos **fj** y se los pasa a **brewhq**. El flujo de noticias se muestra en la figura 16.1.

Las etiquetas en las flechas que parten de **brewhq** pueden requerir ciertas explicaciones. Por defecto, **brewhq** quiere que todas las noticias generadas localmente sean enviadas a **groucho.edu**. Sin embargo, ya que **groucho.edu** no lleva los grupos **fj**, no hay razón para enviar ningún artículo de estos grupos. Por tanto, la alimentación de **brewhq** a la UGM está etiquetada **all,!fj**, lo que significa que se envían todos los grupos excepto los **fj**.

16.3 ¿Cómo maneja Usenet las noticias?

Hoy en día, Usenet ha crecido hasta alcanzar dimensiones enormes. Los servidores que llevan la totalidad de los grupos suelen tener que transferir unos sesenta megabytes por día.[2] Por supuesto esto requiere mucho más que enredar con unos cuantos ficheros. Veamos cómo se las apañan la mayoría de sistemas UN⋆X para manejar las noticias.

Las noticias se distribuyen por la red de varias formas. El medio histórico solía ser UUCP, pero hoy en día el caudal principal es llevado por servidores permanentemente conectados a Internet. El algoritmo para encaminar se denomina *inundación*: cada servidor mantiene cierto número de enlaces con otros servidores. Cualquier artículo generado o

of USENET messages".

[2]Un momento: 60 Mb a 9600 bps, o sea, 60 millones por 1200, eso es... murmullo, murmullo,... ¡Eh! ¡Son 34 horas!

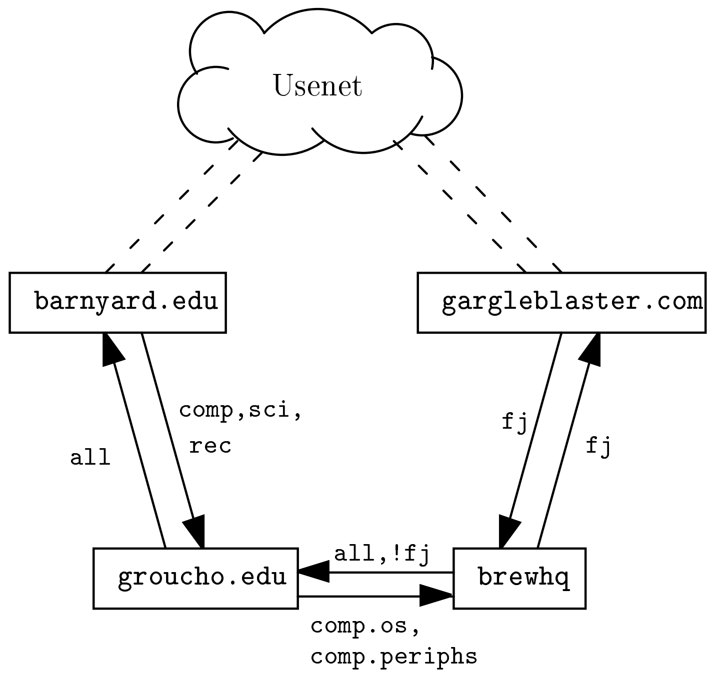

Figura 16.1: Flujo de noticias a través de la Universidad Groucho Marx

recibido por el sistema local de noticias es enviado a estos servidores, a no ser que ya haya pasado por ellos. Se puede saber por qué servidores ha pasado un artículo mirando el campo `Path:` de la cabecera. Este campo contiene una lista con todos los sistemas por los que el artículo ha pasado, separados por un signo de admiración.

Para distinguir entre los artículos y reconocer los duplicados, los artículos de Usenet llevan un identificativo (especificado en el campo `Message-Id:` de la cabecera) que combina el nombre del servidor donde se publicó y un número de serie "<`num-serie@servidor`>". Cada vez que se procesa un artículo, el sistema de noticias registra su identificativo en un fichero (generalmente llamado *history*) con el que después se coteja cualquier nuevo artículo.

El flujo entre dos servidores puede ser limitado por dos criterios: por un lado, al artículo se le asigna una distribución (campo `Distribution:` de la cabecera) que puede ser usada para confinarlo a cierto grupo de servidores. Por otro lado, los grupos de noticias intercambiados pueden limitarse tanto en el sistema emisor como en el receptor. El conjunto de grupos y distribuciones que se permite transmitir a un sistema se suelen especificar en el fichero *sys*.

Debido al gran número de artículos, habitualmente se necesita mejorar el esquema anterior. En las redes UUCP, lo natural es recoger los artículos durante un cierto período de tiempo y combinarlos en un solo fichero, que posteriormente es comprimido y enviado a un sistema remoto. Esto se llama *batching.*[3]

Una técnica alternativa es la del protocolo *ihave/sendme* (*tengo/envíame*) que evita que los duplicados sean enviados en primer lugar, ahorrando ancho de banda. En vez de empaquetar todos los artículos en ficheros y enviarlos, sólo se envían los identificativos de los mensajes en un gigantesco mensaje "ihave". El sistema remoto lee el mensaje, lo compara con su fichero histórico (*history*), y envía un mensaje "sendme" con la lista de artículos que quiere. De este modo, sólo se enviarán estos artículos.

Por supuesto, el protocolo ihave/sendme sólo tiene sentido si atañe a dos grandes sistemas que reciben noticias desde varios sitios independientes, y que se intercambian artículos entre sí con la suficiente frecuencia como para mantener un flujo de noticias eficiente.

Los sevidores conectados a Internet generalmente se basan en programas bajo TCP/IP que usan el protocolo NNTP[4], mencionado anteriormente. Con este protocolo se transfieren artículos entre servidores y se da acceso a Usenet a usuarios individuales en sistemas remotos.

NNTP contempla tres formas diferentes de transferir las noticias. Una es una versión en tiempo real de ihave/sendme, también conocida como *empujar* las noticias. La segunda técnica se denomina *tirar* de las noticias. El cliente solicita una lista de artículos de un grupo o jerarquía determinado que ha llegado al servidor después de una fecha especificada, y elige los que no puede encontrar en su fichero histórico. La tercera forma es para lectura interactiva, y permite al lector escoger artículos de grupos especificados, así como publicar artículos con cabeceras incompletas.

En cada sistema, las noticias se guardan en una estructura de directorios bajo */var/spool/news*, cada artículo en un fichero separado, y cada grupo en un directorio separado. El nombre del directorio se crea a partir del nombre del grupo, donde los componentes del mismo son los componentes de la ruta. Así pues, los artículos de **comp.os.linux.misc** se guardan en */var/spool/news/comp/os/linux/misc.* A cada artículo se le asigna un número según su orden de llegada. Este número sirve como nombre de fichero. El rango de artículos vigentes en un momento dado se guarda en un fichero llamado *active*, que al mismo tiempo sirve como lista de grupos disponibles en el sistema.

Puesto que el espacio en disco es un recurso finito,[5] uno tiene que empezar a desechar los artículos de cierta antigüedad. Esto se denomina *expiración.* Generalmente, los artículos de un determinado grupo o jerarquía expiran al transcurrir un número determinado de días desde de su llegada. El autor puede modificar este valor especificando una fecha de expiración en el campo `Expires:` de la cabecera del artículo.

---

[3] La regla de oro de las noticias de red, según Geoff Collyer: "Empaquetarás tus artículos".

[4] Descrito en la RFC 977.

[5] Alguna gente afirma que Usenet es un conspiración entre vendedores de modems y discos duros.

Capítulo 17

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 */usr/lib/news*, y la mayoría de sus ficheros binarios en el directorio */usr/lib/news/bin*. Los artículos se guardan en */var/spool/news*. Ud. debe estar seguro de que todos los ficheros en estos directorios son propiedad del usuario **news**, grupo **news**. La mayoría de los problemas surgen de la inaccesibilidad de los ficheros por C-News. Ud. debe tener como regla general el ser usuario **news** usando *su* antes de tocar nada ahí. La única excepción es *setnewsids*, que se usa para establecer la identificación real del usuario de algunos programas de noticias. Éste debe ser propiedad del **root** y debe tener el bit setuid activado.

A continuación, describimos todos los ficheros de configuración de C-News en detalle, y le mostraremos lo que tiene que hacer para mantener su servidor en funcionamiento.

17.1 Entrega de Noticias

Los artículos deben ser suministrados a C-News de varias maneras. Cuando un usuario local envía un artículo, el lector de noticias usualmente lo entrega al comando *inews*, el cual completa la información de cabecera. Las noticias del servidor remoto, tanto si es un único mensaje como un lote entero, son entregadas al comando *rnews*, el cual lo almacena en el directorio */var/spool/news/in.coming*, de donde lo cogerá *newsrun* más tarde. Sin embargo, con cualquiera de estas dos técnicas el artículo será finalmente entregado al comando *relaynews*.

Para cada artículo, el comando *relaynews* consulta primero si el artículo ha sido visto en el servidor local buscando el identificador del mensaje en el fichero *history*. Los artículos duplicados serán eliminados. Entonces, *relaynews* mira la línea de cabecera del `Newsgroup:` para averiguar si el servidor local solicita artículos de cualquiera de estos grupos. Si lo hace, y el grupo de noticias está listado en el fichero *active*, *relaynews* intenta almacenar el artículo en el correspondiente directorio en el area de cola de noticias. Si no existe este directorio, se crea. El identificador del mensaje del artículo será entonces registrado en el fichero *history*. De otra manera, *relaynews* elimina el mensaje.

Si *relaynews* falla al almacenar un artículo entrante porque un grupo al que sido enviado no está listado en su fichero *activo*, el artículo será movido al grupo **junk**.[1] *relaynews* también comprobará artículos caducados o mal fechados y los rechazará. Los lotes entrantes que fallan por cualquier razón son movidos a */var/spool/news/in.coming/bad*, y es registrado un mensaje de error.

Después de esto, el artículo será transmitido a todos los otros servidores que soliciten noticias de estos grupos, usando el transporte especificado para cada servidor determinado. Para estar seguro de que no es enviado a un servidor que ya lo ha visto, cada servidor de destino es comparado con el campo `Path:` de cabecera del artículo, el cual contiene la lista de servidores hasta los que el artículo ha llegado, escritos en notación de camino UUCP con signos de admiración. Sólo si el nombre del servidor de destino no aparece en esta lista el artículo le será enviado.

C-News es usado 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 — *uux* es usado para ejecutar el comando *rnews* en el servidor remoto, y entregarle el artículo o lote por su entrada estándar.

Cuando el proceso por lotes está permitido para un servidor dado, C-News no manda inmediatamente ningún artículo entrante, sino que anexiona su nombre de camino a un fichero, usualmente *out.going/nodo/togo*. Periódicamente, un programa por lotes es ejecutado desde la línea de una ***`tabla de tareas planeadas`***, [2] [3] lo que situa a los artículos en uno o más ficheros, opcionalmente los comprime, y los manda a *rnews* en el servidor remoto.

La figura 17.1 muestra las noticias fluyendo a través de *relaynews*. Los artículos deben ser transmitidos al servidor local (denotado por *ME*), a algún servidor llamado **ponderosa** vía correo electrónico, y a un servidor llamado **moria**, para el cual el proceso por lotes

[1] 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 debe especificar **comp.all**, lo que significa todos los grupos de noticias bajo la jerarquía **comp**, pero en su servidor, solo un número de grupos **comp** son listados en *activo*. Los artículos enviados a esos grupos serán movidos a **junk**.

[2] N. del T.: ***`crontab`***

[3] Note que esto debería ser la tabla de tareas planeadas del usuario **noticias**, para no destrozar los permisos de los ficheros.

está permitido.

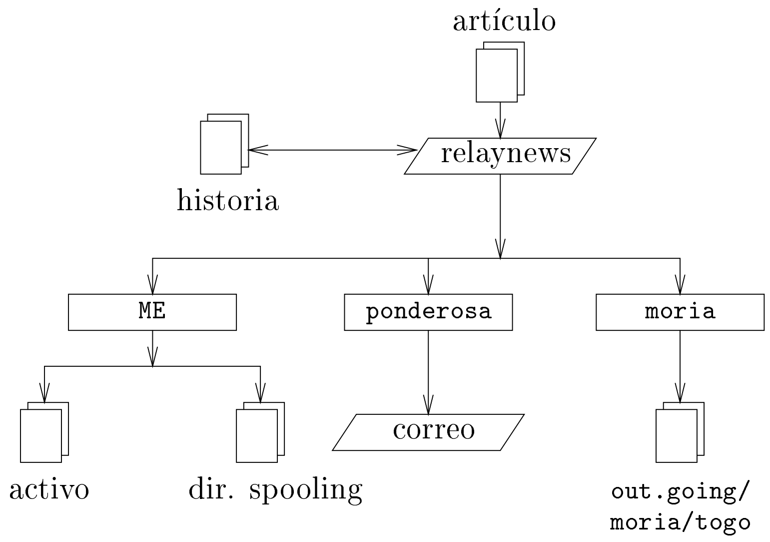

Figura 17.1: Flujo de noticias mediante *relaynews*.

17.2 Instalación

Para instalar C-News, descomprima con `tar` los ficheros en el lugar apropiado, si no lo ha hecho todavía, y edite los ficheros de configuración listados abajo. Todos están situados en */usr/lib/news*. Sus formatos serán descritos en las siguientes secciones.

sys Probablemente Ud. tendrá que modificar la línea *ME* que describe su sistema, aunque usar *all/all* es también una apuesta segura. Ud. también tendrá que añadir una línea por cada servidor al que quiera mandar noticias.

Si Ud. es un servidor hoja, sólo necesita una línea que mande todos los artículos generados localmente a su fuente. Suponga que su fuente es **moria**, entonces su fichero *sys* debería parecerse a:

```
ME:all/all::
moria/moria.orcnet.org:all/all,!local:f:
```

organization El nombre de su organización. Por ejemplo, "`Cervecera Virtual, Inc.`". En su máquina de casa, introduzca "sitio privado", o cualquier cosa que desee. La mayoría de la gente no dirá que su servidor está configurado correctamente hasta que no haya configurado este fichero.

newsgroups

mailname El nombre de su servidor de correo, por ejemplo **vbrew.com**.

whoami El nombre de su servidor para propositos de noticias. Con frecuencia, se usa, por ejemplo, el nombre del servidor UUCP. **vbrew**.

explist Probablemente Ud. debería editar este fichero para reflejar sus tiempos de expiración preferidos para algún grupo de noticias en especial. El espacio de disco debe jugar un importante papel en esto.

Para crear una jerarquía inicial de grupos de noticias, obtenga un fichero *active* y un fichero *newsgroups* del servidor que le provee, e instálelos en */usr/lib/news*, asegurándose de que son propiedad del usuarios news y tienen un modo de protección 664. Elimine todos los grupos **to.*** del fichero *active*, y añada **to.*mi servidor*** y **to.*sitio proveedor***, al igual que **junk** y **control**. Los grupos **to.*** se usan normalmente para intercambiar mensajes ***ihave/sendme***[4] , pero Ud. debería crearlos tanto si planea usar ***ihave/sendme*** como si no. Después, sustituya todos los numeros de los artículos en el segundo y tercer campo de *active* usando el siguiente comando:

```
# cp active active.old
# sed 's/ [0-9]* [0-9]* / 0000000000 00001 /' active.old > active
# rm active.old
```

El segundo comando es una invocación de *sed(1)*, uno de mis comandos UN⋆X favoritos. Esta invocación sustituye dos cadenas de dígitos por una cadena de ceros y la cadena `000001`, respectivamente.

Finalmente, cree el directorio de cola de noticias y los directorios usados para noticias entrantes y salientes:

```
# cd /var/spool
# mkdir news news/in.coming news/out.going
# chown -R news.news news
# chmod -R 755 news
```

Si Ud. está usando una versión de C-News más reciente, deberá crear el directorio *out.master* en el directorio de cola de noticias.

Si está usando lectores de noticias de una distribución diferente de la de C-News, puede descubrir que algunos de ellos esperan encontrar la cola de noticias en */usr/spool/news* en

[4] N. del T.: *Tengo/Envíame*

vez de en */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*.

Ahora, Ud. está preparado para recibir noticias. Note que no tiene que crear ningún otro directorio más que los vistos arriba, porque cada vez que C-News recibe un artículo de un grupo para el que todavía no hay directorio de cola, lo crea.

En particular, esto le ocurre a *todos* los grupos a los que se ha enviado un artículo cruzado. 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 todos los grupos no deseados de *active*, como ejecutando regularmente un *script del shell* 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 defecto, éste es **usenet**. Si usa el valor por defecto, tiene que establecer un alias para él, el cual remite todo su correo a una o más personas responsables. (Los capítulos 14 y 15 explican cómo hacerlo para *smail* y *sendmail*). También puede modificar este comportamiento estableciendo la variable de entorno *NEWSMASTER* con el nombre apropiado. Debe hacerlo en el fichero de la tabla de tareas planeadas de **noticias**, asi como cada vez que invoque manualmente una herramienta administrativa, por lo que instalar un alias es probablemente más fácil.

Aprovechando que está modificando */etc/passwd*, asegúrese de que cada usuario tiene su nombre real en el campo *pw_gecos* del fichero de contraseña (éste es el cuarto campo). Es una cuestión de normas de etiqueta de Usenet el que el nombre real del remitente aparezca en el campo `From:` del artículo. Por supuesto, Ud. querrá hacerlo de cualquier manera cuando use el correo.

17.3 El fichero sys

El fichero *sys*, situado en */usr/lib/news*, controla qué jerarquías recibe y remite a otros servidores. Aunque hay herramientas de mantenimiento llamadas *addfeed* y *delfeed*, creo que es mejor mantener este fichero a mano.

El fichero *sys* contiene entradas para cada servidor al que Ud. reenvía noticias además de descripciones de los grupos de noticias que Ud. acepta. Una entrada se parece a:

```
sitio[/exclusiones]:listagrupos[/listadist] [:flags[:cmds]]
```

Las entradas pueden continuar a lo largo de varias líneas usando una barra invertida (\). Una almohadilla (#) denota un comentario.

`sitio` Éste es el nombre de los servidores a los que se aplica la entrada. Usualmente se elige el nombre del servidor UUCP para esto. Tiene que haber también una entrada para su servidor en el fichero *sys*, si no no recibirá ningún artículo.

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**.

Puesto que C-News compara el ***`servidor`*** con los nombres de los servidores en la cabecera del campo `Path:`, hay que estar seguro de que realmente coinciden. Algunos servidores usan su nombre de dominio completamente cualificado en este campo, o un alias como **news.*`sitio.dominio`***. Para prevenir que cualquier artículo regrese a estos servidores, tiene que añadir esto a la lista de exclusión, separada por comas.

Por ejemplo, para la entrada aplicada al servidor **moria**, el campo del servidor contendría **moria/moria.orcnet.org**.

`listagrupos` Esta es una lista de subscripción, 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 una exclamación. Si un grupo de noticias encaja con más de una definición de la lista, se aplica el emparejamiento más larga. Por ejemplo, si la ***`listagrupos`*** 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 administrados a ese servidor.

Si el servidor requiere que se le reenvíen todas las noticias que Ud. recibe, introduzca *all* como ***`listagrupos`***.

`listadist` está separado de ***`listagrupos`*** por un barra inclinada, y contiene una lista de distribuciones para ser reenviada. Ud. puede de nuevo excluir ciertas distribuciones precediéndolas con una exclamación. Todas las distribuciones se denotan con *all*. El omitir ***`listadist`*** 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 hay al menos dos distribuciones: *world*, que es a menudo la distribución por defecto 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 solamente por C-News; éstas son *sendme* y *ihave*, y son usadas para el protocolo ihave/sendme.

El uso de distribuciones es materia de debate. 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**. 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.[5] Sin embargo, las distribuciones empleadas para una organización, son muy significativas, por ejemplo para evitar la salida de información confidencial de la red de la compañía. No obstante, este proposito generalmente se consigue mejor creando un grupo de noticias o una jerarquía separados.

`flags` este campo describe ciertos parámetros para la fuente. Puede estar vacío, o ser una combinación de lo siguiente:

F Este flag permite el proceso por lotes.

f Éste es casi idéntico al flag *F*, pero permite a C-News calcular el tamaño de los lotes salientes con más precisión.

I Este flag hace que C-News produzca una lista de artículos apta para ser usada por el protocolo ihave/sendme. Hay que hacer modificaciones adicionales al fichero *sys* y al fichero *batchparms* para habilitar ihave/sendme.

n Este flag crea ficheros por lotes para clientes de transferencia NNTP activa como *nntpxmit* (ver capítulo 18). Los ficheros por lotes contienen el nombre de fichero del artículo junto con su identificador de mensaje.

L Este flag indica a C-News que sólo transmita los mensajes generados en su servidor. Este flag puede ir ser seguido por un número decimal *`n`*, el cual hace que C-News sólo transfiera artículos generados a *`n`* saltos desde su servidor. C-News determina el numero de saltos a partir del campo `Path:`.

[5] No es infrecuente para un artículo enviado en, digamos Hamburgo, ir a Frankfurt vía **reston.ans.net** en Holanda, o inclusive vía algún servidor en E.E.U.U.

u Este flag indica a C-News que procese por lotes sólo los artículos de los grupos no moderados.

m Este flag indica a C-News que procese por lotes sólo los artículos de los grupos moderados.

Debe usar a lo sumo uno de *F*, *f*, *I*, o *n*.

cmds Este campo contiene un comando a ser ejecutado para cada artículo, a menos que el proceso por lotes esté habilitado. El artículo será suministrado al comando a través de la entrada estándar. Esto solo debería usarse para fuentes muy pequeñas; de otra manera la carga en ambos sistemas sería demasiado alta.

El comando por defecto es

```
uux - -r -z system!rnews
```

lo que invoca *rnews* en el sistema remoto, administrando el artículo mediante la entrada estándar.

El camino de búsqueda por defecto para los comandos indicados en este campo es */bin:/usr/bin:/usr/lib/news/bin/batch*. El último directorio contiene un un cierto número de guiones del intérprete de comandos cuyos nombres empiezan por *via*; se describen brevemente más adelante en este mismo capítulo.

Si el proceso por lotes está habilitado usando bien los flags *F* o *f*, *I* o *n*, C-News espera encontrar un nombre de fichero en este campo en vez de un comando. Si el nombre de fichero no empieza con una barra inclinada (/), se supone que es relativo a */var/spool/news/out.going*. Si el campo está vacío, su valor por defecto es ***system**/togo*.

Cuando configure C-News, probablemente tendrá que escribir su propio fichero *sys*. Para ayudarle con ello, incluimos abajo un fichero de ejemplo para **vbrew.com**, del cual puede copiar lo que necesite.

```
# Tomamos lo que nos dan.
ME:all/all::

# Enviamos todo lo que recibimos a moria, excepto los articulos locales y
# relacionados con cerveceras. Usamos proceso por lotes.
moria/moria.orcnet.org:all,!to,to.moria/all,!local,!brewery:f:

# Mandamos comp.risks a jack@ponderosa.uucp
```

```
ponderosa:comp.risks/all::rmail jack@ponderosa.uucp

# swim obtiene solo algunos grupos
swim/swim.twobirds.com:comp.os.linux,rec.humor.oracle/all,!local:f:

# Guardar los articulos de mapas de correo para procesarlos luego
usenet-maps:comp.mail.maps/all:F:/var/spool/uumaps/work/batch
```

17.4 El fichero *active*

El fichero *active* está situado en */usr/lib/news* y lista todos los grupos conocidos en su servidor, y los artículos disponibles actualmente. Rara vez tendrá que tocarlo, pero, sin embargo, lo explicamos por completitud. Las entradas tiene la siguiente forma:

```
gruponoticias alto bajo perm
```

`gruponoticias` es, por supuesto, el nombre del grupo. `bajo` y `alto` son los números más bajo y más alto de los artículos actualmente disponibles. Si no hay ninguno disponible en ese momento, `bajo` es igual a `alto`+1.

Al menos, eso es lo que el campo `bajo` pretende hacer. Sin embargo, por razones de eficiencia, C-News no actualiza este campo. Esto no sería una gran perdida si no hubiera algunos lectores de noticias que dependen de él. Por ejemplo, *trn* comprueba este campo para ver si puede purgar cualquier artículo de su base de datos de hilos. Para actualizar el campo `bajo`, tiene por lo tanto que ejecutar regularmente el comando *updatemin* (o, en una versión más antigua de C-News, la macro *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 enviar artículos a este grupo.

n No está permitido a los usuarios enviar artículos a este grupo. Sin embargo, el grupo puede todavía ser leído.

x Este grupo ha sido deshabilitado localmente. Esto ocurre algunas veces cuando los administradores de noticias (o sus superiores) se ofenden por artículos enviados a 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 lo notificará al usuario, y en su lugar enviará el artículo al moderador. La dirección del moderador se toma del fichero *moderators* de */usr/lib/news*.

=*real-group*
Esto marca a ***newsgroup*** como un alias local para otro grupo, a saber ***real-group***. Todos los artículos enviados a ***gruponoticias*** 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* (ver abajo en la sección Herramientas y Tareas de Mantenimiento). Cuando se añaden o borran grupos para la Usenet entera, esto se hace habitualmente por medio de un mensaje de control *newgroup* o *rmgroup*, respectivamente. *¡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**.

Un fichero estrechamente relacionado con *active* es *active.times*. 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 hecho por un mensaje de control *newgroup* o 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 el comando *NEWGROUPS* de NNTP.

17.5 Procesado de 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 cuenta
```

donde ***cuenta*** es el numero 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 descompresión. 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 lo que se llama codificación C7; estos lotes serán marcados por *c7unbatch*.

Cuando se le administra un lote a *rnews* en el servidor remoto, comprueba esas marcas y procesa el lote apropiadamente. Algunos servidores también usan otras herramientas de compresión, como *gzip*, y en su lugar preceden sus ficheros comprimidos con *zunbatch*. C-News no reconoce cabeceras no estándares como esas; Ud. tiene que modificar el código fuente para soportarlas.

En C-News, el proceso por lotes de archivos lo realiza */usr/lib/news/bin/batch/sendbatches*, el cual recoje 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 del tráfico.

Su operación es controlada por el fichero *batchparms* situado en */usr/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 a ser usado, y método de transporte para entregarlo al servidor remoto. Ud. puede especificar los parámetros del proceso por lotes para cada servidor, además de un conjunto de parámetros por defecto para servidores no mencionados explícitamente.

Para llevar a cabo el proceso por lotes para un servidor específico, se invoca como:

```
# su news -c "/usr/lib/news/bin/batch/sendbatches site"
```

Cuando es invocado sin argumentos, *sendbatches* maneja todas las colas de lotes. La interpretación de "todas" depende de la presencia de una entrada por defecto en *batchparms*. Si se encuentra una, se comprueban todos los directorios de */var/spool/news/out.going*, si no, recorre todas las entradas de *batchparms*. Note que *sendbatches*, cuando explora el directorio *out.going*, toma sólo aquellos directorios que no contienen ningún punto o arroba (@) como nombre de servidor.

Cuando instale C-News, seguramente hallará un fichero *batchparms* en su distribución que contenga una entrada por defecto razonable, 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:

```
site size max batcher muncher transport
```

El significado de estos campos es el siguiente:

`site` 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 las noticias. El nombre de servidor */default/* denota la entrada por defecto.

`size` 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 ellos solos.

`max` es el máximo número de lotes creados y programados para la transferencia antes de que el proceso por lotes se pare para este servidor 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 script *queulen* de */usr/lib/news/bin*. La versión *newspak* de Vince Skahan debería contener un guión para UUCPs compatibles con BNU. Si usa una clase diferente de directorios de cola, por ejemplo UUCP de Taylor, tendría que escribir el suyo propio.[6]

El campo `batcher` contiene el comando usado 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/sendme, los cuales se envían al grupo `to.site`. Los comandos encargados de esto son *batchih* y *batchsm*.

El campo `muncher` especifica el comando a usar para la compresión de los lotes. Generalmente, se usa **compcun**, que es un guión que produce un lote comprimido.[7] Alternativamente, puede proporcionar un `muncher` que use *gzip*, digamos *gzipcun* (para ser claros: tiene que escribirlo Ud. mismo). Debe asegurarse de que *uncompress* en el servidor remoto está parcheado para reconocer ficheros comprimidos con *gzip*.

Si el servidor remoto no tiene un comando *uncompress*, debe especificar *nocomp* lo que implica el no hacer ninguna compresión.

El último campo, `transport`, describe el transporte a utilizar. Hay disponibles varios comandos estándar para diferentes transportes cuyos nombres empiezan por *via*. *sendbatches* les pasa el nombre del servidor de destino en la línea de comandos. Si la entrada *batchparms* no era */default/*, el nombre del servidor se obtiene del campo `site` suprimiendo

[6] Si no le importa el numero de ficheros de cola (porque Ud. es la única persona usando el ordenador, y no escribe artículos de megabytes), puede reemplazar los contenidos del guión por una simple sentencia *exit 0*.

[7] 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 servidores. Ud. puede hacer una copia de él, llamémosla **compcun16**, y usar la compresión 16 bit. De todas formas, la mejora no es muy impresionante.

cualquier cosa después e incluyendo el primer punto o barra inclinada. Si la entrada era */default/*, se usan los nombres de directorio de *out.going*.

Hay dos comandos que usan *uux* para ejecutar *rnews* en el servidor remoto; *viauux* y *viauuxz*. El último establece el flag `-z` para (las versiones más antiguas de) *uux* para evitar que devuelva mensajes de éxito por cada artículo entrgado. Otro comando, *viamail*, manda lotes de artículos al usuario **rnews** en el sistema 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 la pagina del manual *newsbatch(8)*.

Todos los comandos de los tres últimos campos deben estar situados, bien en *out.going/*`site` o bien en */usr/lib/news/bin/batch*. La mayoría de ellos son scripts, de tal forma que Ud. pueda confeccionar fácilmente nuevas herramientas para sus necesidades personales. Son invocados con ***tuberías***. Se administra la lista de artículos al ***batcher*** a través de la entrada estándar, quien produce el lote en su salida estándar. Esta a su vez se ***entuba*** en el ***muncher***, y así sucesivamente.

Abajo se ofrece un fichero de ejemplo.

```
# fichero batchparms para la cervecera
# site          | size    |max     |batcher  |muncher    |transport
#-------------+--------+-------+---------+-----------+-----------
/default/        100000  22      batcher   compcun     viauux
swim              10000  10      batcher   nocomp      viauux
```

17.6 Noticias caducadas

En Bnews, la caducidad de las noticias solía realizarse por medio de un programa llamado *expire*, el cual recibía como argumento una lista de grupos de noticias, junto con una especificación del tiempo después del cual los artículos caducaban. Para hacer que diferentes jerarquías caducaran en momentos distintos, Ud. tenía que escribir un script que invocara a *expire* para cada uno de ellos de forma individual. C-News ofrece una solución más conveniente a esto: en un fichero llamado *explist*, Ud. puede especificar los grupos de noticias y los intervalos de caducidad. Una vez al día se suele ejecutar desde *cron* un comando llamado *doexpire*, que procesa todos los grupos de acuerdo 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* se parece a esto:

```
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 una indicar una entrada que se aplique a todos los grupos de **comp.os**, puede introducir en ***grouplist*** o bien **comp.os** o bien **comp.os.all**.

Cuando se van a caducar las noticias de un grupo, se contrasta el nombre del grupo con todas las entradas de *explist* en el orden dado. La entrada empleada es la primera que concuerda. Por ejemplo, para eliminar la mayoría de **comp** después de cuatro días, excepto **comp.os.linux.announce** que quiere mentener durante una semana, debe simplemente tener una entrada para lo último, que especifique un periodo 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 los valores *m*, *u*, o *x*, lo que designa moderados, no moderados, o cualquier tipo.

El tercer campo, ***times***, contiene usualmente 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. Dése cuenta de que este es el número de días contando desde la *llegada* a su servidor, no desde la fecha de emisión.

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 caducado. Rara vez es útil usar otro valor que no sea cero. El segundo campo es el valor mencionado arriba, es decir, el número por defecto de días después de los cuales caducará. El tercero es el número de días después de los cuales un artículo caducará incondicionalmente, sin reparar en si tiene un campo `Expires:` o no. Si sólo se indica el número de en medio, los otros dos toman valores por defecto. Estos pueden especificarse usando la entrada especial */bounds/*, que se describe más abajo.

El cuarto campo, ***archive***, designa si el grupo de noticias tiene que archivarse, y dónde. Si no se desea archivarlo, debería usar un guión. De lo contrario, use un nombre de camino absoluto (apuntando a un directorio), o una arroba (@). La arroba designa el directorio de archivo por defecto, cuyo valor debe darse a *doexpire* usando el flag `-a` en la línea de comandos. El directorio de archivo debe ser propiedad de **news**. Cuando *doexpire* archiva un artículo de, digamos, **comp.sources.unix**, lo almacena en el directorio

comp/sources/unix bajo el directorio de archivo, creándolo si no existe. Sin embargo, no se creará el propio directorio de archivo.

Hay dos entradas especiales en su 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 por defecto para los tres valores del campo `times` descrito arriba.

El campo */expired/* determina cuánto tiempo guardará C-News las líneas del fichero *history*. Esto es necesario porque C-News no borrará una línea del fichero de historial una vez que el (los) artículo(s) correspondiente(s) hayan caducado, pero lo guardará por si acaso llega un duplicado trás esa fecha. Si recibe las noticas de solo un servidor, puede mantener este valor pequeño. De lo contrario, un par de semanas es un valor aconsejable para las redes UUCP, dependiendo de los retrasos que Ud. experimente con los artículos de esos servidores.

A continuación se reproduce un fichero *explist* de ejemplo con unos intervalos de expiración bastante ajustador.

```
# Mantiene las lineas de historial durante dos semanas. Nadie consigue mas
# de tres meses
/expired/                       x       14      -
/bounds/                        x       0-1-90  -

# Los grupos que queremos mantener mas 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 por defecto para los grupos de tecnologia
comp,sci                        x       7       -

# bastante para un fin de semana largo
misc,talk                       x       4       -

# desecha rapidamente lo inservible
junk                            x       1       -

# los mensajes de control tambien son de escaso interes
control                         x       1       -
```

```
# para el resto de ellos la entrada comodin
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*, el cual contiene el número del artículo más bajo disponible. Cuando C-News caduca los artículos no actualiza este campo. Si Ud. necesita (o quiere) que este campo represente la situación real, necesita ejecutar un programa llamado *updatemin* después de cada ejecución de *doexpire*.[8]

Segundo, 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.[9] Si su fichero de historia 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.[10] Puede reparar esto usando el script *addmissing* de */usr/lib/news/bin/maint*, el cual añadirá los artículos perdidos al fichero *history*, o *mkhistory*, el cual 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.

17.7 Ficheros diversos

Hay algunos ficheros que controlan el comportamiento de C-News, pero no son esenciales para su funcionamiento. Todos ellos residen en */usr/lib/news*. Los describiremos brevemente.

newsgroups Se trata de un fichero que 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. Este fichero se actualiza automáticamente cuando C-News recibe un mensaje de control *checknews* (ver sección 17.8).

localgroups Si Ud. tiene grupos locales de los que no quiere que C-News se queje cada vez que Ud. recibe un mensaje *checknews*, ponga sus nombres y una descripción 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 (separada por un tabulador).

[8]En versiones anteriores de C-News, esto era hecho por un un script llamado *upact*.

[9]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.

[10]No se *por qué* ocurre esto, a mí me lo hace de vez en cuando.

Hay dos entradas especiales que son proporcionadas por defecto. Éstas son *backbone* e *internet*. Ambas proporcionan — en notación 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 por defecto son

```
internet     backbone
```

Ud. no tendrá que cambiar la entrada *internet* si no tiene instalado *smail* o *sendmail*, porque entienden direccionamiento RFC 822.

La entrada *backbone* se usa cada vez que un usuario envía un mensaje a un grupo moderado cuyo moderador no esté listado explicítamente. 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 principal pueda reenviar el artículo. Para averiguar qué camino usar, pregunte al administrador de noticias del servidor que le pasa las mismas. Como último recurso, puede usar también **uunet.uu.net!%s**.

distributions Este fichero no es realmente un fichero C-News, pero es usado por algunos lectores de noticias y *nntpd*. Contiene la lista de distribuciones reconocida por su servidor, 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 Cerveceria 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 registro antiguos se guardan en *log.o*, *log.oo*, etc.

errlog Éste es un registro de todos los mensajes de error creados por C-News. Éstos no incluyen artículos desechados debido a grupos incorrectos, etc. De no estar vacío en el momento de ejecutar *newsdaily*, será enviado por correo al administrador de las noticias (**usenet** por defecto) automáticamente.

newsdaily se encarga de limpiar *errlog*. Las copias antiguas se guardan en *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 Éste es un fichero vacío que se crea cada vez que se ejecuta *newswatch*.

17.8 Mensajes de Control

El protocolo de noticias Usenet reconoce artículos de una categoría especial, los cuales provocan ciertas respuestas o acciones del sistema. Éstos 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 de ellas, todas ellas manejadas por guiones del intérprete de comandos 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 defecto, sólo los mensajes *checkgroups* serán entregados al administrador de noticias,[11] pero Ud. puede cambiar esto editando los scripts.

17.8.1 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 en si el artículo ha sido visto ya o no. Esto es para tener el 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.

17.8.2 *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 las jerarquías "usuales" sólo puede ser creados después de que se 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 de **news.announce.newusers** y

[11] Hay una errata divertida en el RFC 1036 (pag.12): "Los implementadores y administradores pueden elegir el permitir que los mensajes de control se lleven a cabo automáticamente, o encolarlos para su proceso anual."

news.announce.newgroups. Nunca mande un mensaje *newgroup* o *rmgroup* usted mismo a menos que sepa con seguridad que tiene permiso para hacerlo.

17.8.3 El Mensaje *checkgroups*

Los mensajes *checkgroups* son mandados 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 programa *checkgroups* usando el siguiente comando:

```
# su news -c "/usr/lib/news/bin/ctl/checkgroups" < /tmp/check
```

Esto actualizará su fichero *newsgroups*, añadiendo los grupos listados en *localgroups*. El antiguo fichero *newsgroups* será movido a *newsgroups.bak*. Note que rara vez funciona el enviar el mensaje localmente, porque *inews* rechaza aceptar un artículo tan grande.

Si C-News encuentra desigualdades entre la lista *checkgroups* y el fichero *active*, producirá una lista de comandos que actualizaría su servidor, y lo enviará por correo al administrador 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/bin/maint/delgroup alt.ascii-art
         /usr/lib/news/bin/maint/delgroup bionet.molbio.gene-org
         /usr/lib/news/bin/maint/delgroup comp.windows.x.intrisics
         /usr/lib/news/bin/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 de jerarquías enteras; por lo tanto debería tener cuidado al borrar cualquier grupo. Si Ud. encuentra grupos listados como no presentes que quiera tener en su servidor, tiene que añadirlos usando la herramienta *addgroup*. Salve la lista de grupos que le faltan en un fichero y páseselo al siguiente guión:

```
#!/bin/sh
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/bin/maint/addgroup $group $mod
done
```

17.8.4 *sendsys*, *version*, y *senduuname*

Finalmente, hay tres mensajes que pueden usarse para averiguar la topología de la red. Éstos 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(1)*. C-News es muy lacónica con respecto a los mensajes *version*; devuelve una simple “C”, sin adornos.

Insistimos de nuevo, Ud. *nunca* debería distribuir tal mensaje, a menos que esté seguro de que no puede dejar su red (regional). Las respuestas a los mensajes *sendsys* pueden hacer caer rápidamente una red UUCP.[12]

17.9 C-News en un Entorno NFS

Una manera simple de distribuir noticias en una red local es guardar todas las noticias 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`** y dejarían los ficheros inconsistentes. También Ud. podría querer proteger su área de cola de noticias exportándola como solo-lectura, lo cual requiere también el reenvío a la máquina central.

C-News maneja esto transparentemente. Cuando envía un artículo, su lector de noticias normalmente llamará a *inews* para inyectar el artículo en el sistema de noticias. Este comando ejecuta algunas comprobaciones sobre el artículo, completa la cabecera, y comprueba el fichero *server* en */usr/lib/news*. Si este fichero existe y contiene un nombre de nodo diferente del nombre del sistema local, se invoca *inews* en ese servidor remoto vía *rsh*. Puesto que el guión *inews* usa comandos binarios y ficheros de apoyo de C-News, Ud. tiene que 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 necesitar contraseña.

Asegúrese de que el nombre del sistema indicado en *server* coincida literalmente con la salida del comando *hostname(1)* en la máquina servidora, si no C-News entrará en un bucle infinito cuando intente entregar el artículo.

17.10 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 algunos de éstos sean ejecutados regularmente desde *cron*, como

[12] Yo tampoco intentaría esto en Internet.

newsdaily. El uso de estos programas reduce drásticamente los requisitos diarios de cuidado y administración de su instalación de C-News.

A menos que se indique lo contrario, estos comandos están situados en */usr/lib/news/bin/maint*. Note que Ud. debe ser el usuario **news** antes de invocar estos comandos. Ejecutándolos como super-usuario puede volver estos ficheros inaccesibles a C-News.

newsdaily El nombre ya lo dice: ejecutar esto una vez al día. Es un guión importante que le ayuda a mantener los ficheros de registro pequeños, conservando copias de cada todos ellos de las tres últimas 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 script 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 operatibilidad 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 de disco.

addgroup Añade un grupo localmente a su servidor. La invocación adecuada 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*).

Ud. 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 el directorio de cola del grupo de noticias. Aunque se puede dejar esta tarea al proceso natural de expiración de los artículos.

addmissing Añade artículos perdidos al fichero *historial.* Ejecute este guión cuando haya artículos que parezcan quedarse para siempre.[13]

newsboot Este guión se debería ejecutar cuando arranca el sistema. Elimina cualquier fichero de bloqueo que se dejó atrás cuando se mataron los procesos al apagar, y cierra y ejecuta cualquier lote dejado por las conexiones NNTP que se cerraron cuando se apagó el sistema.

newsrunning Éste reside en */usr/lib/news/bin/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 de lotes invocando

```
/usr/lib/news/bin/input/newsrunning off
```

Se conecta usando *on* en vez de *off*.

[13] ¿Alguna vez se ha preguntado cómo librarse del artículo "¡Ayuda! ¡¡¡No puedo hacer que las X11 funcionen con 0.97.2!!!"?

Capítulo 18

Una descripción de NNTP

18.1 Introducción

El NNTP proporciona una forma de intercambio de noticias totalmente diferente de Cnews, para adaptarse a los protocolos de transporte usados en la Red. NNTP son las siglas de "Network News Transfer Protocol" (Protocolo de Transferencia de Noticias de Red), y no consiste en un paquete de programas en particular, sino que es un estándar de Internet[1] Está basado en una comunicación orientada a la conexión – generalmente sobre TCP – entre un cliente en algún lugar de la red, y un servidor que almacena las noticias en disco. La conexión de flujo permite al cliente y al servidor negociar la transferencia de artículos interactivamente, sin apenas retrasos, manteniendo bajo el número de artículos duplicados. Junto con los altos ratios de transferencia de Internet, esto supone un transporte de noticias que supera ampliamente a las redes UUCP originales. Mientras que hace algunos años no era extraño que un artículo tardase dos semanas o más en llegar hasta el último rincón de Usenet, ahora suele tardar menos de dos días; en la propia Internet, es incluso cuestión de minutos.[2]

Varios comandos permiten a los clientes obtener, enviar y publicar artículos. La diferencia entre enviar y publicar es que esto último puede incluir a artículos con cabeceras incompletas.[3] La obtención de artículos puede ser usada por clientes de transporte de noticias o por lectores de noticias. Esto hace del NNTP una excelente herramienta para proporcionar acceso a muchos clientes de una red local sin tener que pasar por las dificultades que implica usar NFS.

[1]Especificado formalmente en la RFC 977.

[2]N. del T.: Parece ser que el autor tiene la fortuna de no conocer los servidores españoles.

[3]Cuando se publica un artículo mediante NNTP, el servidor siempre añade como mínimo un campo en la cabecera: `NNTP-Posting-Host:`, que contiene el nombre del sistema del cliente.

El NNTP también proporciona una forma activa y otra pasiva de transmitir las noticias, llamadas coloquialmente "empujar" y "tirar". Empujar es básicamente lo mismo que el protocolo *ihave/sendme* de Cnews. El cliente ofrece un artículo al servidor a través del comando "*IHAVE* <`msgid`>", a lo que el servidor responde con un código que indica si ya tiene el artículo o si lo quiere. En el último caso, el cliente envía el artículo, terminado en un solo punto en una línea separada.

Empujar las noticias tiene la única desventaja de que consume muchos recursos del servidor, ya que éste tiene que buscar en su archivo histórico para cada artículo.

La técnica opuesta es tirar de las noticias, en la que el cliente solicita una lista de todos los artículos disponibles en un grupo que hayan llegado después de una fecha especificada. La consulta es llevada a cabo por el comando *NEWNEWS*. De la lista de identificativos de mensaje obtenida, el cliente selecciona aquéllos que aún no tenga, usando el comando *ARTICLE* para obtener cada uno de ellos.

El problema de esta técnica es que necesita un estricto control por parte del servidor, que debe tener en cuenta qué grupos y distribuciones permite solicitar al cliente. Por ejemplo, debe asegurarse de que ningún material confidencial de sus grupos locales sea enviado a clientes no autorizados.

Existe también cierto número de comandos convenientes para los lectores de noticias que permiten obtener la cabecera del artículo y el cuerpo del mismo separadamente, o incluso sólo ciertos campos de la cabecera de un rango de artículos. Esto permite mantener todas las noticias en un servidor central, con todos los usuarios de la red (presumiblemente local) utilizando programas clientes basados en el NNTP para leer y publicar. Esto es una alternativa a exportar los directorios mediante NFS tal como se describe en el capítulo 17.

Un problema extendido en NNTP es que permite a gente con los conocimientos suficientes insertar artículos con remitentes falsos en el flujo de noticias. Esto se conoce como *falsificar las noticias*.[4] Una extensión del NNTP permite requerir autentificación del usuario para ciertos comandos.

Hay cierto número de paquetes NNTP disponibles. Uno de los más conocidos es el demonio NNTP, también conocido como la *implementación de referencia*. Fue escrito originalmente por Stan Barber y Phil Lapsley para ilustrar los detalles de la RFC 977. Su versión más reciente es *nntpd-1.5.11*, que se describirá a continuación. Usted puede obtener el código fuente y compilarlo por su cuenta, o usar el *nntpd* del paquete binario *net-std* de Fred van Kempen. No se proporcionan ejecutables del *nntpd* listos para funcionar, ya que varios valores específicos de cada sistema deben ser introducidos antes de la compilación.

[4] El mismo problema existe con el SMTP, el *Simple Mail Transfer Protocol* (Protocolo Simple para Transferencia de Correo).

El paquete *nntpd* consiste en un servidor y dos clientes para empujar y tirar de las noticias, respectivamente, así como un sustituto para *inews*. Están pensados para trabajar en el entorno de Bnews, pero trabajándoselo un poco lo harán con Cnews sin demasiada dificultad. Sin embargo, si planea Vd. usar el NNTP para algo más que ofrecer acceso a su servidor a los lectores de noticias, la implementación de referencia no es realmente una opción. Por tanto, discutiremos solamente el demonio NNTP contenido en el paquete *nntpd*, dejando de lado los programas clientes.

También hay un paquete llamado "InterNet News", o INN para abreviar, escrito por Rich Salz. Este paquete proporciona tanto transporte NNTP como UUCP, y es el más adecuado para grandes servidores. En lo que a transporte NNTP se refiere, es definitivamente mejor que *nntpd*. La versión actual de INN es *inn-1.4sec*. Existe un paquete de Arjan de Vet para construir INN en una máquina Linux; está disponible en **sunsite.unc.edu** en el directorio *system/Mail*. Si quiere configurar INN, por favor remítase a la documentación que acompaña al código fuente, así como al FAQ de INN, publicado regularmente en **news.software.b**.

18.2 Instalación del servidor NNTP

El servidor NNTP se llama *nntpd*, y puede ser compilado de dos maneras, según el tráfico que se espera que soporte el sistema de noticias. No hay versiones compiladas disponibles, ya que algunos valores por defecto dependientes del sistema en que se vaya a instalar deben ser especificados antes de la compilación. Toda la configuración se hace a través de macros `#define` en *common/conf.h*.

nntpd puede ser configurado como un servidor independiente que se inicie desde *rc.inet2* al arrancar, o como un demonio controlado por *inetd*. En el último caso se tendrá que añadir la siguiente entrada en */etc/inetd.conf*:

```
nntp    stream  tcp nowait      news    /usr/etc/in.nntpd    nntpd
```

Si configura Vd. *nntpd* como servidor independiente, asegúrese de que la línea anterior está comentada en *inetd.conf*. En cualquier caso, tendrá Vd. que asegurarse de que existe la siguiente línea en */etc/services*:

```
nntp    119/tcp   readnews untp    # Network News Transfer Protocol
```

Para almacenar temporalmente los artículos que llegan al sistema, etc, *nntpd* también necesita un subdirectorio *.tmp* en el directorio de almacenamiento de noticias. Puede Vd. crearlo usando

```
# mkdir /var/spool/news/.tmp
# chown news.news /var/spool/news/.tmp
```

18.3 Restricciones de acceso NNTP

El acceso a los recursos NNTP se rige por el fichero *nntp_access* en */usr/lib/news*. Las líneas del fichero describen los derechos de acceso para ordenadores ajenos. Cada línea tiene el siguiente formato:

site `read|xfer|both|no post|no` [!*exceptgroups*]

Si un cliente se conecta al puerto NNTP, *nntpd* intenta obtener su nombre completo en la red a partir de su dirección IP. El nombre del ordenador del cliente y su dirección IP son contrastados con el campo `site` de cada entrada, en el mismo orden en el que aparecen en el fichero. Las coincidencias pueden ser parciales o exactas. Si una entrada coincide exactamente, se aplica; si la coincidencia es parcial, sólo se aplica si no hay otra entrada posterior igual o mejor. `site` puede especificarse de una de las siguientes formas:

nombre del ordenador
: Este es el nombre completo del ordenador. Si coincide literalmente con el nombre canónico del cliente, se aplica directamente esta entrada ignorándose las siguientes.

dirección IP
: Esta es la dirección IP representada por cuatro números separados por puntos. Si la dirección IP del cliente coincide con ella, se aplica la entrada ignorándose las siguientes.

nombre del dominio
: Esto es un nombre de dominio, especificado como `*.dominio`. Si el dominio del cliente coincide con él, se aplica la entrada.

nombre de la red
: 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 el número de red asociado al nombre de la red, se aplica la entrada.

default
: Es la entrada por omisión; se aplica a cualquier cliente.

Las entradas con especificaciones más generales deberían ser introducidas al principio del fichero, ya que después pueden ser descartadas al encontrarse mejores coincidencias en entradas posteriores.

Los campos segundo y tercero describen los derechos de acceso que se otorgan al cliente. El segundo detalla los permisos de lectura (*read*) y transmisión por empuje (*xfer*) de noticias. El valor *both* habilita ambos, el valor *no* niega el acceso a los dos. El tercer campo detalla si el cliente puede publicar artículos, es decir, enviar artículos con información incompleta en la cabecera que será completada por los programas de noticias. Si el segundo campo contiene *no*, el tercero es ignorado.

El cuarto campo es opcional. Contiene una lista de grupos separados por comas a los que el cliente no puede acceder.

A continuación se muestra un fichero *nntp_access* de ejemplo:

```
#
# por defecto, cualquiera puede transferir noticias, pero no
# leerlas o publicarlas
default                 xfer            no
#
# public.vbrew.com ofrece acceso publico via modem, asi que les
# dejamos leer y publicar en cualquier grupo menos en los local.*
public.vbrew.com        read            post    !local
#
# el resto de ordenadores de vbrew.com puede leer y publicar
*.vbrew.com             read            post
```

18.4 Autorización NNTP

Al poner en mayúsculas los elementos del fichero *nntp_acces*, tales como *xfer* o *read*, *nntpd* exige que el cliente esté autorizado para realizar dichas operaciones. Por ejemplo, si se especifica el permiso *Xfer* o *XFER*, *nntpd* no dejará transmitir artículos al cliente a menos que éste acredite que está autorizado.

El proceso de autorización se lleva a cabo por un nuevo comando del NNTP llamado *AUTHINFO*. Usando este comando, el cliente transmite un nombre de usuario y una contraseña al servidor NNTP. *nntpd* los validará comprobando el fichero */etc/passwd*, y verificando que el usuario pertenece al grupo **nntp**.

Lo actual implementación de la autorización NNTP es sólo experimental, por lo que no se ha hecho muy portable. Por ello sólo funcionará con ficheros */etc/passwd* normales; el *shadow password* no está soportado.

18.5 Interacción de *nntpd* con Cnews

Cuando recibe un artículo, *nntpd* debe pasarlo al sistema de noticias. Dependiendo de si lo recibió mediante el comando *IHAVE* o el *POST*, lo enviará a *rnews* o *inews*, respectivamente. En vez de llamar a *rnews*, *nntpd* también puede configurarse (antes de la compilación) para empaquetar los artículos entrantes, y dejar los paquetes resultantes en el directorio */var/spool/news/in.coming*, donde serán procesados por *relaynews* la próxima vez que se le invoque.

Para llevar a cabo con éxito el protocolo ihave/sendme, *nntpd* tiene que poder acceder al fichero histórico *history*. Por tanto, antes de la compilación hay que asegurarse de que la ruta a dicho fichero está correctamente especificada. También hay que asegurarse de que Cnewsy *nntpd* usen el mismo formato de fichero histórico. Cnewsusa funciones *dbm* para acceder al fichero; sin embargo hay bastantes implementaciones ligeramente incompatibles de la librería *dbm*. Si Cnewsestá enlazado con una librería *dbm* distinta a la de *libc*, también deberá enlazarse *nntpd* con dicha librería.

Un síntoma típico de que *nntpd* y Cnewsdiscrepan en el formato del fichero histórico son los mensajes de error de que *nntpd* no puede abrirlo, o la recepción de artículos duplicados por NNTP. Una prueba conveniente es coger un artículo del sistema, hacer telnet por el puerto del *nntpd*, y ofrecérselo a *nntpd* tal como se muestra en el ejemplo de abajo. Por supuesto, deberá reemplazarse <`msg@id`> con el identificativo del artículo que quiera ofrecerse a *nntpd*.

```
$ telnet localhost nntp
Trying 127.0.0.1...
Connected to loalhost
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
```

Este diálogo muestra la reacción correcta de *nntpd*: el mensaje "`Got it`" indica que ya tiene dicho artículo. En cambio, si se obtuviese un mensaje como "`335 Ok`", significaría que *nntpd* no ha podido acceder adecuadamente al fichero histórico. Cierre la sesión telnet con Ctrl-D. Puede comprobar qué ha ido mal revisando el archivo de registro (*log*) del sistema; *nntpd* envía todo tipo de mensajes a *syslog*. El uso de una librería *dbm* incompatible suele reflejarse en un mensaje que indica que *dbminit* ha fallado.

Capítulo 19

Configuración del lector de noticias

Los lectores de noticias están pensados para ofrecer al usuario un acceso fácil a las funciones de un sistema de noticias, tales como publicar artículos, o purgar los contenidos de un grupo de una manera cómoda. El mayor o menor acierto en cumplir este objetivo es objeto de interminables discusiones en los grupos de noticias.

Algunos de los lectores disponibles que han sido portados a Linux. A continuación se describirá la instalación básica para tres de los 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 un interfaz a pantalla completa con niveles separados para mostrar todos los grupos a los que el usuario está suscrito, para mostrar una lista de todos los artículos de un grupo, y para artículos individuales.

En el nivel del grupo, la mayoría de los lectores muestran una lista de artículos en la que aparece el tema de los mismos y el autor. En los grupos grandes es imposible para el usuario caer en la cuenta de los artículos que se refieren unos a otros, 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 puede indicarse en el campo `References:`. Ordenar los artículos por esos dos criterios genera pequeños árboles llamados *hebras*[1]. Una de las tareas al escribir un lector de noticias es diseñar un algoritmo eficiente

[1] N. del T.: Son los llamados popularmente *threads*

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 los interfaces de usuario. Todos los lectores actualmente disponibles para Linux tienen una buena función de ayuda, así que el usuario puede apañárselas solo.

En lo sucesivo, sólo trataremos cuestiones administrativas. La mayoría de ellas relacionadas con la creación de bases de datos y contabilidad.

19.1 Configuración de *tin*

El lector más versátil en lo que al tratamiento de las hebras se refiere es *tin*. Fue escrito por Iain Lea siguiendo el modelo de un lector anterior llamado *tas*.[2] Ordena las hebras en el momento en el que el usuario accede al grupo, y es muy rápido haciéndolo, salvo que se haga por NNTP.

En un 486DX50 se tarda unos 30 segundos en ordenar mil artículos, leyéndolos directamente desde el disco. Mediante NNTP con un servidor ocupado, rondaría los cinco minutos.[3] Se puede mejorar este tiempo actualizando regularmente los ficheros índice con la opción `-u`, o llamando a *tin* con la opción `-U`.

Normalmente *tin* guarda la información sobre las hebras en el directorio del usuario, bajo *.tin/index*. Esto puede ser costoso en términos de espacio en disco, así que es posible que quiera Vd. mantener una sola copia para todos los usuarios. Esto se puede lograr haciendo a *tin* setuid **news**, por ejemplo, o algún otro usuario sin privilegios.[4] *tin* guardará todos los ficheros índice bajo */var/spool/news/.index*. Para los accesos a ficheros o salidas al shell, volverá a ser del usuario real que lo invocó.[5]

Una solución mejor es instalar el demonio indexador *tind*, que se ejecuta como tarea de fondo y actualiza regularmente los ficheros índice. Sin embargo, este demonio no se incluye en ninguna distribución para Linux, así que tendrá que compilarlo Vd. mismo. Si está Vd. trabajando con una red local con un servidor central de noticias, puede ejecutar *tind* en el servidor, y hacer que los clientes reciban los índices por NNTP. Esto, por supuesto, requiere una extensión del NNTP. Los parches necesarios para que *nntpd* soporte esta extensión se incluyen en las fuentes de *tin*.

[2]Escrito por Rich Skrenta.

[3]El tiempo se reduce drásticamente si el servidor NNTP crea las hebras por sí mismo y permite al cliente recibir estos datos. Un servidor que permite hacer esto es INN-1.4, por ejemplo.

[4]Sin embargo, *no* utilice el usuario **nobody**. Como norma, ningún fichero o programa debería ser asociado con este usuario.

[5]Esta es la razón por la que se obtendrán feos mensajes de error al invocar a tin como superusuario. De todas formas, no se debería trabajar como **root**, es una cuenta para administración.

La versión de *tin* incluida en algunas distribuciones de Linux no tiene soporte NNTP, pero la mayoría sí lo incorporan. Cuando se le invoca como *rtin* o con la opción `-r`, *tin* trata de conectar con el servidor especificado en el fichero */etc/nntpserver* o en la variable de entorno `NNTPSERVER`. El fichero *nntpserver* simplemente contiene el nombre del servidor en una sola línea.

19.2 Configuración de *trn*

trn es también el sucesor de un programa anterior, *rn* (siglas de *read news*[6]). La "t" en su nombre significa *threaded*[7]. Fue escrito por Wayne Davidson.

Al contrario que *tin*, *trn* no tiene opción para generar bases de datos sobre las hebras en el momento de ejecución. En cambio, usa las bases de datos creadas por un programa llamado *mthreads*, el cual debe ser ejecutado regularmente desde *cron* para actualizar los ficheros índice.

No ejecutar *mthreads*, sin embargo, no significa que no se pueda acceder a los artículos nuevos, solo significa que tendrá Vd. todos esos artículos sobre "¡¡Novell compra Linux!!" esparcidos por el menú de selección de artículos en vez de una sola hebra que pueda evitar fácilmente.

Para activar la ordenación en hebras de un grupo particular, *mthreads* se invoca con la lista de grupos desde la línea de comandos. La lista se hace exactamente de la misma manera que la del fichero *sys*:

```
mthreads comp,rec,!rec.games.go
```

ordenará en hebras todos los grupos **comp** y **rec**, excepto **rec.games.go** (la gente que juega al Go no necesita bonitas hebras). 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 la lista de grupos **all**.

Si recibe Vd. las noticias durante la noche, bastaría con ejecutar *mthreads* una vez por la mañana, pero también puede hacerlo más frecuentemente si es necesario. En sistemas con un tráfico muy denso, puede ser deseable ejecutar *mthreads* como tarea de fondo (modo *daemon*). Si se le llama al arrancar con la opción `-d`, se pone como tarea de fondo, comprobando cada diez minutos si han llegado nuevos artículos, y ordenándolos si este es el caso. Para ejecutar *mthreads* como tarea de fondo, ponga la siguiente línea en el script *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 archivo de registro, llamado *mt.log* y situado en el directorio donde esté instalado *trn*.

[6] N. del T.: Leer noticias.
[7] N. del T.: Ordenado en hebras.

Los artículos viejos que ya no estén disponibles en el sistema deben ser eliminados de los ficheros índice regularmente. Por defecto, sólo los artículos cuyo número esté por debajo de la línea de flotación serán eliminados.[8] Los artículos que a pesar de estar por encima de este número hayan caducado (porque tengan el campo `Expires:` en la cabecera) pueden ser purgados usando la opción `-e` del programa *mthreads*. Cuando *mthreads* está ejecutándose como tarea de fondo, esta opción hace que use un modo mejorado de purga una vez al día, poco después de la media noche.

19.3 Configuración de *nn*

nn, escrito por Kim F. Storm, proclama ser un lector cuya última finalidad es no leer noticias. Su nombre signigica "No News"[9], y su lema es "falta de noticias, buenas noticias. *nn* es mejor".

Para alcanzar su ambiciosa meta, *nn* viene equipado con gran cantidad de herramientas de mantenimiento que no sólo permiten la creación de hebras, 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 estos 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 *daemon*, invocado desde el script *rc.news* o *rc.inet2*. 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 subscripciones 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 ponemos *!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 sus bases de datos. El primero es actualizar la base comprobando los directorios de los grupos, y desechando las entradas cuyo artículo correspondiente ya no esté disponible. Este es el método por defecto obtenido al invocar a *nnmaster* con la opción `-E`. Es razonablemente rápido, a menos que se haga por NNTP.

[8] Tenga en cuenta que Cnewsno actualiza su línea de flotación automáticamente; hay que ejecutar *updatemin* para ello. Véase el capítulo 17.

[9] N. del T.: Sin noticias.

El segundo método actúa exactamente como la opción por defecto 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. 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 tarea de fondo, 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 resumen, lea la página de manual de *nnmaster*.

nnmaster se guía usando un fichero llamado *GROUPS*, situado en */usr/local/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.[10] Los indicadores permitidos y sus efectos también vienen detallados en la página de manual de *nnmaster*.

[10] Esto se debe a que el orden debe coincidir con el del fichero binario *MASTER*.

Apéndice A

Un Cable de Impresora para 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.

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:

`D0`	2—15	`ERROR`
`D1`	3—13	`SLCT`
`D2`	4—12	`PAPOUT`
`D3`	5—10	`ACK`
`D4`	6—11	`BUSY`
`GROUND`	25—25	`GROUND`
`ERROR`	15— 2	`D0`
`SLCT`	13— 3	`D1`
`PAPOUT`	12— 4	`D2`
`ACK`	10— 5	`D3`
`BUSY`	11— 6	`D4`

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 solo de los extremos.

Apéndice B

Ejemplos de Archivos de Configuración para *smail*

Este apéndice muestra ejemplos de archivos de configuración para un sistema UUCP terminal en una red de área local. Están basados en los archivos de ejemplo que se incluyen en la distribución de las fuentes de *smail-3.1.28*. Aún cuando se intenta explicar someramente cómo trabajan dichos archivos, le aconsejo que lea la página de manual de *smail(8)*, que trata acerca de estos archivos con gran detalle. Una vez que Ud. comprendió la idea básica de la configuración de *smail*, merece la pena leerla. ¡Es fácil!

El primer archivo que se muestra es el *routers*, mediante el cual se establecen los encaminadores de *smail*. Cuando *smail* tiene que enviar un mensaje a una dirección dada, prueba con las direcciones de todos los encaminadores, uno por vez, hasta que concuerda con la de uno de ellos. La concordancia significa que el encaminador encuentra el nodo de destino en su base de datos, sea en el archivo *paths*, en el */etc/hosts*, o en cual sea el mecanismo de encaminamiento que se utilice.

Las entradas en los archivos de configuración de *smail* siempre comienzan con un nombre único que identifica el encaminador, transporte, o programa de entrega local. Luego le sigue una lista de atributos que definen su comportamiento. Esta lista consta de un conjunto de atributos globales, tales como el *controlador* utilizado, y atributos privados que sólo tienen sentido para ese controlador particular. Los atributos están separados mediante comas. El conjunto de atributos globales se separa de los privados mediante un punto y coma.

Intentemos clasificar estas distinciones. Supongamos que Ud. quiere mantener dos archivos de alias de caminos distintos; uno que contiene la información de encaminamiento para su dominio, y otro que almacena la información de encaminamiento global, generada probablemente por los mapas de UUCP. Con *smail*, puede especificar dos encaminadores en el archivo *routers*, y ambos utilizarán el controlador *pathalias*. Este controlador busca los nombres de nodo en la base de datos `pathalias`.

```
#
# base de datos de alias de caminos para el encaminamiento dentro del dominio
domain_paths:
```

```
        driver=pathalias,          # busca el nodo en el archivo de caminos
        transport=uux;             # si lo encuentra lo envia a traves de UUCP

        file=paths/domain,         # el archivo es /usr/lib/smail/paths/domain
        proto=lsearch,             # el archivo no esta ordenado (busqueda lineal)
        optional,                  # si el archivo no existe, no importa
        required=vbrew.com,        # buscar solo los nodos tipo *.vbrew.com

#
# base de datos de pathalias para encaminamientos fuera de nuestro dominio
world_paths:
        driver=pathalias,          # busca el nodo en el archivo de caminos
        transport=uux;             # si lo encuentra, lo envia a traves de UUCP

        file=paths/world,          # el archivo es /usr/lib/smail/paths/world
        proto=bsearch,             # el archivo fue clasificado con sort(1)
        optional,                  # si el archivo no existe, no importa
        -required,                 # no es obligatorio tener dominio
        domain=uucp,               # quitar el ".uucp" final antes de la busqueda
```

El segundo atributo global que se ha mostrado en cada una de las dos entradas de *routers* define el transporte que se deberá utilizar cuando el encaminador haga concordar las direcciones. En nuestro caso, el mensaje se enviará utilizando el transporte *uux*. Los transportes se definen en el archivo *transports*, que se explica más adelante.

Se puede hacer un ajuste más fino con respecto a qué transporte se utilizará para enviar un mensaje si especifica un archivo de método en lugar del atributo *transport*. Los archivos de método son una forma de traducir los nombres de nodo a los transportes necesarios. No los trataremos aquí.

El siguiente archivo *routers* define los encaminadores para una red de área local que utiliza la biblioteca de resolución. Sin embargo, en un nodo Internet Ud. querrá utilizar un encaminador que maneje registros MX. Por lo tanto, deberá quitar los caracteres de comentario del encaminador alternativo *inet_bind* que usa el controlador BIND incorporado en *smail*.

En un contexto en el cual se utilizan UUCP y TCP/IP a la vez, puede Ud. encontrarse con el problema de que haya ciertos nodos que figuran en su archivo */etc/hosts* con los cuales se contacta sólo ocasionalmente mediante SLIP o PPP. En general, el correo hacia estos sistemas se debe enviar mediante UUCP. Para evitar que el controlador *inet_hosts* concuerde con dichas máquinas, deberá agregarlos al archivo *paths/force*. Este archivo es otra base de datos del estilo de alias de caminos, y se consulta antes de que *smail* consulte al sistema de resolución.

```
# Ejemplo de archivo /usr/lib/smail/routers
#
# force - obliga a enviar mediante UUCP a ciertos nodos, aun en el caso
#       en que esten en nuestro /etc/hosts
force:
```

```
        driver=pathalias,         # busca el nodo en el archivo de caminos
        transport=uux;            # si lo encuentra, envio a traves de UUCP

        file=paths/force,         # el archivo es /usr/lib/smail/paths/force
        optional,                 # si el archivo no existe, no importa
        proto=lsearch,            # el archivo no esta ordenado
                                  # (busqueda lineal)
        -required,                # no es obligatorio tener dominio
        domain=uucp,              # quitar el ".uucp" final antes de la busqueda

# inet_addrs - encuentra literales de dominio que contienen literales
#       de direcciones de IP, como por ejemplo janet@[191.72.2.1]
inet_addrs:
        driver=gethostbyaddr,     # controlador para encontrar literales
                                  # de dominios IP
        transport=smtp;           # enviar utilizando SMTP sobre TCP/IP

        fail_if_error,            # fallar si la direccion esta mal formada
        check_for_local,          # enviar directamente si nosotros somos
                                  # el nodo

# inet_hosts - encuentra nombres de nodo con gethostbyname(3N)
#       Quitelo de los comentarios si desea usar en su lugar la version BIND
inet_hosts:
        driver=gethostbyname,     # busca nodos con la funcion de biblioteca
        transport=smtp;           # usar SMTP de forma predeterminada

        -required,                # no es obligatorio tener dominio
        -domain,                  # no hay sufijos de dominio definidos
        -only_local_domain,       # no se restrinja a los dominios definidos

# inet_hosts - version alternativa usando BIND para acceder al DNS
#inet_hosts:
#       driver=bind,              # utilizar el controlador BIND incorporado
#       transport=smtp;           # usar TCP/IP SMTP para el envio
#
#       defnames,                 # usar busqueda de dominio estandar
#       defer_no_connect,         # intentar de nuevo si el servidor de
                                  # nombres no esta activo
#       -local_mx_okay,           #

#
# base de datos tipo pathalias para el encaminamiento dentro del dominio
domain_paths:
        driver=pathalias,         # busca el nodo en el archivo de caminos
        transport=uux;            # si lo encuentra, envio a traves de UUCP
```

```
        file=paths/domain,       # el archivo es /usr/lib/smail/paths/domain
        proto=lsearch,           # el archivo no esta ordenado
                                 # (busqueda lineal)
        optional,                # si el archivo no existe, no importa
        required=vbrew.com,      # buscar solo los nodos tipo *.vbrew.com

#
# base de datos tipo pathalias p/encaminar hacia nodos fuera de nuestro dominio
world_paths:
        driver=pathalias,        # busca el nodo en el archivo de caminos
        transport=uux;           # si lo encuentra, envio a traves de UUCP

        file=paths/world,        # el archivo es /usr/lib/smail/paths/world
        proto=bsearch,           # el archivo fue clasificado con sort(1)
        optional,                # si el archivo no existe, no importa
        -required,               # no es obligatorio tener dominios
        domain=uucp,             # quitar el ".uucp" final antes de la busqueda


# smart_host - redireccionador de nodo inteligente parcialmente especificado
#       Si el atributo smart_path  no se define en
#       /usr/lib/smail/config, se ignorara este encaminador.
#       La variable global smart_transport tiene precedencia sobre
#       el atributo transport
smart_host:
        driver=smarthost,        # controlador para el caso especial
        transport=uux;           # si no hay otra especificacion,
                                 # envio a traves de UUCP

        -path,                   # usar la variable del archivo
                                 # de configuracion smart_path
```

El manejo del correo para las direcciones locales se configura en el archivo *directors*. Éste se construye de la misma manera que el archivo *routers*, y consta de una lista de entradas que definen los redirectores respectivos. Los redirectores *no* envían los mensajes, sino que solamente realizan todas las redirecciones que sean posibles, por ejemplo a través de alias, reenvío de correo y cosas por el estilo.

Cuando se envía correo a una dirección local, como **janet**, *smail* pasa el nombre del usuario a todos los redirectores del módulo de entrega local, uno por vez. Si un redirector concuerda, entonces o bien especifica el transporte a través del cual el mensaje debe enviarse (por ejemplo, el nombre de archivo del buzón del usuario) o, si no, genera una nueva dirección (por ejemplo al evaluar un alias).

Por las cuestiones de seguridad involucradas, los redirectores generalmente realizan varios controles para ver si los archivos que se usan son archivos sensibles del sistema. Las direcciones que se obtienen a partir de medios dudosos (por ejemplo desde un archivo *aliases* con permisos de escritura para todo el mundo) se indican como inseguras. Algunos controladores de transporte se negarán a utilizar dichas direcciones, por ejemplo el transporte que envía mensajes a un archivo.

Además, *smail* también *asocia un usuario* con cada dirección. Cualquier operación de lectura o escritura se realizan operando con los permisos y privilegios del usuario correspondiente. Para enviar un mensaje a, por ejemplo el buzón de **janet**, la dirección está asociada por supuesto a **janet**. Las otras direcciones, tales como las que se obtienen del archivo *aliases*, tienen otros usuarios asociados a ellas, por ejemplo, el usuario **nobody**.

Para más detalles de estas características, refiérase por favor a la página de manual de *smail(8)*.

```
# Ejemplo de archivo /usr/lib/smail/directors

# aliasinclude - expande las direcciones ":include:filename"
#       producidas por los archivos de alias
aliasinclude:
        driver=aliasinclude,     # use este controlador para caso especial
        nobody;                  # si es inseguro, acceder al archivo como
                                 # usuario nobody

        copysecure,              # obtener los permisos desde el
                                 # redireccionador de alias
        copyowners,              # obtener los propietarios a partir del
                                 # redireccionador de alias

# forwardinclude - expande las direcciones ":include:filename"
#       producidas por los archivos de reenvio (forward)
forwardinclude:
        driver=forwardinclude,   # use este controlador de caso especial
        nobody;                  # si es inseguro, acceder al archivo como
                                 # usuario nobody

        checkpath,               # controlar la accesibilidad del camino
        copysecure,              # obtener los permisos desde el
                                 # redireccionador de reenvios
        copyowners,              # obtener los propietarios desde el
                                 # redireccionador de reenvios

# aliases - buscar las expansiones de alias almacenadas en la base de datos
aliases:
        driver=aliasfile,        # redireccionador de alias de proposito general
        -nobody,                 # de manera predeterminada, todas las direcciones
                                 # estan siempre asociadas a nobody
        sender_okay,             # no quitar el remitente de las expansiones
        owner=owner-$user;       # los problemas se mandan a la direccion
```

```
                                # del propietario

        file=/usr/lib/aliases,    # predeterminado: compatible con sendmail
        modemask=002,             # no debe ser de escritura para todo el mundo
        optional,                 # si el archivo no existe, no importa
        proto=lsearch,            # archivo ASCII sin ordenar

# dotforward - expande los archivos .forward de los directorios 'home'
#              de los usuarios
dotforward:
        driver=forwardfile,       # redireccionador de reenvios de
                                  # proposito general
        owner=real-$user,         # los problemas se envian al buzon del usuario
        nobody,                   # usar usuario nobody, si es inseguro
        sender_okay;              # nunca se quita el remitente en la expansion

        file=~/.forward,          # archivo .forward en los directorios 'home'
        checkowner,               # el usuario puede ser el propietario
                                  # de este archivo
        owners=root,              # o el root puede serlo
        modemask=002,             # no debe ser de escritura para todo el mundo
        caution=0-10:uucp:daemon, # no correr como root o daemon
        # hay que ser extremadamente cuidadoso con los directorios 'home'
        # que se acceden remotamente
        unsecure="~ftp:~uucp:~nuucp:/tmp:/usr/tmp",

# forwardto - expande la linea "Forward to " al frente
# del archivo buzon del usuario
forwardto:
        driver=forwardfile,
        owner=Postmaster,         # errores al Postmaster
        nobody,                   # usar usuario nobody, si es inseguro
        sender_okay;              # no quitar remitente en la expansion

        file=/var/spool/mail/${lc:user}, # posicion del buzon del usuario
        forwardto,                # habilitar el control "Forward to "
        checkowner,               # el usuario puede ser propietario
                                  # de este archivo
        owners=root,              # o el root puede serlo
        modemask=0002,            # bajo System V, el grupo mail puede escribirlo
        caution=0-10:uucp:daemon, # no corra como root o daemon

# user - encuentra usuarios en el nodo local con envio a sus respectivos buzones
user:   driver=user;              # controlador para encontrar nombres de usuario

        transport=local,          # el transporte local es hacia los buzones
```

```
# real_user - encuentra nombres de usuario cuando estan prefijados
#             con la cadena "real-"
real_user:
        driver=user;            # controlador para encontrar nombres de usuario

        transport=local,        # el transporte local es hacia los buzones
        prefix="real-",         # por ejemplo, encontrar real-root

# lists - expande listas de correo almacenadas debajo de /usr/lib/smail/lists
lists:  driver=forwardfile,
        caution,                # marcar todas las direcciones con prudencia
        nobody,                 # y luego asociarlas al usuario nobody
        sender_okay,            # NO quitar el remitente
        owner=owner-$user;      # el propietario de la lista

        # pasar a minusculas el nombre de la lista de correo
        file=lists/${lc:user},
```

Después de encaminar o redireccionar un mensaje, *smail* lleva el mensaje al transporte especificado por el encaminador o redireccionador que concordó con la dirección. Estos transportes están definidos en el archivo *transports*. Nuevamente, se define un transporte mediante un conjunto de opciones globales y privadas.

La opción más importante que se define para cada entrada se refiere al controlador que realiza el transporte, por ejemplo el controlador *pipe*, que invoca el comando especificado en el atributo *cmd*. Además de éste, existen una cierta cantidad de atributos globales que puede utilizar un transporte, que realizan varias transformaciones en la cabecera del mensaje, y posiblemente en su cuerpo. El atributo *return_path*, por ejemplo, hace que el transporte inserte un campo *return_path* en la cabecera del mensaje. El atributo *unix_from_hack* hace que se preceda toda ocurrencia de la palabra `From` al principio de una línea con el signo >.

```
# Ejemplo de archivo /usr/lib/smail/transports

# local -  envia correo a los usuarios locales
local:  driver=appendfile,       # agrega el mensaje al archivo
        return_path,             # incluir un campo Return-Path:
        from,                    # proveer una linea de sobre con From_
        unix_from_hack,          # insertar > antes de From en el cuerpo
        local;                   # usar formatos locales para el envio

        file=/var/spool/mail/${lc:user}, # posicion del archivo buzon
        group=mail,              # el grupo propietario del archivo
                                 # para System V
        mode=0660,               # el grupo mail puede acceder
```

```
        suffix="\n",              # agregar un cambio de linea extra

# pipe - enviar el correo a traves de comandos de shell
pipe:   driver=pipe,              # encauzar el mensaje hacia otro programa
        return_path,              # incluir un campo Return-Path:
        from,                     # proveer una linea de sobre con From_
        unix_from_hack,           # insertar > antes de From  en el cuerpo
        local;                    # usar formatos locales para el envio

        cmd="/bin/sh -c $user",   # enviar las direcciones al shell Bourne
        parent_env,               # info de entorno a partir de la direccion
                                  # del padre
        pipe_as_user,             # usar user-id asociado con la direccion
        ignore_status,            # ignore un status de salida distinto de cero
        ignore_write_errors,      # ignore errores de escritura,
                                  # por ejemplo: tuberia cortada
        umask=0022,               # umask para el proceso hijo
        -log_output,              # no registrar stdout/stderr en el
                                  # archivo de registro

# file - enviar el correo a archivos
file:   driver=appendfile,
        return_path,              # incluir un campo Return-Path:
        from,                     # proveer una linea de cabecera con From_
        unix_from_hack,           # insertar > antes de From en el cuerpo
        local;                    # usar formatos locales para el envio

        file=$user,               # el nombre del archivo se toma de
                                  # la direccion
        append_as_user,           # usar el user-id asociado con la direccion
        expand_user,              # expandir ˜ y $ dentro de la direccion
        suffix="\n",              # agregar un cambio de linea extra
        mode=0600,                # poner los permisos en 600

# uux - envios al programa rmail de una instalacion UUCP remota
uux:    driver=pipe,
        uucp,                     # usar formatos de direcciones estilo UUCP
        from,                     # proveer una linea de sobre con Form_
        max_addrs=5,              # maximo 5 direcciones por invocacion
        max_chars=200;            # maximo 200 caracteres en direccion

        cmd="/usr/bin/uux - -r -a$sender -g$grade $host!rmail $(($user)$)",
        pipe_as_sender,           # que los registros uucp registren al llamador
        log_output,               # guardar las salidas de error para
                                  # los mensajes rebotados
#       defer_child_errors,       # reintentar si uux retorna un error
```

```
# demand - envios al programa rmail remoto, el sondeo es inmediato
demand: driver=pipe,
        uucp,                   # usar formatos de direcciones estilo UUCP
        from,                   # proveer una linea de sobre con Form_
        max_addrs=5,            # maximo 5 direcciones por invocacion
        max_chars=200;          # maximo 200 caracteres en direccion

        cmd="/usr/bin/uux - -a$sender -g$grade $host!rmail $(($user)$)",
        pipe_as_sender,         # que los registros uucp registren al llamador
        log_output,             # guardar las salidas de error para
                                # los mensajes rebotados
#       defer_child_errors,     # reintentar si uux retorna un error

# hbsmtp - half-baked BSMTP. Los archivos de salida deben procesarse
#       regularmente y enviarse via UUCP.
hbsmtp: driver=appendfile,
        inet,                   # usar direccionamiento RFC 822
        hbsmtp,                 # SMTP por lotes sin HELO ni QUIT
        -max_addrs, -max_chars; # no hay limite en el numero de direcciones

        file="/var/spool/smail/hbsmtp/$host",
        user=root,              # el archivo es propiedad de root
        mode=0600,              # solo legible-escribible por root.

# smtp - envios utilizando SMTP sobre TCP/IP
smtp:   driver=tcpsmtp,
        inet,
        -max_addrs, -max_chars; # no hay limite en el numero de direcciones

        short_timeout=5m,             # timeout para operaciones breves
        long_timeout=2h,              # timeout para operaciones SMTP
                                      # de mayor duracion
        service=smtp,                 # conectar a este puerto de servicio
# Para uso en internet: descomente las siguientes 4 lineas
#       use_bind,                     # resolver MX y registros A multiples
#       defnames,                     # usar busqueda de dominio estandar
#       defer_no_connect,             # reintentar si el servidor de nombres
#                                     # no esta activo
#       -local_mx_okay,               # no usar MX con el sistema local
```

Apéndice C

The GNU General Public License

Printed below is the GNU General Public License (the *GPL* or *copyleft*), under which Linux is licensed. It is reproduced here to clear up some of the confusion about Linux's copyright status—Linux is *not* shareware, and it is *not* in the public domain. The bulk of the Linux kernel is copyright ©1993 by Linus Torvalds, and other software and parts of the kernel are copyrighted by their authors. Thus, Linux *is* copyrighted, however, you may redistribute it under the terms of the GPL printed below.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave, Cambridge, MA 02139, USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

C.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

C.2 Terms and Conditions for Copying, Distribution, and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program 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.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public 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.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO

USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

C.3 Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> ⟨*one line to give the program's name and a brief idea of what it does.*⟩ Copyright ©19yy ⟨*name of author*⟩
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author Gnomovision
comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.  This
is free software, and you are welcome to redistribute it under certain
conditions; type ‘show c’ for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> ⟨*signature of Ty Coon*⟩, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Glosario

[Nota: Este glosario está lógicamente incompleto, así que estoy abierto a cualquier sugerencia.]

Una de las mayores dificultades del estudio de las redes de ordenadores, es recordar todas las abreviaturas y términos que rodean la teleinformática. Aquí se tratará de mostrar los términos que se han usado en este libro, junto a una pequeña explicación.[1]

ACU
: Automatic Call Unit, *Unidad de Llamada Automática*. Un módem.[2]

ARP
: Address Resolution Protocol, *Protocolo de Resolución de Direcciones*. Se utiliza para conocer la correspondencia entre direcciones IP y direcciones Ethernet.

ARPA
: Advanced Research Project Agency, posteriormente DARPA. En Castellano sería algo así como *Instituto de Proyectos Avanzados de Investigación*. Es la creadora de Internet.

ARPANET
: Antecesora de lo que hoy es Internet: era una red experimental norteamericana, fundada por el DARPA (Defense Advanced Research Project Agency) que es como el instituto ARPA pero con fines militares.

Assigned Numbers
: Título del *RFC* publicado regularmente y que lista la correspondencia convenida entre números de puerto y similares, con respecto a su significado. Por ejemplo, incluye los números de servicios *bien conocidos* (well known) como el puerto de *rlogin* o *SMTP*. La última versión de este RFC es la 1340.

BBS
: Bulletin Board System, *Sistema de Boletín electrónico*. Es un nodo de noticias y correo al que se accede por teléfono.

BGP
: Border Gateway Protocol, *Protocolo para Pasarelas de Extremo*. Es un protocolo que sirve para intercambiar información de encaminamiento entre los sistemas autonómos.

[1] N. del T.: Algunos términos han sido incluidos por el traductor, y otros actualizados a los tiempos que corren. Algunas traducciones de los nombres de protocolos pueden no ser correctas, así que ruego nos hagan llegar cualquier sugerencia al respecto.

[2] También puede significar *un adolescente con un teléfono en sus manos*.

BIND — Berkeley Internet Name Domain, servidor de *Nombres de Dominios de Berkeley.* Es una implementación del servicio DNS.

BNU — Basic Networking Utilities, *Utilidades Básicas de Red.* Es el paquete de aplicaciones UUCP más usual en este momento; que también se conoce como UUCP de HoneyDanBer. El nombre se deriva de sus autores: P. Honeyman, D.A. Novitz, y B.E. Redman.

BSD — Berkeley Software Distribution. Es una versión de UN*X producida en la Universidad de Berkeley.

CCITT — Comiteé Consultatif International de Télégraphique et Téléphonique, *Comité Consultivo Internacional de Telefonía y Telegrafía*[3]. Es una organización internacional que estudia los estándares para los servicios de telefonía, redes, etc.

CSLIP — Compressed Serial Line IP, *IP por Línea Serie con Compresión.* Protocolo para el intercambio de paquetes IP por una línea serie, añadiendo compresión de cabeceras a la mayor parte de los datagramas TCP/IP.

demonio de encaminamiento — La topología de las redes grandes no es estática y resulta difícil el mantenimiento de adaptación de todos los nodos a esos cambios. Por ello se trata de automatizar mediante *demonios de encaminamiento* usando protocolos como RIP.

DNS — Domain Name System, *Sistema de Nombres de Dominios.* Es una base de datos distribuida que se utiliza en Internet para obtener las direcciones IP asociadas a los nombres.

EGP — External Gateway Protocol, *Protocolo para Encaminadores Externo.* Es un protocolo con el mismo fin que el mencionado BGP, es decir, intercambiar información de encaminamiento entre sistemas autónomos.

Ethernet — En sentido coloquial, es el nombre que se le da a una instalación de red (*La ethernet de la Universidad*, etc). En realidad, Ethernet es parte de un conjunto de estándares propuestos por el IEEE. Los dispositivos de conexión Ethernet utilizan un solo cable, normalmente coaxial, que permite transferencias de 10 Mbits/segundo. El protocolo utilizado determina cómo las máquinas tienen acceso al cable.[4]

FQDN — Fully Qualified Domain Name, *Nombre de Nodo Totalmente Calificado.* Es un nombre de nodo completo, es decir, incluyendo el dominio al que pertenece: se encuentra en la base de datos del sistema DNS.

FTP — File Transfer Protocol, *Protocolo de Transferencia de Ficheros.* Es uno de los protocolos de TCP/IP más conocido, pues es el que se usa para enviar ficheros de un lugar a otro de la red.

---

[3] Hoy conocido como ITU, International Telecommunication Union o *Unión Internacional de Telecomunicaciones.*

[4] Además, el *protocolo* de Ethernet usado normalmente con TCP/IP no es exactamente el que se definió en la norma IEEE 802.3, ya que donde éste habla del campo *longitud*, el otro habla del campo *tipo.*

FYI	"For Your Information", *para su información.* Documentos más o menos informales que tratan sobre temas de Internet.
GMU	Groucho Marx University, *la Universidad de Groucho Marx.* Universidad que nos hemos inventado en este libro para los ejemplos que ilustran los conceptos que se dan a conocer en éste.
GNU	GNU's not Unix, *GNU No es Unix.* Acrónimo *recursivo* del proyecto de la Fundación del Software Libre (FSF) que pretende proporcionar un conjunto de utilidades tipo UN⋆X, que pueden usarse y distribuirse libremente. Todo el software de GNU está cubierto por la Licencia Pública General de GNU, o *Copyleft.* En el apéndice C se reproduce dicha Licencia.
HoneyDanBer	Nombre de una variedad de UUCP. Véase BNU.
host	En general, un *host* es un nodo de red, es decir, algo que es capaz de recibir y enviar algún tipo de mensajes en la red. Normalmente será un ordenador, pero puede ser también un terminal X o una impresora de red.
ICMP	Internet Control Message Protocol, *Protocolo de Mensajes de Control de Internet.* Es un protocolo utilizado por TCP/IP para devolver información de errores al nodo que envía el paquete, etc.
IEEE	Institute of Electrical and Electronics Engineers, *Instituto de Ingenieros Eléctricos y Electrónicos.* Es otra organización que establece estándares. Desde el punto de vista del usuario de UNIX, lo que interesa es que establece los estándares POSIX que definen aspectos que van desde las llamadas al sistema hasta qué comandos de administración deben existir. Además, el IEEE desarrolló las especificaciones para las redes de difusión Ethernet, las de paso de testigo en bus (TokenBus) y en anillo (TokenRing). Finalmente, la representación interna de coma flotante más utilizada también es un estándar propuesto por el IEEE.
IETF	Internet Engineering Task Force, *Grupo de Ingeniería en Internet.*
internet	Red de redes: unión de pequeñas redes de ordenadores.
Internet	El nombre que se le ha dado a la Internet distribuida por todo el mundo.
IP	Internet Protocol, *protocolo de internet.*
ISO	International Standards Organization, *Organización Internacional de Estandarización.*
ISDN	Integrated Services Digital Network, *Red Digital de Servicios Integrados.* Nueva tecnología para comunicaciones que utiliza circuitos digitales, y está llamada a sustituir a la telefonía tradicional.
HTML	HiperText Markup Language, *Lenguaje de Descripción de Hipertexto.* Es el lenguaje en el que se escriben los documentos que luego se van a transmitir en la WWW

(véanse, WWW y HTTP).

HTTP — HiperText Transfer Protocol, *Protocolo para Transferencia de HiperTexto.* Es el protocolo que se utiliza para transmitir documentos de la WWW a los programas de usuario conocidos como *navegadores.*[5]

LAN — Local Area Network, *red de área local.*

MX — Mail Exchanger, *Intercambiador de Correo.* Un tipo de registro DNS que se utiliza para indicar cuál es el servidor de correo asociado a un dominio.

NFS — Network File System, *Sistema de Ficheros en Red.* Es un protocolo estándar junto con un conjunto de programas que permite acceder a los datos de discos remotos de manera transparente, como si fueran discos locales.

NIS — Network Information System, *Sistema de Información de Red.* Es una aplicación basada en RPC que permite compartir ficheros de configuración como el */etc/passwd* entre distintas máquinas. Véase también YP.

NNTP — Network News Transfer Protocol, *Protocolo de Transferencia de Noticias.* Se utiliza para transmitir noticias a través de conexiones TCP.

nombre canónico de nodo — En un sistema DNS, es el nombre principal de una máquina, es decir, establecido mediante un registro DNS tipo A, y es el que se devuelve cuando se hace una petición de nombre a partir de la dirección IP.

octeto — En la Internet, este término se refiere a la cantidad de ocho bits. Se usa en lugar de *byte* (palabra) debido a que en algunas se tienen palabras de más de 8 bits.

OSI — Open Systems Interconnection, *Interconexión de Sistemas Abiertos.* Es un estándar del ISO acerca del software de red.

path — En UUCP es sinónimo a *ruta* (camino seguido por los mensajes). Es también lo que hemos llamado *ruta de signos de admiración.*

PLIP — Parallel Line IP, *IP por Línea Paralela.* Es un protocolo que permite intercambiar paquetes IP usando el puerto de la impresora.

puerto, TCP o UDP — En TCP y UDP, un puerto es lo que en OSI se conoce como *punto de acceso al servicio.* Antes de que un proceso acceda o dé un servicio de red, debe pedir un puerto (bind). Junto con la dirección IP, identifica totalmente los dos extremos de una conexión TCP.

portmapper — El portmapper o mapeador de puertos, es el programa que traduce entre números de programa RPC y los puertos TCP o UDP por los que escuchan dichos programas.

PPP — Point-to-point Protocol, *Protocolo Punto-a-Punto.* Es un protocolo rápido y flexi-

[5] Algunas voces se han alzado ya, culpando a HTTP y a su descuidado diseño, de la actual lentitud de la red.

ble, usado para intercambiar paquetes IP e IPX a través de una línea serie (teléfono) o incluso sobre un protocolo de nivel de enlace HDLC para su uso en la RDSI.

RARP
: Reverse Address Resolution Protocol, *Protocolo de Resolución Inversa de Direcciones.* Permite a las máquinas preguntar a la red por su dirección IP cuando se ponen en marcha.

Red de almacenamiento-y-reenvío
: Opuesto al concepto de conmutación de paquetes. Estas redes transfieren paquetes pero no tienen conexiones permanentes. En su lugar, las máquinas conectan de vez en cuando con el otro extremo y transmiten los datos de una vez. Esto requiere que el nodo tenga capacidad de almacenamiento.

Red de Conmutación de Paquetes
: Es un tipo de red formada por nodos que se limitan a intercambiarse paquetes entre sí de forma segura, de tal forma que en ella se establecen circuitos virtuales permanentes o semipermanentes.

red de difusión
: Es una red que permite a las estaciones enviar datagramas y que sean vistos por varias estaciones a la vez.

registro de recurso, RR
: Es la unidad de información en la base de datos DNS. Cada registro es de un tipo, por ejemplo, contiene la dirección IP de una máquina (registro tipo A), o el servidor de correo de un dominio (registro tipo MX).

resolutor, sistema de resolución
: Es un conjunto de funciones de biblioteca que permiten a los programas obtener las direcciones IP de las máquinas por su nombre, y viceversa.

resolución inversa
: El acto de obtener un nombre dada la dirección IP. En DNS, dichas direcciones se devuelven como parte del dominio **in-addr.arpa**.

RFC
: Request for Comments, *Petición de Sugerencias.* Son documentos que describen los estándares de Internet. Su nombre procede de que inicialmente eran solo "propuestas" de los autores, para ser discutidas.

RIP
: Routing Information Protocol, *Protocolo de Información de Encaminamiento.* Es un protocolo de encaminamiento dinámico, útiles en redes no muy grandes.

ruta
: Secuencia de máquinas por las que una unidad de información debe pasar para llegar al nodo destino. El proceso de encontrar una ruta apropiada se conoce como *encaminamiento.*

ruta de signos de admiración
: En las redes UUCP, la ruta de un sistema a otro se nota mediante nombres y signos de admiración. Por ejemplo, **nodo0!nodo1!nodo2!nodo3** denota que la ruta al sistema **nodo3** pasa por **nodo0**, **nodo1** y **nodo2**.

RPC
: Remote Procedure Call, *Llamada a Procedimiento Remoto.* Es el mecanismo por el cual se pide la ejecución de procedimientos en máquinas remotas.

RR	Abreviatura de *registro de recurso*, en DNS.
RS-232	Estándar habitual para enlaces serie.
RTS/CTS	Nombre coloquial que recibe el control de flujo por hardware, realizado entre dos máquinas que se comunican usando RS-232. El nombre procede de los dos circuitos utilizados, RTS ("Request To Send") y CTS ("Clear To Send").
RTM Internet Worm	Un programa a modo de virus que utiliza ciertas características de VMS y Unix BSD 4.3 para propagarse por la red. RTM son las siglas de su autor (Robert T. Morris).
sitio	Conglomerado de nodos que desde fuera se ven como un nodo de la red. Por ejemplo, desde fuera vemos como nodo de Internet a la Universidad de Groucho Marx, cuando en realidad es una red de ordenadores bastante compleja[6].
SLIP	Serial Line IP, *IP por Línea Serie*. Es un protocolo para intercambiar paquetes IP usando una línea serie. Véase también CSLIP.
SMTP	Simple Mail Transfer Protocol, *Protocolo Simple para Transferencia de Correo*. Es un protocolo usado para transportar correo mediante conexiones TCP, así como correo por lotes en UUCP (BSMTP).
SOA	Start of Authority, *Inicio de Autoridad*. Es un tipo de registro que existe en la base de datos del DNS.
System V	Una versión de UN⋆X.
TCP	Transmission Control Protocol, *Protocolo de Control de la Conexión*. Es un protocolo de red *orientado a la conexión*.
TCP/IP	Abreviatura usada para denotar los protocolos de Internet.
UDP	User Datagram Protocol, *Protocolo de Datagramas de Usuario*. Es un protocolo de red *no orientado a la conexión*.
UUCP	Unix to Unix Copy, *Copiador de Unix a Unix*. Es un conjunto de comandos para transporte en red, usado en redes de marcado telefónico.
UUCP Versión 2	Versión evolucionada de UUCP.
cerveza virtual	Es la bebida preferida de todo Linuxero[7]. La primera vez que recuerdo verlo escrito fue en la nota sobre la versión 0.98.X del núcleo de Linux, donde Linus incluía la "Oxford Beer Trolls" en los créditos como sitio desde donde enviaban cervezas virtuales a cualquiera.
well-known services, servicios bien-conocidos	El término se usa para referirse a los servicios de red habituales como *telnet* o *ftp*.

[6] N. del T: Actualmente, por *sitio* también se conocen a los servidores de WWW o FTP.

[7] O casi cualquier Linuxero. El traductor, por ejemplo, prefiere la cerveza real.

Técnicamente, son servicios que tienen un número de puerto asignado en el RFC de Asignación de Números.

YP — Yellow Pages, *Páginas Amarillas.* Es el antiguo nombre de lo que hoy se conoce como NIS, dado que el término Yellow Pages es marca registrada de la British Telecom. Sin embargo, muchas utilidades de NIS se nombran empezando con *yp*, como los comandos *ypcat* o *ypwhich*.

WWW — World Wide Web, *la Telaraña de Ámbito Mundial.* Es el servicio que ha catapultado a la Internet a la fama. El la WWW se distribuye documentación de todo tipo, en formato de hipertexto (usando el lenguaje de descripción HTML), con imágenes, sonido y acceso a ficheros. La WWW se está convirtiendo en el *escaparate* con el que muchas empresas pueden dar a conocer sus productos al mundo cibernético.

Bibliografía Comentada

Libros

A continuación se incluye una lista de libros a los que puede referirse si le interesa saber más sobre alguno de los temas cubiertos por la Guía de Administración de Redes con Linux. No se trata de una lista completa o sistemática, simplemente son libros que he leído y que encuentro bastante útiles. Se agradece cualquier información o mejora de esta lista.

Libros sobre Internet en general

[**Kehoe92**] Brendan P. Kehoe: *Zen and the Art of the Internet.* .

La "Zen" fue probablemente la primera guía sobre la Internet, o al menos una de las primeras. Es una introducción para el usuario novato a las costumbres, los servicios y el folklore de la Internet. Se trata de un tomo de unas 100 páginas que cubre temas que van desde el correo electrónico o las noticias Usenet al virus Worm de la Internet. Está disponible via FTP anónimo en muchos servidores FTP y puede ser distribuido e impreso libremente. También hay un tomo editado por Prentice-Hall.

Temas de Administración

[**Hunt92**] Craig Hunt: *TCP/IP Network Administration.* O'Reilly and Associates, 1992. ISBN 0-937175-82-X.

Si la Guía de Administración de Redes con Linux no es suficiente, consígase este libro. Trata todo tipo de temas, desde cómo conseguir una dirección de IP o la solución de problemas de la red o la seguridad.

Se centra en establecer TCP/IP, configuración de la interface, establecimiento de las tablas de encaminamiento y resolución de nombres. Incluye una descripción detallada de las opciones disponibles en los demonios `routed` y `gated`, que implementan encaminamiento dinámico.

También describe la configuración de aplicaciones y demonios de red como `inetd`, los comandos `r`, NIS, y NFS.

El apéndice contiene una referencia detallada de `gated`, y `named`, y una descripción del proceso de configuración del `sendmail` de Berkeley.

[Stern92] Hal Stern: *Managing NIS and NFS.* O'Reilly and Associates, 1992. ISBN 0-937175-75-7.

Se trata de un libro que complementa a "TCP/IP Network Administration" de Craig Hunt. Cubre con detalle el uso de NIS (Sistema de información de red) y NFS (sistema de ficheros de red), incluyendo la configuración de automontado de sistemas de ficheros y PC/NFS.

[OReilly89] Tim O'Reilly y Grace Todino: *Managing UUCP and Usenet, 10th ed.* O'Reilly and Associates, 1992. ISBN 0-93717593-5.

Es el estándar para redes UUCP. Cubre la versión 2 de UUCP y la de BNU. Le servirá de ayuda desde el principio cuando establezca su nodo UUCP, dándole consejos prácticos y soluciones a múltiples problemas, como la verificación de conexiones o cómo escribir buenas macros para conectar mediante chat. También trata temas más exóticos como el establecimiento de un nodo UUCP *móvil* o las sutilezas presentes en los distintos tipos de UUCP.

La segunda parte del libro trata del software de noticias de red y Usenet. Explica la configuración tanto de Bnews (version 2.11) como C-News, y sirve de introducción a las tareas de mantenimiento de noticias de red.

[Spaf93] Gene Spafford y Simson Garfinkel: *Practical UNIX Security.* O'Reilly and Associates, 1992. ISBN 0-937175-72-2.

Se trata de un libro imprescindible para cualquiera que administre sistemas con acceso de red o de otra índole. El libro discute temas relevantes en seguridad de ordenadores, incluyendo las características básicas de seguridad física que proporciona UN⋆X. A pesar de que es necesario atender a la seguridad en todas las areas del sistema, la explicación de redes y seguridad es la parte más interesante del libro en nuestro contexto. Además de las técnicas de seguridad que conciernen a los servicios de Berkeley (*telnet*, *rlogin*, etc), NFS y NIS, también trata de sistemas más sofisticados como Kerberos del MIT, RPC Seguro de Sun y el uso de los cortafuegos como protección frente a ataques desde la Internet.

[AlbitzLiu92] Paul Albitz y Cricket Liu: *DNS and BIND.* O'Reilly and Associates, 1992. ISBN 1-56592-010-4.

Este libro resulta útil para todos aquellos que administren un servicio de nombres DNS. Explica con gran detalle las características de DNS y da ejemplos que explican las opciones que a primera vista resultan extrañas en BIND. Me divirtió mucho leerlo y aprendí muchísimo de él.

[NISPlus] Rick Ramsey: *All about Administering NIS+.* Prentice-Hall, 1993. ISBN 0-13-068800-2.

Conocimientos Básicos

A continuación hay una lista de libros que pueden resultar de interés para aquellas personas que quieran saber más sobre *cómo* funciona TCP/IP y sus aplicaciones pero no quieren leer los RFCs.

[Stevens90] Richard W. Stevens: *UNIX Network Programming.* Prentice-Hall International, 1990. ISBN 0-13-949876-X.

Se trata probablemente del libro más usado sobre programación en redes TCP/IP que, al mismo tiempo, explica las entrañas de los protocolos de Internet.[8]

[Tanen89] Andrew S. Tanenbaum: *Computer Networks.* Prentice-Hall International, 1989. ISBN 0-13-166836-6[9].

Este libro trata de temas sobre redes en general. A través del Modelo de Referencia OSI, explica los problemas de diseño de cada una de las capas, y los algoritmos que pueden usarse para solucionarlos. Para cada capa, compara diferentes implementaciones incluyendo la de ARPAnet.

El único problema de este libro es que el uso abundante de abreviaturas dificulta a veces la comprensión de lo que quiere decir el autor[10]. Aunque seguramente ésta es una característica inherente a los libros de redes.

---

[8]Stevens acaba de escribir otro libro sobre TCP/IP, titulado *TCP/IP Illustrated, Volume 1, The Protocols* y publicado por Addison Wesley aunque no he tenido tiempo de echarle un vistazo.

[9]El número de ISBN en otros paises puede ser diferente.

[10]N. del T.: De este libro existe una traducción al castellano editada por la propia Prentice-Hall. Sin embargo, no se la recomendaría a nadie: se entiende peor que la original en Inglés.

[Comer88] Douglas R. Comer: *Internetworking with TCP/IP: Principles, Protocols, and Architecture.* Prentice-Hall International, 1988.

HOWTOs

A continuación hay un extracto del índice HOWTO-INDEX, version 2.0 (17 de Marzo de 1994), escrito por Matt Welsh.

¿Cuáles son los HOWTOs de Linux?

Los HOWTOs de Linux son pequeños documentos disponibles electrónicamente que describen con detalle cietos aspectos de cómo configurar o usar el sistema Linux. Por ejemplo, existe una Installation HOWTO, que cuenta cómo instalar Linux, una Mail HOWTO, que describe como establecer y configurar el servicio de correo electrónico en Linux. Otros ejemplos incluyen el NET-2-HOWTO (lo que antes eran las NET-2-FAQ) y el Printing HOWTO.

La información de los HOWTOs es generalmente más detallada y profunda de lo que se pueda incluir en las FAQ de Linux. Por esta razón las FAQ están siendo reescritas. Gran parte de la información contenida en ellas será relegada a los diferentes documentos HOWTO. Las FAQ quedan como una lista más breve de las preguntas más habituales sobre Linux, cubriendo escuetamente temas específicos. La mayoría de la información más "útil" en las FAQ se incluirá en los HOWTOs.

Los HOWTOs son documentos extensos, parecidos a unas FAQ, aunque en general no responden a un formato de pregunta-respuesta. Sin embargo muchos HOWTOs incluyen una pequeña sección de FAQ al final. Por ejemplo, las NET-2-FAQ simplemente cambiaron el nombre por NET-2-HOWTO, ya que nunca fueron escritas en este formato. Aunque se cite al NET-2-HOWTO como NET-2-FAQ en muchos sitios, se trata del mismo documento.

¿Dónde se consiguen los HOWTOs de Linux?

Los HOWTOs pueden obtenerse via FTP anónimo desde cualquiera de los siguientes servidores:

- *sunsite.unc.edu:/pub/Linux/docs/HOWTO*
- *tsx-11.mit.edu:/pub/linux/docs/HOWTO*

así como en cualquiera de los mirrors de estos servidores citados en las META-FAQ de Linux.

El índice, impreso a continuación, contiene los HOWTOs existentes en la actualidad.

Los HOWTOs se publican regularmente en los grupos de noticias **comp.os.linux** y **comp.os.linux.announce**. Muchos son publicados también en **news.answers**. Por tanto se pueden obtener los HOWTOs en el archivo del **news.answers** en el servidor **rtfm.mit.edu**.

Índice de HOWTOs

A continuación hay una relación de los HOWTOs disponibles actualmente[11].

- Linux Busmouse HOWTO, escrito por **mike@starbug.apana.org.au** (Mike Battersby). Información sobre la compatibilidad del ratón bus en Linux.
- Linux CDROM HOWTO, escrito por **tranter@software.mitel.com** (Jeff Tranter). Información sobre la compatibilidad de los lectores de CD-ROM en Linux.
- Linux DOSEMU HOWTO, escrito por **deisher@enws125.EAS.ASU.EDU** (Michael E. Deisher). HOWTO sobre el emulador de MS-DOS bajo Linux, DOSEMU.
- Linux Distribution HOWTO, escrito por **mdw@sunsite.unc.edu** (Matt Welsh). Lista de distribuciones de venta por correo y otros servicios comerciales.
- Linux Ethernet HOWTO, escrito por Paul Gortmaker **gpg109@rsphysse.anu.edu.au**. Información sobre la compatibilidad del hardware Ethernet en Linux.
- Linux Ftape HOWTO, escrito por **ftape@mic.dth.dk** (Linux ftape-HOWTO maintainer). Información sobre los grabadores de cinta y su compatibilidad con Linux.
- Linux HOWTO Index, escrito por **mdw@sunsite.unc.edu** (Matt Welsh). Índice de los documentos HOWTO en Linux.
- Linux Hardware Compatibility HOWTO, escrito por **erc@apple.com** (Ed Carp). Lista casi completa del hardware que funciona con Linux.
- Linux Installation HOWTO, escrito por **mdw@sunsite.unc.edu** (Matt Welsh). Describe cómo obtener e instalar el Linux.
- Linux JE-HOWTO, escrito por Yasuhiro Yamazaki **hiro@rainbow.physics.utoronto.ca**. Información sobre JE, un conjunto de extensiones de Linux en lengua japonesa.
- Linux Keystroke HOWTO, escrito por Zenon Fortuna (**zenon@netcom.com**). Describe como asociar macros a las diferentes teclas en Linux.
- Linux MGR HOWTO, escrito por **broman@Np.nosc.mil** (Vincent Broman). Información sobre la interface gráfica MGR para Linux.
- Linux Electronic Mail HOWTO, escrito por **vince@victrola.wa.com** (Vince Skahan). Información sobre servidores y clientes de correo electrónico disponibles en Linux.
- Linux NET-2 HOWTO, escrito por **terryd@extro.ucc.su.oz.au** (Terry Dawson). Explica cómo configurar las comunicaciones via TCP/IP en Linux y en particular SLIP, PLIP y PPP.
- Linux News HOWTO, escrito por **vince@victrola.wa.com** (Vince Skahan). Información sobre el software servidor y cliente de noticias USENET disponibles para Linux.
- Linux PCI-HOWTO, escrito por Michael Will **michaelw@desaster.student.uni-tuebingen.de**. Información sobre la compatibilidad de la arquitectura PCI en Linux.

[11] N. del T.: Actualmente hay muchos más HOWTOs, que cubren nuevos e interesantes temas como el PCI, el uso de disquetes ZIP, etc. No deje de revisar las nuevas distribuciones de Linux por si acaso.

- Linux Printing HOWTO, escrito por **gtaylor@cs.tufts.edu** (Grant Taylor). Trata del sofware de impresión en Linux.
- Linux SCSI HOWTO, escrito por Drew Eckhardt **drew@kinglear.cs.Colorado.EDU**. Información sobre la compatibilidad del manejador SCSI de Linux.
- Linux Serial HOWTO, escrito por **gregh@cc.gatech.edu** (Greg Hankins). Información sobre el uso de dispositivos serie y sobre el software de comunicaciones.
- Linux Sound HOWTO, escrito por **tranter@software.mitel.com** (Jeff Tranter). Software y hardware, relacionado con el sonido, disponible para el sistema operativo Linux.
- Linux Term HOWTO, escrito por Bill Reynolds **bill@goshawk.lanl.gov**. Explica el uso del paquete de comunicaciones "*term*" con sistemas Linux.
- Linux Tips HOWTO, escrito por Vince Reed **reedv@rpi.edu**. HOWTO con trucos y consejos varios sobre Linux.
- Linux UUCP HOWTO, escrito por **vince@victrola.wa.com** (Vince Skahan). Información sobre software UUCP para Linux.
- Linux XFree86 HOWTO, escrito por **geyer@polyhymnia.iwr.uni-heidelberg.de** (Helmut Geyer). Explica la instalación del servidor XFree86 (X11R5) para Linux.

Asuntos varios y cuestiones legales

Si tiene usted alguna pregunta, por favor no dude en ponerse en contacto por correo electrónico con **mdw@sunsite.unc.edu**. El proceso de reescritura de las FAQ de Linux esta siendo coordinado por Ian Jackson, **ijackson@nyx.cs.du.edu**, con ayuda de otras personas.

Salvo mención expresa en contra, los derechos de copyright de los documentos HOWTO de Linux pertenecen a sus respectivos autores. Los documentos HOWTO de Linux pueden ser reproducidos de forma completa o en parte, en cualquier medio físico o electrónico sin permiso del autor. Igualmente, cualquier traducción o trabajo derivado está permitido sin necesidad de un permiso expreso del autor. La redistribución con fines comerciales está permitida e incluso se anima a ello; sin embargo, el autor querría ser avisado si semejante distribución tuviera lugar.

En resumen, queremos promocionar la diseminación de esta información a través de tantos canales como sea posible. No obstante, queremos retener los derechos de copyright sobre los documentos HOWTO y nos gustaría ser notificados de cualquier plan de redistribuir los HOWTOs. Si tiene alguna pregunta, por favor póngase en contacto con Matt Welsh, el coordinador de los HOWTOs de Linux en **mdw@sunsite.unc.edu**.

Los HOWTOs en Castellano

Como contábamos en la introducción, actualmente existe otro grupo que trabaja junto con LuCAS; y se ocupa de traducir los HOWTOs al Castellano; que se han venido a denominar **COMOs**.

Son muchos los COMOs disponibles actualmente, con lo que creemos que le resultará útil a numerosos lectores.

Los COMOs traducidos se pueden encontrar en el mismo servidor principal de LuCAS (`http://lucas.hispalinux.es/`) aunque su servidor principal es el siguiente:

- **WWW**: `http://www.insflug.org/`
- **FTP**: `ftp.insflug.org`

La coordinación de este grupo corre a cargo de Francisco José Montilla, **pacopepe@insflug.org**.

RFCs

A continuación hay una lista de los RFCs mencionados a lo largo del libro. Todos los RFCs están disponibles via FTP anónimo en los servidores **nic.ddn.mil**, **ftp.uu.net**. Para obtener un RFC via correo electrónico, envíe un mensaje a **service@nic.ddn.mil**, incluyendo la petición `send RFC-`*`numero`*`.TXT` en el *asunto*[12].

1340	Números Asignados, *Postel, J.*, y *Reynolds, J.* El RFC de Números Asignados define el significado de los números empleados por los diferentes protocolos, tales como los números de puertos estándar atendidos por los servidores TCP y UDP, así como los números de protocolo contenidos en la cabecera del datagrama IP.
1144	Compresión de cabeceras TCP/IP para enlaces tipo serie de baja velocidad, *Jacobson, V.* Este documento describe el algoritmo usado para comprimir las cabeceras TCP/IP en CSLIP y PPP. ¡Realmente merece la pena leerlo!
1033	Guía de los Administradores de Dominio, *Lottor, M.* Junto con los RFCs complementarios, RFC 1034 y RFC 1035, es la fuente de información más completa sobre DNS, el sistema de resolución de nombres de dominio.
1034	Nombres de Dominios - Conceptos y Características, *Mockapetris, P.V.* Complemento de la RFC 1033.
1035	Nombres de Dominios - Implementación y Especificación, *Mockapetris, P.V.* Complemento de la RFC 1033.
974	Distribución de Correo Electrónico y el Sistema de Dominios, *Partridge, C.* Este RFC describe la distribución de correo en Internet. Lea esto para conocer la historia completa de los registros MX.
977	Protocolo de Transferencia de Noticas de Red, *Kantor, B.*, y *Lapsley, P.* Definición del NNTP, el método más común de transporte de noticias de red en Internet.
1094	NFS: Especificación del Protocolo de Sistema de Fichero de Red *Nowicki, B.* Especificación formal de NFS y del protocolo de montaje (version 2).

[12]N. del T.: *subject* en programas de correo anglosajones.

1055 No-estándar de Transmisión de Datagramas de IP sobre lineas de Comunicación Serie: SLIP, *Romkey, J.L.* Describe SLIP, el Protocolo de Comunicación via linea Serie de Internet.

1057 RPC: Especificación del Protocolo de Llamada a Procedimiento Remoto: Version 2, *Sun Microsystems, Inc*

1058 Protocolo de Información de Encaminamiento, *Hedrick, C.L.* Describe RIP, protocolo usado para el intercambio de información de rutas en LANs y MANs.

821 Protocolo Simple de Transferencia de Correo,*Postel, J.B.* Define SMTP, el protocolo de transporte de correo sobre TCP/IP.

1036 Estándar para el Intercambio de mensajes USENET, *Adams, R.* y *Horton, M.R.* Este RFC describe el formato de los mensajes de noticias USENET, y cómo son intercambiados tanto en la Internet como en redes UUCP. Se espera que pronto haya una revisión de este protocolo.

822 Estándar sobre el Formato de mensajes de texto en la Internet de ARPA, *Crocker, D.* Es la más completa fuente de sabiduría sobre el correo que cumple el estándar RFC. Todo el mundo lo conoce, aunque son pocos los que realmente lo han leído.

968 Twas The Night Before Startup[13], *Cerf, V.* ¿Quién dice que los heroes de las redes no son poetas?

[13]N. del T.: Al tratarse de un poema, hemos preferido no traducir su título

Índice de Materias