Instituto Nacional de
Estadística e Informática

Sub-Jefatura de Informática

ADMINISTRACION DE REDES CON LINUX

NORMATIVIDAD Y PROMOCION

Julio 2001

Indice

Presentación

Las redes de datos parecen estar acercándose a la situación de los televisores y los hornos microondas, debido a la publicidad que Internet esta teniendo, y el hecho de que la gente de opinión mas respetada en el mundo de la Informática, este todo el tiempo hab***LAN***do de las virtudes de la red. Es decir, Internet esta 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" y la Sociedad de la Información. 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 área 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 computadoras privadas. 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 datos 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 si ha estado en el negocio de las redes por mucho tiempo y seguramente continuará haciéndolo 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 re-implementación completa, desde cero.

El núcleo de Linux fué escrito enteramente por Linus Trovalds, 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 ***Unix***, y fué escrito por el Profesor Andrew S. Tanenbaum.

Linux esta cubierto por la Licencia Publica General (GPL) GNU, que permite la libre distribución del código ("free software"). Superando sus males de joven, y atraído por una siempre creciente base de programas de aplicación gratuitos, se esta 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 ***Unix***, 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 sutiles notas de instalación y README s (LEAME s). Estos daban a los magos de ***Unix***, moderadamente experimentados, suficiente información para instalarlo y ejecutarlo con exito, 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 "¿Cuál es el significado de la vida, el universo y todo lo demas?", estos manuales intentan cubrir la mayoría de aspectos del uso y funcionamiento de un sistema Linux sin necesidad de graduarse previamente en ***Unix***.

Entre los logros del LDP están el Installation and Getting Started Guide1, escrita por Matt Welsh, el Kernel Hacker's Guide2 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 del manual para la mayoría de las llamadas al sistema y librerías de C. La System Administrators' Guide3 escrita por Lars Wirzenius, esta aun en estado alpha. También se esta 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 mas 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 paginas 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 mas frecuentes. Generalmente proporcionan la información mas precisa y actualizada disponible sobre la materia.

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 Quin***LAN***. Después de seis meses de discusión, el grupo presento un diseño que muestra una estructura de sistema de ficheros coherente y define la localización de los programas mas 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; solo 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 Quin***LAN***, coordinador del grupo FSSTND puede ser localizado en quin***LAN***@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. ¿Cuales son las opciones de A?. Podría:

1) ir a la choza de B,
2) hacerse con un tambor mas 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 computadoras que hablan entre si a través de vastas conexiones de cables, fibras ópticas, microondas, y otros medios parecidos, para coordinar la reunión del sábado.

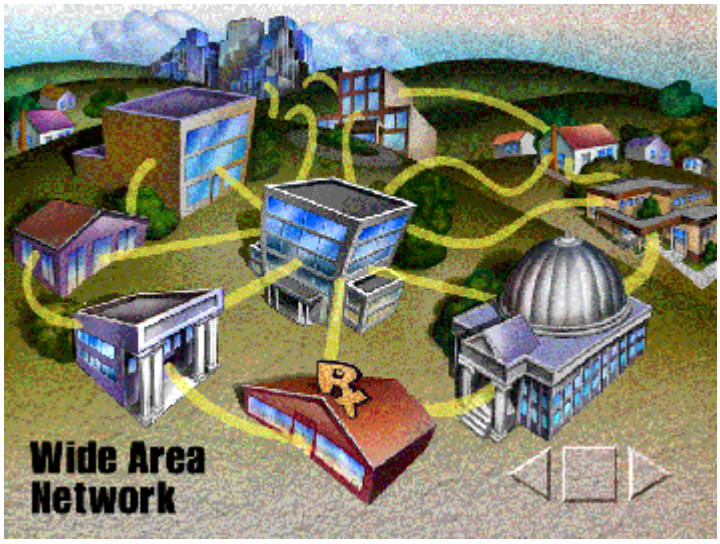
A continuación trataremos sobre las maneras en que esto se realiza, pero olvidándonos de los cables, así como de la parte de la reunión.

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

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

La comunicación seria 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 mas 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 mas 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. Hoy día, ***UUCP*** ya no esta confinado en el entorno ***Unix***. Hay versiones comerciales disponibles para diversas plataformas, incluyendo Amiga ***OS, 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 limite 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 si 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, aun 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 la computadora conectada 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.

1.2.1 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.

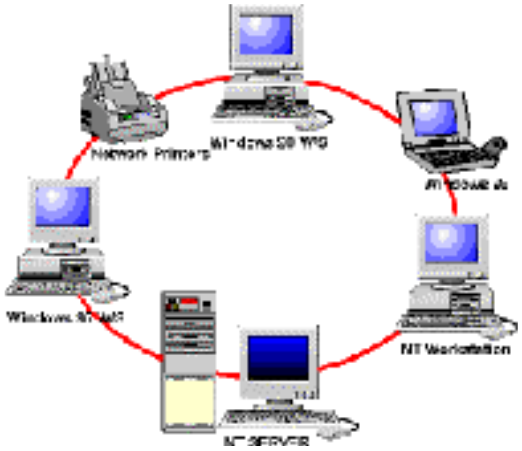
Suponga que le esta permitido que su maquina acceda a un nodo hipotético llamado ***ineidtt***, y le va ha hacer ejecutar el comando de impresión lpr para Ud. Entonces, podría escribir lo siguiente en su línea de comandos para que le imprima este libro en swim:

$ ***uux*** -r ***ineidtt***!lpr !netguide.dvi

Esto hace que ***uux***, un comando del repertorio ***UUCP***, planifique un trabajo para ***ineidtt***. 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 estos colaboren. Asumiremos que ***ineidtt***, el nodo del ejemplo anterior, tiene un enlace ***UUCP*** con UNAC, el cual mantiene un gran numero de aplicaciones ***Unix***. Para transferir el fichero tripwire-1.0.tar.gz hasta su máquina debería indicarlo así:

$ ***UUCP*** -mr
ineidtt!unac!~/security/tripwire-1.0.tar.gz trip.tgz

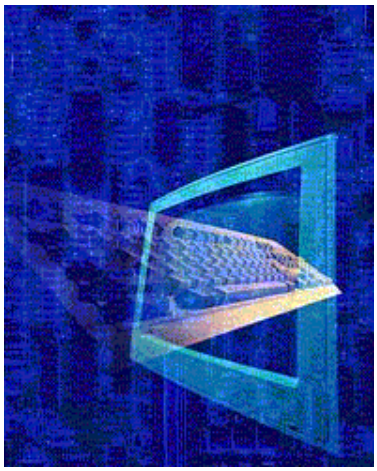


El trabajo creado pedirá a ***ineidtt*** que traiga el fichero desde groucho, y lo envié hasta su maquina, donde ***UUCP*** lo almacenara en trip.tgz y le notificara por correo la llegada del fichero. Esto ocurrirá en tres pasos. Primero, su maquina envía el trabajo a ***ineidtt***. La siguiente vez que ***ineidtt*** establezca contacto con unac, se transferirá el fichero de unac a ***ineidtt***. El ultimo paso es la transferencia del mismo desde ***ineidtt*** hasta su maquina.

Los servicios más importantes que proporcionan las redes ***UUCP*** hoy en día son él Correo electrónico y las noticias. Lo introduciremos brevemente y después lo veremos en mas detalle. El correo electrónico - e-mail para abreviar - le permite intercambiar mensajes con usuarios de nodos remotos sin tener realmente que saber como acceder a estos nodos.

La tarea de dirigir un mensaje desde su maquina 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 reenviara entonces el mensaje a otro nodo, y seguirá así, hasta que alcance el nodo destino.

La mejor forma de definir el servicio de noticias es considerarlo como un sistema de tablón de anuncios distribuido. Muy a menudo, éste termino se refiere a las noticias de Usenet, que es, con mucho, la mas conocida red de intercambio de noticias, con un numero de nodos participantes estimado en 120.000. 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***. Estos escribieron algunos scripts, creando el primer sistema de noticias en red. En 1980, esta red



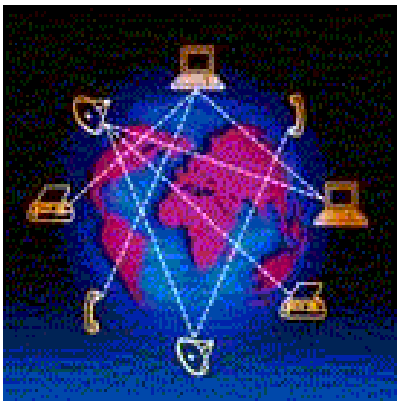
conectaba duke, unc, y phs, y dos Universidades de Carolina del Norte, de forma aislada. Usenet creció mas todavía posteriormente. Aunque su origen fue como una red basada en ***UUCP***, ya no esta limitada a un único tipo de redes.

La unidad básica de información es el articulo, 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 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 empaquetándolos en varios lotes . Estos 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 publico. Generalmente podrá acceder a ellos llamando con ***UUCP***, accediendo como usuario invitado, y transfiriéndose los archivos desde un área de ficheros públicamente accesible. Estas cuentas de invitado tienen, a menudo, un nombre de acceso y password como ***UUCP***/n***UUCP*** o algo similar.

1.3 Redes *TCP/IP*



Aunque ***UUCP*** puede resultar una elección razonable para enlaces de red mediante llamada de bajo costo, hay muchas situaciones en las que su técnica de almacenamiento y reenvió se muestra demasiado inflexible, por ejemplo en Redes de Area Local (***LAN***s, o RALs). Estas redes están compuestas generalmente por un pequeño número de máquinas localizadas en el mismo edificio, o incluso en la misma p***LAN***ta, que están interconectadas para proporcionar un entorno de trabajo homogéneo. 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 ***Unix*** y muchos no-***Unix*** 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. Esta 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 mas allá de la propia Internet. Las mas destacables son las redes locales ***Unix***, 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, como la Universidad Nacional del Callao (UNAC) situada, en la provincia Constitucional del Callao, en Lima-Perú. En esta universidad, la mayoría de las facultades 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 o denominada comúnmente como línea dedicada.

Supongamos una máquina Linux conectada a una ***LAN*** de nodos ***Unix*** en la Facultad de Economía, y su nombre es Econo. Para acceder a un nodo de la Facultad de Contabilidad, por ejemplo conta, introducirá el siguiente comando:

```
$ rlogin conta.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 César, y su clave. Entonces dispondrá de un shell de conta, sobre la que puede escribir como si estuviera sentado en la consola del sistema. Tras salir del 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 este conectado a ***quark***, podria 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 este ejecutando X11 en erdos. La clave esta en que ***TCP/IP*** permite a ***quark*** y a erdos enviarse paquetes X11 en ambos sentidos para darle a Ud. la impresión de que esta 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 maquinas, de modo que aparezcan como sistemas de ficheros locales. Por ejemplo, todos los directorios "home", o personales, de los usuarios pueden estar en una maquina servidor central, desde la cual montan los directorios el resto de maquinas de la ***LAN***. El efecto de esto es que los usuarios pueden acceder a cualquier máquina, y encontrarse a si 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 maquina, y exportar estos directorios a otras maquinas.

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

Ahora echaremos una mirada mas de cerca al modo en que trabaja ***TCP/IP***. Esto es necesario para comprender como y por que 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 mas utilizado en ***LAN***s es lo que comúnmente conocemos como Ethernet. Consta de un solo cable con los nodos colgando de el 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, difiriendo 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 computador grueso requiere que realice un pequeño agujero en el cable, y conecte un transceptor utilizando un "conector vampiro". Entonces se pueden conectar uno o mas 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 tirara 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 esta por unos centavos el metro. Sin embargo, para instalaciones de gran escala, es mas 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 trafico 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 ***LAN***s. Sin embargo, pueden enlazarse varios segmentos de Ethernet entre si utilizando repetidores, puentes o encaminadores. Los repetidores simplemente copian las señales entre dos o mas segmentos, de forma que todos los segmentos juntos actúan como si fuese una única Ethernet. Debido a requisitos de tiempos, no puede haber mas de cuatro repetidores entre cualquier par de nodos de la red. Los puentes y encaminadores son mas sofisticados, analizan los datos de entrada y los reenvían solo si el nodo receptor no esta en la Ethernet local.

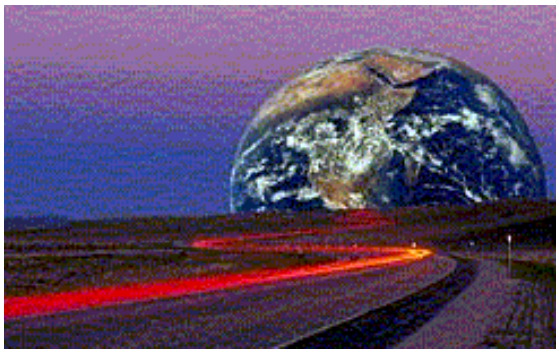
Ethernet funciona como un sistema de bus, donde un nodo puede mandar paquetes (o tramas) de hasta 1500 bytes a otro nodo de la misma Ethernet. Cada nodo se direcciona por una dirección de seis bytes grabada en el firmware de su tarjeta Ethernet. Estas direcciones se especifican generalmente como una 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 solo 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 envió, y reintentándolo 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 esta 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 envió de un número de testigos, de modo que una estación solo 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 Publicas 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),estos 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 aún otros protocolos para la transmisión de paquetes, como ***SLIP*** o ***PPP***, que se describen mas adelante.

1.3.4 El Protocolo IP (Internet Protocol)

Por supuesto, Ud. no querrá que su red este limitada a una Ethernet. Idealmente, Ud. desearía poder acceder a la red sin importarle ni el hardware del que dispone ni el numero 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 mas 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 maquina Linux, el software de red no puede mandar paquetes a ***quark*** directamente, porque no esta en la misma Ethernet. Por tanto, tiene que confiar en la pasarela para que actué como re-transmisor. 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 maquina destino.

Este esquema de envió 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

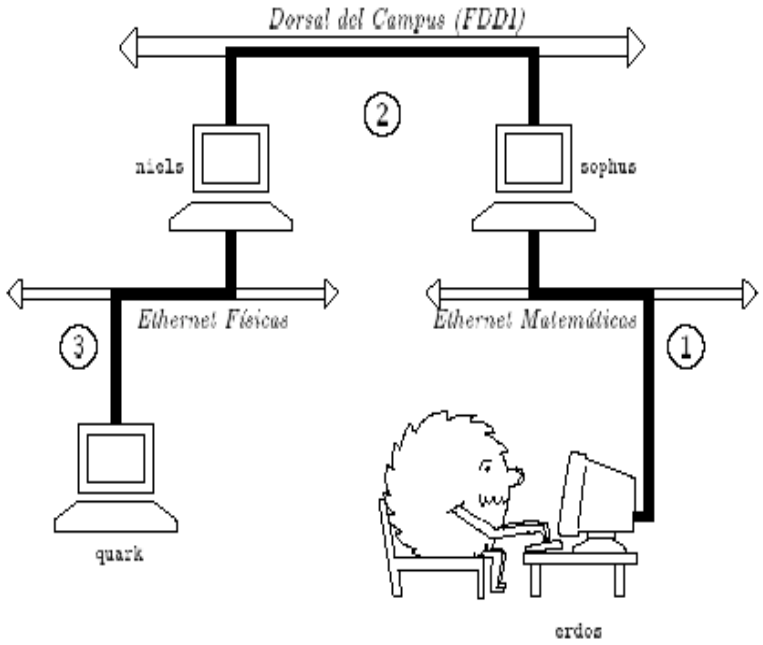
intercambio de datagramas esta gobernado por un único protocolo que es independiente del hardware utilizado: IP, o Internet Protocol. En el capitulo 2, trataremos el IP y el encaminamiento en mayor detalle.

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

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.



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

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. Un protocolo serie distinto es el *PPP*, o Point-to-Point Protocol (Protocolo Punto a Punto). ***PPP*** dispone de muchas mas características que ***SLIP***, incluyendo una fase de negociación del enlace. Su principal ventaja sobre ***SLIP*** es, sin embargo, que no se limita a transportar data gramas ***IP***, sino que se diseño 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 envió 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 si solo, no es fiable. Suponga que diez personas de su Ethernet comienzan a transferirse la última versión del StarOffice 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.

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 que 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 mas, 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 esta 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 maquina será pasado a través del canal ***TCP*** y entregado al Shell como entrada estándar.

1.3.7 El Protocolo de Data gramas 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, este 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 maquina 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 mas rápido. Sin embargo, ***UDP*** no se hizo para controlar la perdida o corrupción de paquetes. Es responsabilidad de la aplicación - en este caso ***NFS*** - tener en cuenta esto.

1.3.8 Mas 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 numero 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 especifico para obtener un servicio especifico. 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 números 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 (Numeros 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.

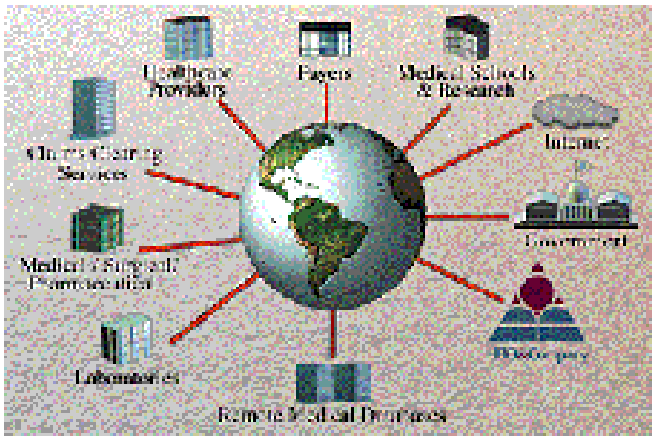
Merece la pena indicar que aunque las conexiones ***TCP*** y ***UDP*** se basan en puertos, estos números no entran en conflicto. Esto significa que el puerto ***TCP*** 513, por ejemplo, es diferente del puerto ***UDP*** 513. De hecho, estos puertos sirven como puntos de acceso para dos servicios diferentes,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

1.3.9 La Librería de Sockets

En sistemas operativos ***UNIX***, 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 mas común en el mundo ***Unix*** 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 mas 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 maquina (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 solo 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 mas 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 princ***IP***io 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 publica de Net-2d (como parte del kernel 0.99.10), y ha sido mantenida y ampliada por varias personas, muy especialmente por Alan Cox, dando lugar al Net-2Debugged. Tras una dura corrección y numerosas mejoras en el código, cambio 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 trafico de red sobre líneas serie), y PL***IP*** (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, mostrándose 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 mejoraran la versatilidad de Linux. Un controlador para ***PPP*** (el protocolo punto a punto, otra forma de enviar tráfico de red sobre líneas serie) esta en estado Beta actualmente, y otro controlador AX.25 para ham radio esta en estado Alfa. Alan Cox también ha implementado un controlador para el protocolo ***IP***X 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.

1.4.1 Diferentes Etapas de Desarrollo

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

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

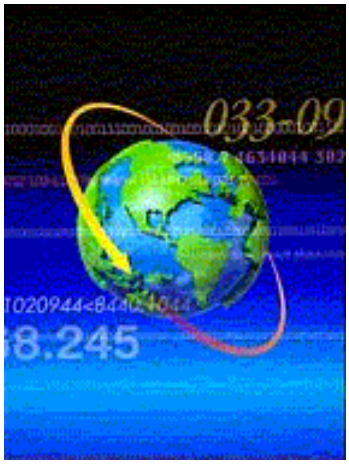
En un futuro previsible, sin embargo, Net-3 parece que llegará para quedarse. Alan trabaja actualmente en una implementación del protocolo AX.25 usado por radioaficionados.

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 solo 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 este participando en el desarrollo activo, no tendrá que preocuparse de que versión del código ***TCP/IP*** esta 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 ultima 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 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 princ***IP***almente 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. Solo tendrá que retocarlos para adecuarlos a sus necesidades y preferencias.

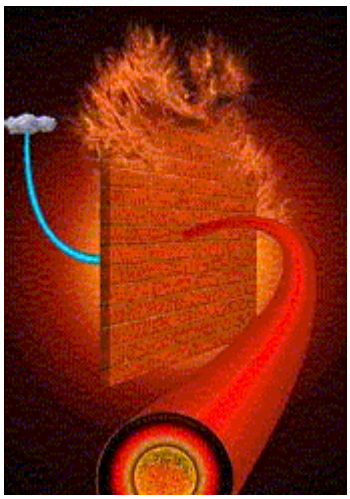
La salida de cualquiera de sus trabajos del cron se debería enviar a una cuenta de administración. Por defecto, muchas aplicaciones enviarán informes, estadísticas de uso, o resúmenes del fichero de registro a la cuenta de ***root***. Esto solo tiene sentido si accede como ***root*** frecuentemente;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

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 esta 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 Simón Garfinkel, 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 diseño. Por ejemplo, debería hacer sus programas con setuid ***root*** o alguna otra cuenta privilegiada solo si realmente lo necesitan. También, si quiere usar un servicio sólo para una aplicación muy limitada, no dude en configurarla 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 que no restringir el servicio ***TFTP*** al directorio /boot.

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 capitulo 9, presentaremos ***TCPD***, que hace esto para una variedad de aplicaciones de red.

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 esta ocurren, y no hay protección segura contra ello. Este problema afecta al software libre y a productos comerciales por igual. Sin embargo, programas que requieren privilegio especial son inherentemente más peligrosos que otros, ya que un agujero de estos puede tener consecuencias drásticas. 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 que permite comprobar ficheros vitales del sistema para ver si sus contenidos o permisos han cambiado. tripwire realiza varios checksums 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.

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 capitulo le enseñará la base con la que podrá entender los pasos para su configuración particular, pasos que son cubiertos exhaustivamente en otros capítulos.

Redes: Comunicación sin limites

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 esta ante las demás estaciones de trabajo de la red. Esta dirección es diferente del nombre de interfaz considerado anteriormente; puede realizarse la siguiente analogía: la interfaz seria 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 MTU1. Otros parámetros serán introducidos mas 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 maquinas libremente. Sin embargo, si las maquinas se conectan a

Internet, las direcciones de los ordenadores serán asignadas por una autoridad principal, el NIC2 o Centro de Información de la Red.

Para facilitar la lectura, las direcciones ***IP*** se dividen en cuatro números de 8 bits llamados octetos. Por ejemplo, si la maquina ***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 validas, 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 mas 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 numero de red esta 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.

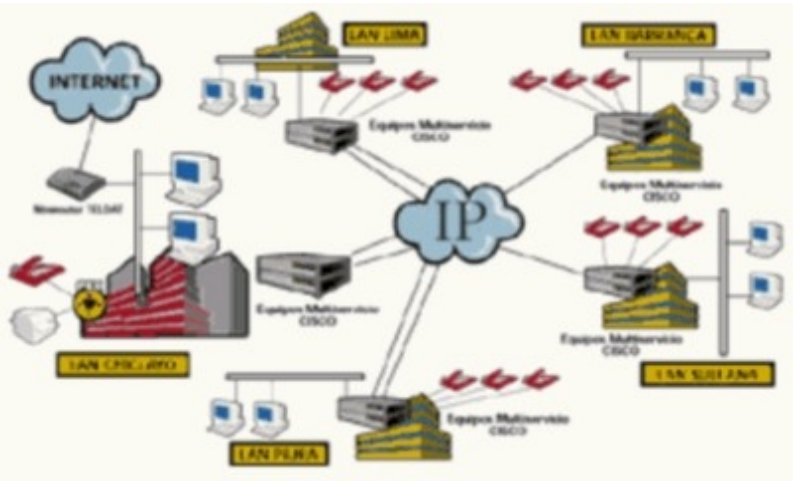
Clase B
La clase B comprende redes desde 128.0.0.0 hasta 191.255.0.0; siendo el número de red de 16 bits (los dos primeros octetos). Esto permite 16320 redes de 65024 nodos cada una.

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 t***IP***o 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 experimentales4. 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 seria 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 donde 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 solo 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 que son las direcciones ***IP***, nos preguntaremos como se utilizan en una red Ethernet. Después de todo, el protocolo Ethernet usa direcciones identificativas de 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. 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 enviara una respuesta ***ARP*** al nodo llamante, con su dirección física.

Por supuesto, le preocupara saber como 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 caché ***ARP***, para así no preguntar por lo mismo cada vez que envié 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 caché ***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 esta grabada en memoria no volátil. Para ello, se usa el Protocolo de Resolución Inversa de Direcciones o ***RARP***: 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, que permite a las maquinas sin disco conocer como ponerse en marcha en la red.

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 enviara al del país que corresponda, y este último la entregara 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 que 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 entregara al encaminador correspondiente, sin preocuparse del destino final del paquete.

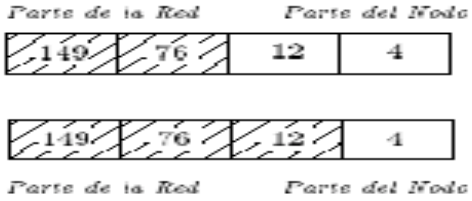
2.4.2 Subredes

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 encontraran en la misma red ***TCP/IP***.

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 data gramas 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 numero ***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 mascara de red. Es un numero de 32 bits que especifica una mascara 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 mascara 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 si mismo una subred, la 149.76.1.0. Las subredes comparten el mismo



División de una red clase B en subredes

numero, pero se usa el tercer octeto de esta para distinguir las distintas subredes. Por lo tanto, tendrán una mascara de subred igual a 255.255.255.0. En la figura 2.1 se muestra como 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 solo 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: solo ve los nodos con los que directamente este conectado (por ejemplo, en la Ethernet), mientras que a los demás los accede a través de lo que se conoce como pasarela10 , que no es mas que un nodo conectado a dos o mas 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 maquinas de la (sub)red 149.76.4.0 serian las que están en la ***LAN*** del Departamento de Matemáticas. Cuando se envía un data grama a la maquina ***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 data gramas a través de la pasarela (sophus por defecto).

Sophus se encuentra conectado a dos subredes: la del Departamento de Matemáticas y la de la dorsal del campus, accediendo a cada una a través de una interfaz diferente, respectivamente eth0 y ***FDDI0.*** Nos preguntaremos entonces, que 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, usara 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, este será el esquema de interfaces, direcciones y mascara de sophus, nuestra pasarela:

Interfaz	Dirección	Máscara
Eth0	149.76.4.1	255.255.255.0
FDDI*0*	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 comento anteriormente.

2.4.4 Tablas de Encaminamiento

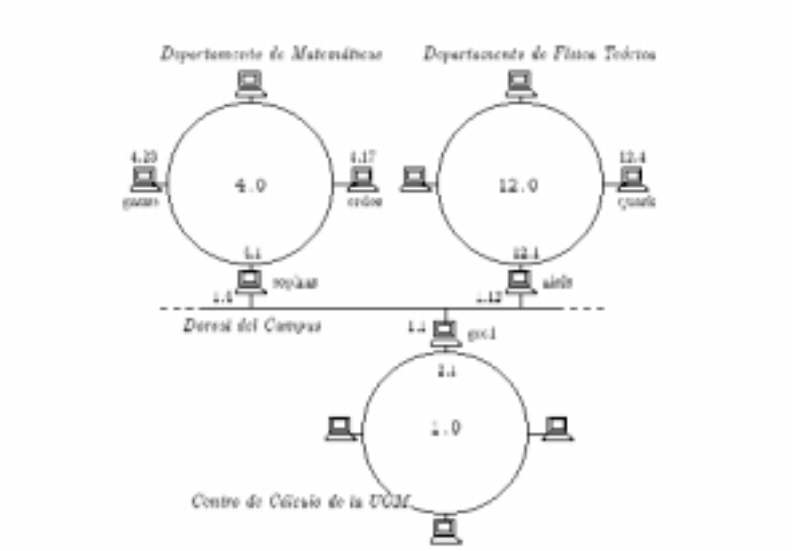
Vamos ahora a centrarnos en como se selecciona una pasarela para entregar un datagrama a una red remota.

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 esta 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, esta sería la tabla para sophus:

Vista parcial de la topología de la red de la UGM.



Red	Pasarela	Interfaz
149.76.1.0	-	***FDDI***0
149.76.2.0	*149.76.1.2*	***FDDI***0
149.76.3.0	*149.76.1.3*	***FDDI***0
149.76.4.0	-	eth0
149.76.5.0	*149.76.1.5*	***FDDI***0
…	…	…
0.0.0.0	*149.76.1.2*	***FDDI***0

Las rutas a una red a la que sophus este 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 mas eficiente construirlas a mano usando el comando route de Linux (véase el capitulo 5). Para redes más grandes, las tablas se mantienen y modifican automáticamente mediante los demonios de encaminamiento. Estos corren en nodos centrales e intercambian información de encaminamiento entre ellos para tener en todo momento las rutas "optimas" 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***. El más utilizado es ***RIP*** o protocolo de información de encaminamiento, que implementa el demonio routed de BSD. Para encaminamiento entre redes se usan protocolos ***EGP*** , como ***BGP***. Se implementan en programas como ***gated*** de la Universidad de Cornell.

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 numero de "saltos", es decir, las pasarelas que tiene que atravesar el Datagramas hasta llegar a su destino. La ruta mas corta será la elegida, y si hay 16 o más saltos se descartara por exceso de distancia.

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

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 especifica (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***, 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 mas mensajes ***ICMP***, muchos de ellos referidos a condiciones de error. Sin embargo, hay uno interesante que es el de redirección. Lo genera el modulo de encaminamiento al detectar que otro nodo esta usándolo como pasarela, a pesar de existir una ruta mucho mas corta. Por ejemplo, tras configurarse la tabla de encaminamiento de sophus, esta puede estar incompleta, conteniendo rutas a través del encaminador por defecto gcc1. Por lo tanto, los paquetes enviados inicialmente a ***quark*** iran por gcc1 en lugar de niels.

En este caso gcc1 notificara a sophus que esta usando una ruta costosa y reenviara 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 trafico 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 comento antes, el direccionamiento en ***TCP/IP*** se basa en números de 32 bits. Evidentemente, esos números no son fáciles de recordar, mientras que si lo es el nombre que se le asigna a cada maquina, 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 ***DNS16*** .

Una aplicación que desee encontrar la dirección ***IP*** correspondiente a una maquina 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 maquina, y modificarla al agregar, eliminar o modificar nodos. Pero resulta complicado cuando hay muchas maquinas ya que, en principio, cada una necesita una copia de ***/etc/hosts.***

Una solución a esto es compartir esta y otras bases de datos con el ***NIS***, o sistema de información de red, 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***.

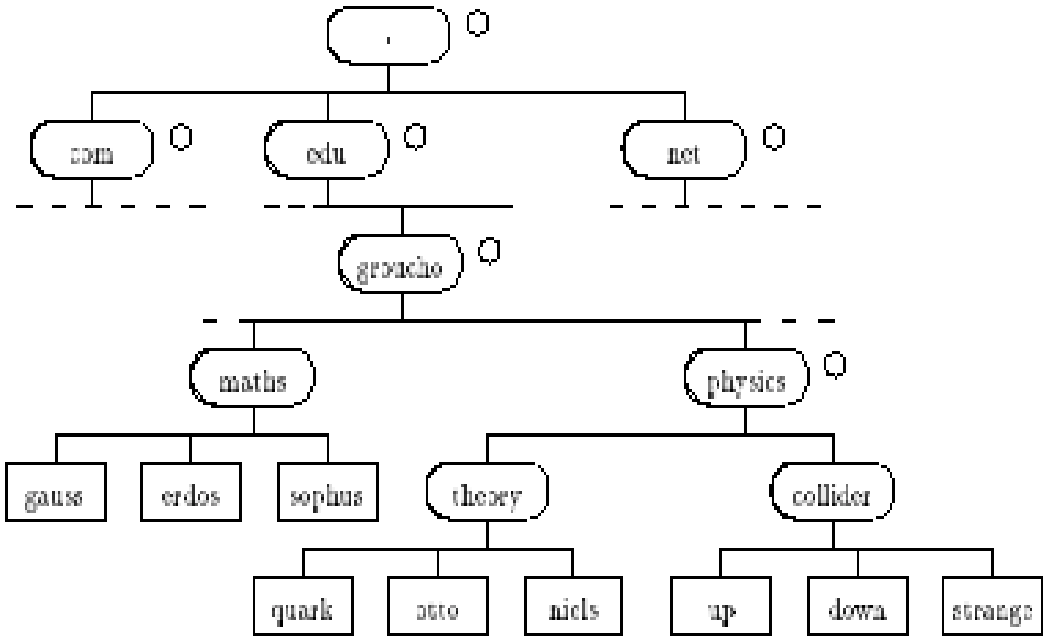
En Internet, se comenzó almacenando la información en un fichero similar al hosts, mantenido por el ***NIC***, y obtenido regularmente por los demás servidores. Cuando la red crecio comenzaron los problemas de sobrecarga de servidores, además de que el ***NIC*** tenia que ocuparse de

todos los nombres de los nodos de Internet, y evitar la duplicidad de los mismos. Por esto, en 1984 se diseño y adopto oficialmente un nuevo sistema, el ***DNS*** o sistema de nombres por dominios, diseñado por Paúl 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 maquinas del departamento de Matematicas se encontrarian dentro del dominio maths.gmu.edu.

De este modo el nombre completo de la maquina erdos será erdos.maths.gmu.edu. El nombre completo se conoce como nombre totalmente cualificado o FQDN18 , e identifica a ese nodo en todo el mundo.



Parte del espacio de dominios

En la figura se muestra una sección del espacio de dominios. La entrada de la raiz 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.

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. Fin***LAN***dia, 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 el. 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 Calculo de dicha universidad. Y si el

mantenimiento del subdominio maths.gmu.edu fuese complicado (por numero elevado de nodos, existencia de subdominios internos, etc), el Centro de Calculo 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 delego, 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 solo los nodos que mantiene directamente el Centro de Calculo, ya que los nodos del subdominio physics.groucho.edu pertenecen a la zona controlada por el Departamento de Físicas. En la figura se marca el inicio de una zona con un pequeño circulo a la derecha del nombre del dominio.

2.6.3 Búsquedas de nombres con *DNS*

Trataremos aquí el problema de como 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 estos mantiene la información de uno o varios dominios.

Para cada zona hay al menos dos (o mas) 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 este nos devolverá los datos pedidos.

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

Aparentemente esto provoca mucho trafico, 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, aun se puede mejorar algo mas. La información obtenida en una búsqueda puede que se necesite después. Por ello, el servidor de nombres local la guardara 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*** 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 sera enviada a uno de 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 dominio20 . 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 mas 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 t***IP***o asociado a el, describiendo que clase de datos contiene, y una clase indicando el t***IP***o 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 MIT21). El registro mas habitual es el de t***IP***o A, que relaciona un nombre totalmente cualificado con una dirección ***IP***.

Un nodo puede admitir mas de un nombre. Pero solo 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 capitulo posterior se trata todo esto en profundidad. Aquí nos vamos a limitar a ver algunos ejemplos. En la figura se muestra una parte de la base de datos para la zona physics.groucho.edu.

```
    ;
; información 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
    ;
    ; Física Teórica (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
    ...
```

Extracto del fichero named.hosts del departamento de Físicas.

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

```
;
; 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
 ...
```

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.

Cuandose 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.***ARP***a, 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.***ARP***a. El t***IP***o de registro para estos datos se llama PTR.

Cuando se crea una zona de autoridad suele significar que sus administradores tienen control total sobre como 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 siguiente figura se muestra el contenido del fichero de zona para el servidor de la subred 12. Los registros "importantes" de delegación se muestran en la siguiente figura.

```
;
; 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.
```

Extracto del fichero named.rev de la subred 12.

Una importante consecuencia de esto es que las zonas solo pueden crearse como súper onjuntos de redes ***IP***, y además, las mascaras de red deberán redondearse a nivel de octeto, es decir, las subredes de la ***UGM*** tendrán una mascara 255.255.255.0, y para cada subred deberá existir una zona in-addr.***ARP***a. Sin embargo, si la mascara fuese 255.255.255.128, la creación de zonas para la subred 149.76.12.128 seria imposible ya que no hay forma de decir en ***DNS*** que el dominio 12.76.149.in-addr.***ARP***a 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 Matemáticas
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 físicas, 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
...
```

Extracto del fichero named.rev de la red 149.76.

Capítulo 3 Configuración del Hardware de Red

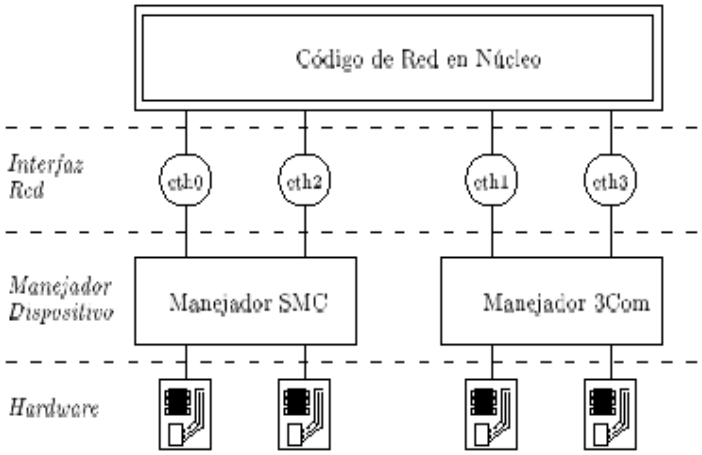
3.1 Dispositivos, Controladores, y todo lo demás

Hasta ahora, hemos estado hablando bastante sobre los interfaces de red pero sin explicar realmente que 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, esta el hardware por si 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 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 ¿que 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 ordenes y datos a la tarjeta, mientras que la tarjeta debe transmitir al controlador cualquier dato recibido.



Relación entre controladores, interfaces, y hardware.

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 ordenes 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 autoverificacion2, 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 replicas exactas de tarjetas estándar de otros fabricantes. Por otro lado, el núcleo intentara detectar un único dispositivo Ethernet al arrancar. Si se utiliza mas de una tarjeta, hay que indicárselo explícitamente al núcleo.

Otro de los parámetros que se pueden decir de forma explicita al núcleo es el canal de petición de interrupción. Los componentes hardware normalmente interrumpen al núcleo cuando tienen necesidad de que este se ocupe de ellos, por ejemplo cuando han llegado datos, 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 numero de interrupción asignado a un componente hardware se denomina su numero de petición de interrupción, o IRQ .

Como se explicaba en el capitulo 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 data gramas.

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 interfase al puerto serie.

Al arrancar, el núcleo muestra cada dispositivo que es detectado, y que 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 disc***IP***line 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

Esta indica que el núcleo ha sido compilado para ***TCP/IP***, y se incluyen los controladores para ***SLIP***, C***SLIP***, y ***PPP*** . La tercera línea antes del final indica que se ha detectado un adaptador de bolsillo D-Link e instalado como el interfase dl0. Si usted tiene un tipo diferente de tarjeta Ethernet, el núcleo mostrara 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 utilizara, pero que ocupan un espacio innecesario en el sistema de memoria ya que con el núcleo no puede hacerse swapping. Por tanto seria conveniente crear un núcleo propio, incluyendo solo 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 como 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 preguntara por una serie de configuraciones 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 pasara 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 esta 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 esta (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 activara 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]
PL***IP*** (parallel port) support (CONFIG PL***IP***) [n]

Estas preguntas conciernen a los diversos protocolos de nivel de enlace soportados por Linux. ***SLIP*** permite transportar data gramas ***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 trafico a la red a través de líneas serie. Es mucho mas flexible que ***SLIP***, y no se limita a ***IP***, sino que, una vez que se implemente, también soportara ***IP***X. Ya que el soporte para ***PPP*** ha sido incluido hace poco, esto opción puede no aparecer en su núcleo.

PL***IP*** proporciona una forma de enviar data gramas ***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 mas controladores, la lista de preguntas se hace mayor. Si desea construir un núcleo que quiera utilizar en varias maquinas, tiene la posibilidad de activar mas 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 PC***LAN*** support (CONFIG HP***LAN***) [n]
AT1500 and EN2100 (***LAN***CE and PCnet-ISA) support (CONFIG ***LAN***CE) [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 preguntara, 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 maquina 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 ***IP***X, el proceso de configuración vario 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 ***IP***X.

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 maquina como un cortafuegos. Los cortafuegos son nodos que se conectan a una o mas redes, pero no encaminan trafico entre ellos. Se utilizan normalmente para proporcionar a los usuarios en una

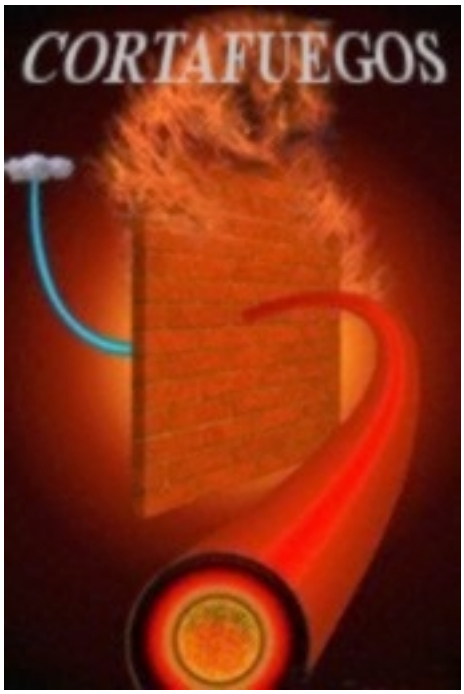
empresa acceso a Internet con un riesgo mínimo para la red interna. A los usuarios se les permitirá acceder al cortafuegos y utilizar servicios Internet, pero las maquinas de la empresa estarán protegidas de ataques externos ya que cualquier conexión entrante no puede cruzar el cortafuegos.

*

* (it is safe to leave these untouched)

PC/***TCP*** compatibility mode (CONFIG INET PC***TCP***) [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 maquinas ***Unix*** normales, pero bajara el rendimiento cuando el enlace sea lento.



Reverse ***ARP*** (CONFIG INET R***ARP***) [n]

Esta función activa R***ARP***, Protocolo de Resolución de Direcciones Inverso. R***ARP*** se utiliza en clientes sin disco y terminales X para pedir su dirección ***IP*** al arrancar. deberá activar ***RARP*** solo cuando planee que su maquina sea un servidor para este tipo de clientes. El último paquete de utilidades de red (net-0.32d) contiene una pequeña utilidad llamada r***ARP*** 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 maquinas accesibles en redes locales como Ethernet , se utilizaran paquetes mas grandes que para maquinas cuyos datos son enviados a través de enlaces de larga distancia. Si no se activa la opción SNARL, el núcleo asumirá como locales solo a aquellas redes con las que en ese momento tenga una interfase. 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 interfase con solo una o dos subredes. Si se activa la opción SNARL, el núcleo asumirá todas las subredes como locales y utilizara paquetes grandes cuando se comunique con todos los nodos del campus.

Si no desea utilizar tamaños de paquete pequeños para enviar datos a maquinas especificas (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 formula de Nagle es un método heurística para evitar enviar paquetes ***IP*** particularmente pequeños, también denominados pequegramas. Los pequegramas son utilizados normalmente por herramientas de red interactivos que transmiten pulsaciones únicas de teclas, como telnet o rsh (remote shell). Los peque gramas 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 circunstancias. Es recomendable desactivar el algoritmo de Nagle si tiene graves problemas por paquetes perdidos.

The ***IP***X protocol (CONFIG ***IP***X) [n]

Activa la capacidad de soportar el protocolo ***IP***X, el protocolo de transporte utilizado por Novell Networking; que sigue todavía bajo desarrollo, y aun no es realmente útil. Una ventaja de esto será cuando algún día se intercambien datos con utilidades ***IP***X basadas en DOS, y encaminen trafico 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 vació (dummy). La siguiente pregunta aparece hacia el comienzo de la sección de controladores de dispositivos:

Dummy net driver support (CONFIG DUMMY) [y]

El controlador vació no hace realmente gran cosa, pero es bastante útil en maquinas aisladas o conectadas mediante ***SLIP***. Es básicamente un interfase en bucle cerrado. La razón de tener este tipo de interfase es que en las maquinas que se conectan con ***SLIP*** que no disponen de Ethernet, es necesario tener un interfase que continuamente maneje las direcciones ***IP***. Esto se discute mas profundamente en las sección La interfase Comodín del capitulo 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 mas de un interfase, 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 data

grama 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 mas de uno.

Ethn Tarjeta Ethernet n -sima. Este es el nombre de interfase genérico para la mayoría de las tarjetas Ethernet.

Dln Esta interfase accede a un adaptador de bolsillo D-Link DE00, otro dispositivo Ethernet. Tiene un carácter un poco especial ya que esta conectado a un puerto paralelo.

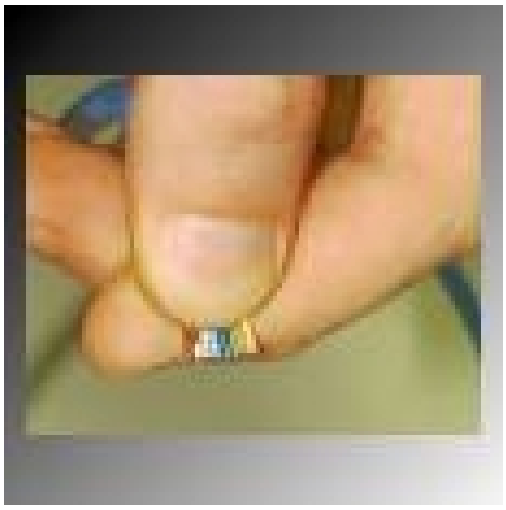
Sln Interface ***SLIP*** n -sima. Las interfaces ***SLIP*** se asocian a líneas serie en el orden en el que son instalados; por ejemplo, sl0, será la primera línea serie en ser configurada para ***SLIP***, etc. El núcleo soporta hasta cuatro interfaces ***SLIP***.

PPP*n* Interface ***PPP*** n -sima. Como ocurre con las interfaces ***SLIP***, una interfase ***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.

*Pl****IP****n* Interfase ***PLIP*** n -sima. ***PLIP*** transporta data gramas ***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 utilizaran otros nombres. Controladores como el de ***IP***X (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) desarrollo 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 fue programado por Bjorn Ekwall (bjorn@blox.se), mientras que el controlador DEPCA lo programo David C. Davies (davies@wanton.lkg.dec.com).

3.4.1 Cableado de Ethernet

Si usted esta 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 esta 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 esta disponible en los Ethernet HOWTOs, publicada mensualmente en comp.os.linux.announce por Paúl Gortmaker.

Incluimos una lista de las tarjetas mas 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 DE20-T. También hay un kit de ampliación para la DE-650-T, denominado tarjeta **PCM-CIA**.

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

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 solo con los controladores que se necesiten en ese momento.

3.4.3 Auto verificación de red Ethernet

En tiempo de arranque, el código para Ethernet intentara 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 auto verificación tiene dos limitaciones. Una de ellas es que no reconoce todas las tarjetas correctamente. Esto ocurre frecuentemente con algunas de las tarjetas clónicas mas baratas, pero también con algunas tarjetas WD80x3. La otra limitación es que el

núcleo no verificara mas de una tarjeta por defecto. Esto se hace así porque se asume que se desea tener control sobre que tarjeta es asignada a que interfase.

Si esta utilizando mas de una tarjeta, o si el proceso de auto verificación no consigue detectar su tarjeta, debe indicar explícitamente 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 auto verificación. 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. Solo es recomendable en el caso de que el código 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 intentara averiguar su valor mediante auto verificación, o utilizando un valor por defecto.

El primer parámetro indica el ***IRQ*** asignado al dispositivo. Por defecto, el núcleo intentara auto detectar 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 probara 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 auto detecte 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 detectara. 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 auto verificación, 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 maquinas. Utiliza un puerto paralelo y un cable especial, alcanzando velocidades entre 10kBps a 20 kBps.

PLIP fue desarrollado en princ***IP***io por Crynwr, Inc. Su diseño es mas 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 numero 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 interfase completa de 8 bits.

Actualmente Linux solo 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. Para conectar dos maquinas 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 numero casi incontable de personas.

Actualmente esta mantenido por Nibel Yutaka. Si se compila en el núcleo, configura una interfase de red para cada posible puerto de impresora, con plip0 correspondiendo al puerto paralelo lp0, plip1 correspondiendo a lp1, etc. El mapeado de la interfase a los puertos es ahora mismo el siguiente:

Interfaz	Puerto E/S	***IRQ***
pl***IP***0	*0x3BC*	7
pl***IP***1	*0x378*	7
pl***IP***2	*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, solo son accedidos por el controlador de ***PLIP*** cuando la interfase correspondiente ha sido configurado como *activo*.

3.6 Los controladores *SLIP* y *PPP*

SLIP (Serial Line ***IP***, Protocolo Internet en Linea 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 maquinas conectadas a Internet: esto proporciona conectividad ***IP*** a los particulares (algo que de otra forma seria 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 especifica la configuración del puerto serie para interconexión ***TCP/IP***, se le dedicara un capitulo aparte. Para mas información consultar el capitulo 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 publicas.

Este capitulo 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 como configurar el módem para marcar. Todos esos temas están contemplados en el "Serial HOWTO"1 de Greg Hankins, 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 ***Unix*** es kermit. Sin embargo resulta algo duro de usar. Hay programas disponibles mas 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 numero de programas para instalar BBS bajo Linux disponiblespara aquellos que quieran ofrecer dicho servicio. Varios de esos paquetes se encuentran en sunsite.unc.edu, en el directorio /pub/Linux/system/Network.

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 mas 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 transmitiría.

El compendio de esta clase de software de comunicaciones es ***UUCP***. Este es un conjunto de programas ficheros de una maquina 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 capitulo 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, esta 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 publico y poder ejecutar sesiones FTP, etc. Sin embargo, ***SLIP*** también puede ser usado en conexiones permanentes o sem***IP***ermanentes para uniones de ***LAN*** a ***LAN***, aunque esto último solo 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 ***UNIX*** para el acceso a dispositivos serie son llamados normalmente ttys. Esta es una abreviatura de TeletypeTM, quienes eran unos de los mayores productores de terminales en los primeros días de ***Unix***. El termino se usa actualmente para cualquier terminal de texto. En este capitulo, 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 este un terminal conectado por cable o un ordenador remoto a través de la línea telefónica.

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

Entre los parámetros mas 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 esta implementado como si fuera una línea dedicada.

Existe algo de ambigüedad sobre como medir la velocidad de la línea. El termino correcto es bit rate, el cual esta relacionado con la velocidad de transferencia de la línea medida en Bits por segundo (Pbs. 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 "bitrate", indica el estado actual de una conexión serie existente entre dos puntos, a saber, el numero medio de bits transferidos por segundo. Es importante saber que estos dos valores suelen ser diferentes, ya que la mayoría de los dispositivos codifican mas de un bit por cada impulso eléctrico.

4.3 Acceso a los Dispositivos Serie

Como ocurre con todos los dispositivos de un sistema ***Unix***, 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 comportara de forma diferente.

El primer tipo se utiliza para las llamadas entrantes y tiene un numero principal de dispositivo 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 numero principal de dispositivo igual a 5.

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

Asumamos que su módem esta en el COM2. En este caso su numero secundario será 65, Y su numero princ***IP***al será 5 para realizar llamadas. Debería existir por ello, un dispositivo 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
```

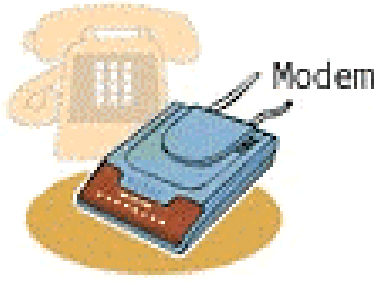
Si no existiesen tales dispositivos, entonces tendría que crearlos. Para ello, conviértase en super usuario y teclee comandos como el siguiente:

```
# 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 donde tenga un 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 esta 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 simultaneo 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. Este 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 cuando esta lista para recibir mas 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.

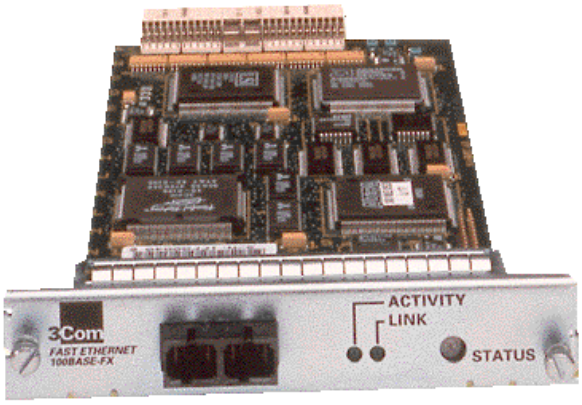
En ordenadores PC, la interfaz RS-232 es controlada generalmente por un ch***IP*** UARTdescendiente del ch***IP*** 16450 de National Semiconductor, o bien de una nueva versión deeste: el NSC16550A. Algunas marcas (princ***IP***almente los módems internos equ***IP***ados con un ch***IP*** Rockwell) también usan ch***IP***s completamente diferentes que han sido programados para comportarse como si fueran un 16550.

La princ***IP***al 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 valido para velocidades máximas de 9600 baudios, mientras que para velocidades superiores se requiere un ch***IP*** compatible con el 16550. Además de estos ch***IP***s, Linux también soporta el ch***IP*** 8250, que era el ch***IP*** 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 asignara los números 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 - guión de configuración de la línea serie
#
# Detección 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 configuración de dispositivos serie
/sbin/setserial -bg /dev/cua*
```



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. Esta comprobado que los módems internos equ***IP***ados con los ch***IP***s de Rockwell experimentan este tipo de problemas. Así, por ejemplo, si se obtiene que el ch***IP*** 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 pagina del manual de setserial(8) para mas 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
```

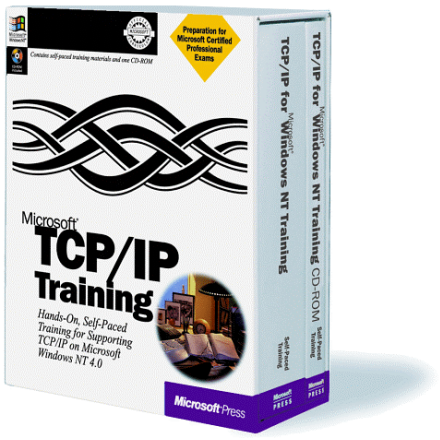
Para comprobar si el control de flujo por hardware esta activo use:

```
$ 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 capitulo recorreremos todos los pasos necesarios para configurar el protocolo ***TCP/IP*** en su maquina. 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 capitulo, generalmente, solo habrá de ejecutarlas una única vez. Una vez hecho esto, solo 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 scr***IP***ts /etc/rc.

Generalmente, las partes especificas 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 esta la estructura presente en el sistema.

Mas abajo describiré las acciones llevadas a cabo por rc.inet1, mientras que las aplicaciones son cubiertas por los capítulos posteriores. Al finalizar este capitulo, 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 maquina. Los scr***IP***ts 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 interfase 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. Sino 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 asegurandose de configurarlo incluyendo el soporte para *procfs*.

5.2 Instalación de los Ejecutables

Si esta 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 numero 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 ultimo parche y los ejecutables de Net2e 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 mas a las tareas que realmente tendrá que realizar. La Cervecera Virtual, es una pequeña compañía que produce, como su nombre indica, cerveza virtual. Para gestionar su negocio mas 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 estas se encuentren operacionales. Como un primer paso, quieren establecer una pasarela que pase los datagramas de una subred a otra. Para mas tarde, tienen planeado establecer un enlace ***UUCP*** con el exterior, a través del cual intercambiaran noticias y correo electrónico. A largo plazo, quieren establecer una conexión ocasional usando ***SLIP*** con la Internet.

5.4 Establecimiento del Nombre de la Maquina

La mayoría de las aplicaciones de red, si no todas, asumen que el nombre dado a la maquina 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
```

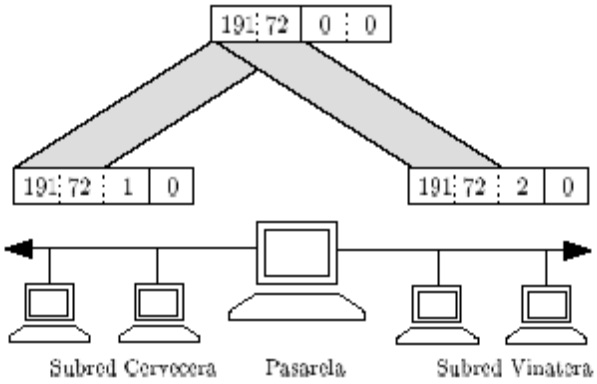
Es una practica común usar el nombre sin cualificarlo con el dominio de red. Así pues, supongamos que las maquinas de la Cervecera Virtual se llamaran vale.vbrew.com,vlager.vbrew.com, etc. Estos son los nombres oficiales, los nombres completamente cualificados de dominio (FQDN). Los nombres locales serian 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 mas abajo).

Algunas personas sugieren la utilización del comando domain name para fijar el valor del dominio para el núcleo. Así, para obtener el FQDN combinaríamos la salida de hostname y domain name. Sin embargo esto es, en el mejor de los casos, una verdad a medias. Domain name se usa por lo general para establecer el domino NIS al que pertenece la maquina que puede ser completamente diferente al del servidor de nombres (***DNS***). Hablaremos de NIS en el capitulo 10.

5.5 Asignación de una dirección *IP*

Si configura su software de red para operar su maquina de forma aislada (por ejemplo con el objeto de utilizar el software de noticias de red ***INN***) puede saltarse esta sección pues solo necesita la dirección de la interfase de lazo.

Las cosas son algo mas 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 esta estableciendo la red, tendrá que ser usted quien asigne las direcciones ***IP*** según se describe a continuación.



Cervecera Virtual y Vinatera Virtual - las dos subredes.

Las maquinas 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 (valida). Solo 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 este disponible), debe dividir su red en subredes. Es importante notar que esto es únicamente necesario si tiene mas de una dirección de "difusión"(broadcast) en la red; las conexiones punto a punto no cuentan. Así, por ejemplo, si tiene una red Ethernet y uno o mas enlaces ***SLIP*** con el exterior no hace falta que divida su red. La razón se explica en el capitulo 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 numero 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 maquinas 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 mascara de red será 255.255.255.0.

A vlager, que actúa de pasarela entre las redes, se le asigna el numero de maquina 1 en ambas redes, lo que significa que tiene las direcciones ***IP***, 191.72.1.1 y 191.72.2.1, respectivamente. La figura superior muestra las dos subredes y la maquina 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 seria mas realista. Con el nuevo código de red, la división en subredes no esta 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 maquinas cada una.

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 maquinas en el fichero *hosts*.

Incluso si planea utilizar los servicios ***DNS*** y ***NIS*** en condiciones normales de operación, es conveniente tener un reducido numero de maquinas 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 solo de una cuestión de conveniencia, sino que permite el uso de nombres simbólicos para las maquinas citadas en las macros *rc.inet*. De esta forma, para cambiar las direcciones ***IP***, solo tiene que copiar el fichero *hosts* modificado a todas las maquinas y rearrancar, en vez de tener que modificar un gran numero de macros *rc* por separado. Generalmente, también debe incluir los nombres y direcciones locales en *hosts*, añadiendo los de las maquinas que enlacen varias redes y los servidores ***NIS*** si existen.

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 capitulo 6.

El fichero hosts contiene un registro por línea, consistente en una dirección ***IP***, un nombre de maquina y de forma opcional, una lista de alias para esa maquina. 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 maquinas pueden ser con calificación completa, o relativos al dominio actual. Para la maquina vale, el registro generalmente incluiría el nombre con cualificación completa, vale.vbrew.com, y vale en el fichero hosts, de forma que pueda ser referido usando el nombre oficial y el nombre local que es mas corto.

Este 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 maquina 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, podrimos instalar un fichero networks como este:

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

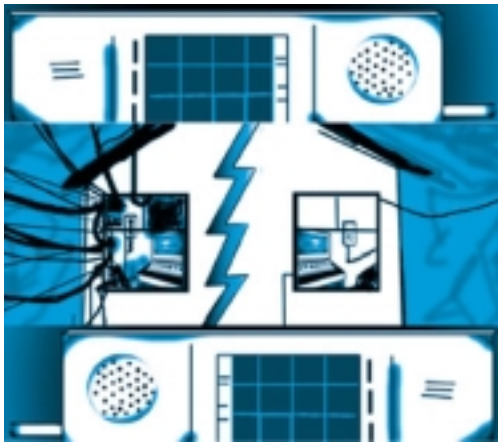
5.7 Configuración de la Interfase para *IP*

Una vez ha configurado su hardware según se ha explicado en el capitulo 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 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 interfase. Esto incluye la asignación de una dirección ***IP*** y otros parámetros, así como la activación de la interfase. Por activación nos referimos a permitir que el núcleo envía y recibe datagramas ***IP*** a través de la interfase. El modo mas sencillo de invocar esta herramienta es

ifconfig interfase dirección-***IP***

que asigna **dirección-*IP*** a **interfase** y la activa. Los otros parámetros toman valores asignados por defecto. Por ejemplo, la mascara 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 capitulo.



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 Interfase de Bucle o Loopback

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

```
# ifconfig lo 127.0.0.1
```

Ocasionalmente, también vera que el nombre comodín localhost es usado en vez de la dirección de ***IP***. ifconfig buscara el nombre en el fichero hosts que debe contener un registro declarando localhost como nombre valido 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 interfase, basta ejecutar el programa ifconfig usando el nombre de la interfase 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 mascara asignada a la interfase de lazo es 255.0.0.0, debido a que 127.0.0.1 es una dirección de clase A. La interfase no tiene establecida ninguna dirección de difusión, ya que esta no suele ser demasiado util para lazos. Sin embargo, si va a ejecutar el demonio rwhod en su maquina, tendrá seguramente que fijar la dirección de difusión del dispositivo de lazo para que rwho funcione correctamente. El modo de fijar dicha dirección se explica en la sección "5.8", mas abajo.

Ahora, ya casi puede empezar a jugar con su "mini-red". Solo resta añadir una entrada en la tabla de encaminamiento que comunique al ***IP*** que puede usar esa interfase 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 red5 y se usa para verificar que una dirección dada es accesible y para medir el retraso entre el envió 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 continua a menos que sea interrumpida por el usuario. El ^C marca el momento en el que se apretó Ctrl-C. Este ejemplo muestra que los paquetes dirigidos a la maquina 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 interfase de red.

Si la salida de ping no se parece a la de mas 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 este 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 ejecuto el comando route incorrectamente. Asegúrese de que es la misma dirección que la que uso con ifconfig.

Los pasos descritos arriba son suficientes para poder ejecutar aplicaciones de red en una maquina aislada. Una vez esas líneas 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 maquina y probar las diferentes aplicaciones de red. Por ejemplo "telnet localhost" debería establecer una conexión telnet con su maquina, pidiéndole el nombre de usuario y la contraseña.
Sin embargo, la interfase de lazo es útil, no solo como ejemplo en libros de redes, o como método de pruebas durante el desarrollo: también la utilizan algunas aplicaciones como modo normal de operación. Por ello, debe usted configurarla siempre, independientemente de que su maquina este conectada a una red o no.

5.7.2 Interfaces Ethernet

La configuración de una interfase Ethernet es mas o menos igual que la de la interfase de lazo. Solo requiere algunos parámetros mas cuando esta usando varias subredes. En la Cervecera Virtual, hemos dividido la red ***IP***, originalmente de clase B, en subredes de clase C. Para que la interfase reconozca esto, el comando usando *ifconfig* sería:

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

Esto asigna a la interfase ***eth0*** la dirección ***IP*** de la maquina vstout(191.72.1.2). Si hubiésemos omitido la mascara de red, ifconfig habría deducido la mascara 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
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 maquina activados. Además se fija la unidad de transferencia de mensajes (tamaño máximo que el núcleo va a generar para esa interfase) a un máximo de 1500 bytes. Todos estos valores pueden ser especificados mediante opciones especiales que se explican mas tarde.

De forma semejante al caso de la interfase 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 esta claro como route detecta cual es la interfase 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 interfase y la mascara de red). La única interfase que cumple esto es ***eth0***.

Veamos, ¿que significa la opción -net? Esta opción es necesaria porque el programa route es capaz de trabajar con rutas a redes o a maquinas 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 maquina fijándose en los bits de maquina 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 maquina. Por tanto, route supondría que 191.72.1.0 es la dirección de una maquina 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 explicita utilizando el indicador -net.

Por supuesto, escribir el comando route es tedioso y susceptible de muchos errores de escritura. Un método mas conveniente es usar los nombres definidos en /etc/networks como vimos mas arriba. Esto hace el comando mas 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 interfase Ethernet esta funcionando correctamente. Elija una maquina 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    ets transmitted, 4 packets received, 0% packet loss
round-trIP (ms)  min/avg/max = 3/8/12
```

Si el resultado no es similar a este, algo va mal, obviamente. Una tasa de perdida de paquetes inusualmente alta, sugiere un problema de hardware, como terminaciones en mal estado o incluso la ausencia de las mismas, etc. Si no recibe ningún paquete, debe comprobar la configuración de la interfase mediante netstat. Las estadísticas de paquetes producidas por ifconfig le indican si algún paquete ha sido enviado mediante esa interfase. Si tiene acceso a una maquina remota, también debería dirigirse a esa maquina y comprobar las estadísticas de la interfase. De este modo puede determinar exactamente en que momento se han descartado los paquetes. Además, debe consultar la información de encaminamiento con route para ver si ambas maquinas han registrado esta 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 maquinas):

```
# 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 mas adelante en la sección "Comprobación mediante netstat". La columna Flags contiene una lista de los indicadores activos en cada interfase. U indica que la interfase esta activa y H indica que la dirección de destino es una maquina. Si encuentra que el indicador H se ha activado para una ruta que pretendía usar para una red, entonces debe usar la opción -net con el comando route. Para comprobar si alguna ruta esta siendo usada o no, debe mirar si el campo Use en la penúltima columna se incrementa entre dos ejecuciones sucesivas de ping.

5.7.3 Encaminamiento a través de una Pasarela

En la sección anterior, cubrí solo el caso en el que la maquina solo tiene una única Ethernet. Frecuentemente, es posible encontrar redes conectadas unas a otras a través de pasarelas o maquinas de enlace. Estas pasarelas pueden simplemente unir dos o mas Ethernets, pero pueden también 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 maquina vlager ha sido configurada ya, solo tenemos que añadir otro registro a la tabla de encaminamiento de la maquina vstout que le comunique al núcleo que puede acceder a todos las maquinas 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 solo podría enviar datos de vstout a vbardolino, pero la respuesta del segundo iría a parar al cubo de la basura.

Este ejemplo describe unicamente 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 esta contenida en el código de route.

Una tasa alta de perdida de paquetes usando ping hacia una maquina situada detrás de una o mas pasarelas, puede deberse a que la red esta muy congestionada. La perdida de paquetes no se debe tanto a deficiencias técnicas como a exceso temporal de carga en las maquinas 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, respectiva- mente conectadas a cada una de las dos redes. Todo lo que necesitara 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 Interfase PL*IP*

Si usa un enlace ***PLIP*** para conectar dos maquinas, 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 solo requiere dos maquinas ("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 PL***IP***. El portatil se llama vlite, y tiene un único puerto paralelo. Durante el arranque, este puerto será registrado como pl***IP***1. Para activar el enlace, ha de configurar la interfase pl***IP***1 mediante los siguientes comandos:

```
# ifconfig plIP1 vlite pointopoint vlager
# route add default gw vlager
```

El primer comando configura la interfase, 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 interfase plip1 en vlager no necesita tener una dirección ***IP*** diferente, sino que puede usar la misma dirección 191.72.1.1. Una vez hemos configurado el encaminamiento desde el portátil a la red de la Cervecera, solo resta arbitrar un modo para que cualquier maquina en esa red pueda acceder a vlite. Un modo particularmente enrevesado seria añadir una ruta a las tablas de encaminamiento de cada una de las maquinas de la red para usar vlager como pasarela hacia vlite:

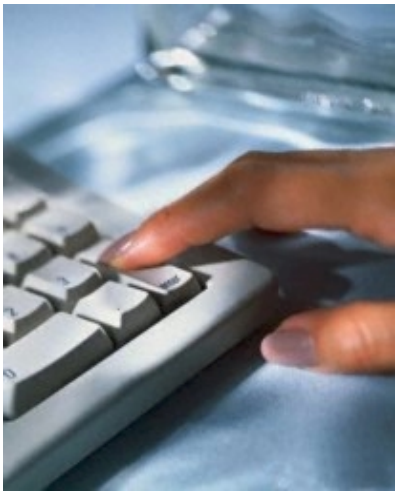
```
# route add vlite gw vlager
```

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 maquinas de la red de modo que distribuya la información de encaminamiento de forma dinámica. La forma mas 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***", mas ade***LAN***te.

Las versiones futuras de Net-3 contendrán una herramienta llamada plipconfig capaz de fijar el numero de ***IRQ*** del puerto de la impresora. Mas tarde se sustituirá por un comando ifconfig mas 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 mas 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 capitulos 7 y 8.

5.7.7 La Interfase Comodín

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

El dilema con las maquinas 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 resultara necesario enviar datos a la dirección ***IP*** "oficial" de la maquina. Supongamos, por ejemplo, el caso del portátil vlite cuando no esta 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 intentara enviar los datos a esa dirección. Como la única interfase activa en ese momento es la de lazo, el núcleo no sabe que la dirección se refiere a la misma maquina. En consecuencia el núcleo descarta el datagrama y genera un error en la aplicación.

En esta situación es cuando la interfase comodín es útil, resolviendo el dilema actuando como alter ego de la interfase 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 mas parámetros que los descritos hasta ahora. General mente se ejecuta en la forma:

ifconfig interface [[-netl-host] direccion [parametros]]

interface es el nombre de la interfase y dirección es la dirección ***IP*** que se asigna a dicha interfase. La dirección puede estar en forma de cuaterna o usando un nombre que ifconfig buscara 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 maquina respectivamente.

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

```
# 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 interfase 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 numero de errores ocurridos, de cuantos paquetes han sido descartados, seguramente por memoria insuficiente, y cuantos 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 mas o menos con los parámetros con los que se puede ejecutar; estos parámetros son explicados mas 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 interfase 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 interfase que se ha desactivado temporalmente mediante la opción “down”.
Esta opción corresponde a los indicadores UP RUNNING.

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

netmask mascara

Esto asigna una mascara de subred a una interfase. 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 maquinas. Esta opción es necesaria para, por ejemplo, configurar las interfaces ***SLIP*** o PL***IP***.

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 maquina. Algunas implementaciones de los protocolos ***IP*** utilizan un esquema diferente; esta opción proporciona un método para adaptarse a esos entornos mas raros.
(***ifconfig*** confirma el establecimiento de una dirección de difusión incluyendo el indicador BROADCAST.)

metric numero

Esta opción puede ser usada para asignar un valor de métrica a la tabla de encaminamiento creada para la interfase. Esta métrica es usada por el Protocolo de Información de Encaminamiento (R***IP***, 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 esta ejecutando un demonio R***IP***, no necesita usar esta opción para nada; si por el contrario si lo usa, al menos solo tendrá que modificar este valor en contadas ocasiones.

mtu bytes Esto fija la unidad máxima de transferencia, o lo que es lo mismo, el maximo numero de octetos que la interfase 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 especifica 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 maquinas conectadas a la red. Para redes de difusión, esta opción es habilitada por defecto.
ifconfig avisa que ***ARP*** ha sido inhabilitado mediante el indicador NO***ARP***.

- ***ARP*** Inhabilita el uso de ***ARP*** para esta interfase.

promisc Pone la interfase en modo promiscuo. En una red de difusión, esto hace que la interfase reciba todos los paquetes, independientemente de si eran para ella o no. Esto permite el análisis del trafico de red utilizando utilidades como filtros de paquetes, también llamado fisgar. Se trata de una buena técnica para localizar problemas de red que de otra forma resultan difíciles.
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.

Esta opción corresponde al indicador PROMISC.

- promisc Desactiva el modo promiscuo.

allmulti Las direcciones de envió múltiple son un tipo de difusión pero a un grupo de maquinas que no tienen necesariamente que pertenecer a la misma subred. El núcleo no soporta todavía direcciones de envió múlt***IP***le o de multidifusion11.

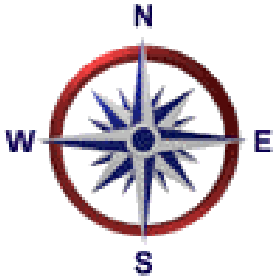
Esta opción corresponde al indicador ALLMULTI.

allmulti Desactiva las direcciones de envió múlt***IP***le.

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 maquinas o las redes. Esto es especialmente útil si pretende evitar consultas para esos nombres a través de la red (por ejemplo consultas a un servidor ***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 mascara 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 interfase esta activa.

H — Esta interfase permite el acceso a una sola maquina. Este es el caso de la interfase 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 numero de referencias a esta ruta, esto es, cuantas otras rutas dependen de esta (por ejemplo a través de pasarelas). Las dos ultimas columnas muestran el numero de veces que cada ruta ha sido usada y la interfase que procesa los datagramas para dicha ruta.

5.9.2 Consulta de las Estadísticas de una Interfase

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 solo aquellas que ya han sido configuradas. En vstout, netstat producira:

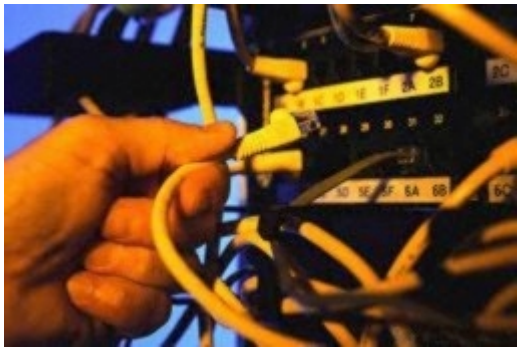
```
$ netstat -i
Kernel Interface table
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flags
lo      0   0  3185      0      0      0   3185      0      0      0
BLRU
eth0   1500 0  972633    17     20     120 628711    217    0      0
BRU
```

Los campos MTU y Met informan de los valores de MTU y métrica configurados en la interfase. Las columnas RX y TX muestran el total de paquetes recibidos y enviados respectivamente sin errores (RX-OK/TX-OK), da"nados (RX-ERR/TX-ERR), cuantos han sido descartados (RX-DRP/TX-DRP), y cuantos se han perdido por desbordamiento (RX-OVR/TX-OVR).

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

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

5.9.3 Mostrar Conexiones



netstat soporta una serie de opciones que permiten mostrar los sockets 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)
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.

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 maquina especificada o, en el caso de que no se especifique ninguna, se muestran todas. Así, si ejecutáramos ***ARP*** en vlager obtendríamos:

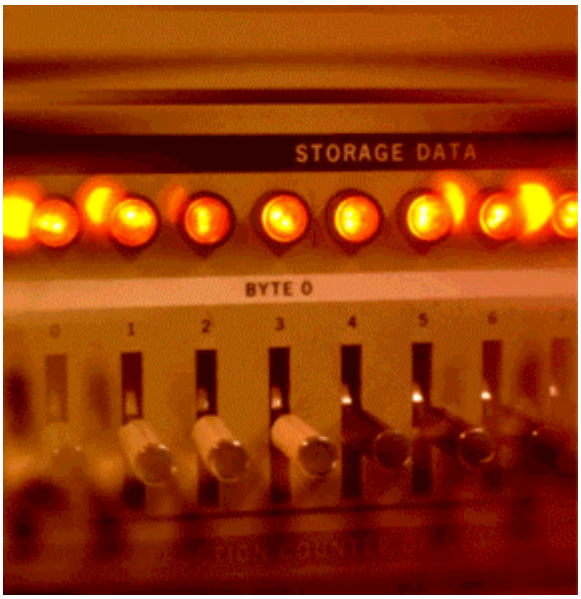
```
# ARP -a
IP address      HW type              HW address
191.72.1.3      10Mbps Ethernet      00:00:C0:5ª:42:C1
191.72.1.2      10Mbps Ethernet      00:00:C0:90:B3:42
191.72.2.4      10Mbps Ethernet      00:00:C0:04:69:AA
```

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

Se puede usar la opción -t para mostrar la información referente a un tipo especifico de hardware. Los valores posibles son ether, ax25, o pronet, y se refieren a Ethernet a 10Mbps, AMPR AX.25, y equ***IP***os 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 maquina remota falla, por ejemplo debido a que su controlador ***ARP*** no funciona correctamente o cuando alguna otra maquina en la red se identifica erróneamente como si ella misma tuviera esa dirección ***IP***. También es un modo, aunque algo drástico, de protegerse frente a maquinas 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 maquina dada. De este modo se puede forzar a una interfase 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 maquina, digamos gate, actúa como pasarela para otra diferente llamada fnord, haciendo como que ambas direcciones pertenecen a la misma maquina, en este caso gate. Esto se consigue haciendo publico un registro ***ARP*** para fnord que apunta a su propia interfase Ethernet. De este modo, cuando cualquier maquina de la red realiza una consulta sobre fnord, gate responde con un registro que contiene su propia dirección Ethernet. La maquina que ha realizado la consulta enviara 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 maquina DOS que tiene una implementación de ***TCP*** incorrecta que no entiende el encaminamiento demasiado bien. Cuando usa ***ARP*** sustituto, a todos los efectos la maquina DOS ve a fnord en la subred local y, por tanto, no necesita preocuparse de como realizar el encaminamiento a través de una pasarela.

Otro aplicación muy útil del ***ARP*** sustituto es cuando una de sus maquinas actúa como pasarela para otra maquina aunque solo 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 solo funcionara si la dirección de la maquina 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 maquina de la subred de la Cervecera (191.72.1.0), pero nunca para maquinas de la subred de la Vinatera (191.72.2.0).

Abajo vemos el comando correcto para activar un ***ARP*** sustituto para fnord; por su puesto, 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 bastara:

```
# 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 especifica de una ruta invalida el valor especificado para la interfase. El uso típico de esta opción es para rutas a través de pasarelas, en las que el enlace entre la pasarela y la maquina destinataria requiere un MTU muy bajo. Por ejemplo, supongamos que la maquina wanderer este 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 mas peque"nos que quepan en 296 bytes. Si en vez de eso, configura la interfase en vstout para que use una MTU de 296 desde el princ***IP***io, se puede evitar el proceso de división, que es relativamente costoso.:

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

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

Capítulo 6 Servicio de Nombres. Configuración

Como se comento en el capitulo 2, la red ***TCP/IP*** puede utilizar diferentes métodos para convertir nombres en direcciones ***IP***. El mecanismo mas simple consiste en almacenar los nombres en una tabla de maquinas en el fichero /etc/hosts. Esto es unicamente interesante en el caso de pequeñas redes de área local que solo requieran la administración de una persona, y que no tengan trafico ***IP*** con el mundo exterior. Recordamos que el formato del fichero hosts fue descrito en el capitulo 5.

Alternativamente, puede utilizarse BIND - el servicio de nombres Internet de Berkeley o "Berkeley Internet Name Domain" - para traducir nombres de maquinas 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 mas 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 maquina: generalmente, uno para toda la red es suficiente.

Este capitulo le dará ideas generales acerca de como configurar y ejecutar un servidor de nombres. Si pretende usar ***BIND*** en un entorno mas 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 interesara echar un vistazo a los comentarios adicionales que aparecen junto a las fuentes de su versión de ***BIND***. Tambien 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 princ***IP***ales son gethostbyname(2) y gethostbyaddr(2), que buscan la dirección ***IP*** de una maquina 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 que servicios debe usar y en que orden.

Las opciones del fichero host.conf deben estar en líneas distintas. Los campos deben separarse por b***LAN***cos (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 resolucion se usan. Opciones validas 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 que servicio se prueba en primer lugar para intentar resolver el nombre.

Multi Va con las opciones on u off. Determina si una maquina del fichero /etc/hosts puede tener distintas direcciones ***IP*** o no. Esta opción no tiene efecto en peticiones vía ***NIS*** o ***DNS***.

Nospoof Como se explico en el capitulo anterior, ***DNS*** le permite encontrar un nombre de maquina perteneciente a una dirección ***IP*** dada utilizando el dominio in-addr.***ARP***a. Los intentos de los servidores de nombres de proporcionar un nombre falso se conocen en Ingles 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 retornara un error. Esta opción se activa poniendo nospoof on.

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 quitara a los nombres antes de buscar su dirección. Es util para las entradas del fichero *hosts*, que podrán así ir solos los nombres de maquinas, sin el dominio. Cuando se busque una maquina con el nombre de dominio localeste 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 maquina podría ser local a diversos dominios.

Un ejemplo de este fichero para la maquina **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 mas prioridad sobre las puestas en el fichero host.conf. Estas 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 realizaran comprobaciones contra los nombres falsos, pero en el primer caso se mandaran 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 que servidores utilizar. El fichero resolv.conf contiene una lista de servidores, que si esta vacía hará considerar al sistema que el servidor esta en su máquina.

Si ejecuta un servidor de nombres en su maquina local, tendrá que configurarlo por separado, como se explicara 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 intentaran 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 mas: domain y search, indicando la primera dominios alternativos a probar si la búsqueda inicial del nombre falla. Estos dominios irán separados por b***LAN***cos o tabuladores.

Si no se incluye una opción search, se construirá una lista de búsqueda por defecto por el dominio local mas 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 getdoma***INN***ame(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 princ***IP***al va a ser vlager:
***nameserver***      191.72.1.1
```

Cuando se trate de traducir el nombre vale, el sistema empezara por buscar directamente vale y si falla, probara 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 princ***IP***ales 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 este.

Una solución es poner un servidor de nombres local que se ocupe de sus nombres locales, y reenvié todas las peticiones de otros nombres a los servidores princ***IP***ales. Por supuesto, esto solo 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 princ***IP***al de nombres falla.

6.2 Ejecución de *named*

El programa que proporciona servicio de nombres en las máquinas ***Unix*** suele ser named . Es un servidor desarrollado inicialmente para ***Unix*** tipo BSD, con el propósito de proporcionar servicio de nombres a maquinas 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, esta en este momento en versión Beta, y pronto estará disponible para Linux.

Esta sección requiere ideas acerca de como funciona el Sistema de Nombres y Dominios (***DNS***). Si lo que sigue a continuación le suena a chino, puede releer el capitulo 2, que le dará información acerca de como funciona básicamente el ***DNS***.

El programa named suele iniciarse al arrancar la maquina, 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 iniciara y leerá el fichero named.boot y los ficheros de zona que se especifiquen en el. Su numero de proceso será anotado en ASCII en el

fichero /var/run/named.pid, recibirá ficheros de zona de los servidores princ***IP***ales si es necesario y comenzara a escuchar las peticiones de ***DNS*** por el puerto 53.

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 mas detalle el formato de este fichero, observaremos el ejemplo para la maquina vlager dado en la figura 6.1.

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 princ***IP***al para tres dominios: los que se indican con el comando primary. La primera línea dice que named actué como servidor princ***IP***al 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.

```
;
; 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 maquinas 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 mas tarde.

A continuación se presenta una lista de las opciones mas 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 ira 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 intentara 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. Solo 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 este 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 realizara consultas recursivas, sino que las redirigirá a los servidores especificados con *forwarders*.

Hay dos opciones adicionales que no vamos a describir: sortlist y domain. Ademas, 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 maquinas 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 si 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 maquina con su nombre oficial. Como ejemplo, obsérvese la figura 6.3 de la pagina 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 ultimo 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*. El campo TTL especifica, en segundos, el tiempo de validez de la información desde que se obtiene del servidor. Es un numero decimal de hasta ocho dígitos.
Si no se especifica ningún valor, tomara 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 describiran mas adelante.

A continuación se presenta una lista incompleta de **RRs** que se utilizan en los ficheros de **DNS**. Hay algunos mas que no vamos a comentar. Son experimentales, y de escaso uso.

SOA Describe una zona de autoridad (SOA significa "Start of Authority", es decir, "Comienzo de Autoridad"). Señala que los registros siguientes contienen información "autorizada" para el dominio. Cada fichero incluido en la opción *primary* debe tener un registro **SOA** para esta zona. Los datos asociados contienen los siguientes campos:

Origin Nombre canónico del servidor de nombres primario para este dominio. Se suele dar como nombre absoluto.

Contact Dirección de correo electrónico de la personaresponsable 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 numero de versión del fichero de zona, expresado con un numero decimal. Cuando se cambien datos del fichero, deberá incrementarse este numero. El numero de versión es utilizado por los servidores secundarios para saber cuando 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 numero 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 numero decimal de hasta ocho dígitos.
Normalmente, la topología de la red no cambia mucho, con lo que este numero será como poco de un día para grandes redes, y de mucho mas tiempo para redes pequeñas.

retry Este número determina los intervalos de tiempo entre reintentos de comunicación con servidores primarios cuando petición de una zona falla. No debe ser peque"no 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 tardara 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 maquina solo 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 que 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.*ARP*a** con sus nombres normales. Se usa para obtener nombres a partir de direcciones ***IP*** (traducción inversa). El nombre de la maquina debe ser el canónico.

MX Especifica el *servidor de correo para* un dominio. En la sección "Encaminado de correo en la Internet" del capitulo 13 se explica por que son necesarios estos servidores. La sintaxis del registro **MX** es:

[domain] [***TTL***] [class] MX preference host

host es el nombre del servidor de correo para el domain. Cada servidor tiene un valor entero de preferencia (**preference**) asociado. Una gente de transporte de correo que desee entregar mensajes al dominio indicado en domain lo intentara 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 maquina. 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 b***LAN***cos, 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

El fichero *named.ca* mostrado en la figura 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 mas Ade***LAN***te

```
;
  ; /var/named/named.ca          Fichero de cache.
  ;              No estamos en Internet, luego no necesitamos
  ;              servidores raíz. 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
```

Fichero named.ca.

```
  ;
  ; /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 día
                         3600      ; reintentos: una hora
                         3600000   ; expiración:  42 días
                         604800    ; mínimo: 1 semana )
                 IN  NS    vlager.vbrew.com.
  ;
  ; el correo local se distribuye en vlager
                 IN  MX    10 vlager
  ;
  ; dirección 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 también 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
```

Fichero *named.hosts.*

```
;
; /var/named/named.local      Traducción 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    ; expiración:  42 días
                          360000     ; mínimo: 100 horas )
                IN  NS    vlager.vbrew.com.
1               IN  PTR   localhost.
```

Fichero named.local.

```
  ;
; /var/named/named.rev        Traducción inversa de nuestros números IP
;                             El origen es 72.191.in-addr.ARPa.
;
@               IN  SOA   vlager.vbrew.com. (
                          joe.vbrew.com.
                          16         ; serial
                          86400      ; refresco: una vez al día
                          3600       ; reintento: una hora
                          3600000    ; expiración:  42 días
                          604800     ; mínimo: 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.
```

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 ultimo caso, se invoca simplemente como:

nslookup nombre

y pedirá el nombre indicado al servidor de nombres que aparezca en *resolv.conf* (si aparece mas 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 mostrara el servidor de nombres en uso y entrara en modo interactivo. En el prompt '>' que se mostrara, puede teclear cualquier nombre de dominio por el que quiera preguntar. Por defecto, preguntara 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 mas 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 practica 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 como 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 ayudara 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 mi) es el programa ***DNS***walk, 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 ***DNS***walk 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. Solo se necesita un módem y un puerto serie con buffer **FIFO**. Utilizarlo no es mas complicado que usar un buzón, y cada vez existen mas 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 mas veterano y por tanto mas estable. ***PPP*** para Linux ha sido recientemente desarrollado por Michael Callahan y Al Longyear; y se describirá en el próximo capitulo.

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 mas fácil es poner la dirección de algún servidor de nombres en el fichero *resolv.conf* ; este servidor se usara tan pronto como ***SLIP*** conecte. Lo mejor es poner el servidor de nombres mas cercano.
Sin embargo, esta solución no es la optima, 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 *solo con cache*. No requiere un dominio ya que solo actuara como *relevo*, es decir, pasara 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, solo se contactara con el servidor externo la primera vez. Un fichero *named.boot* que sirva, 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 raíz
```

Además debe tener un fichero db.cache con una lista de servidores raíz válidos. Este fichero esta descrito al final del capitulo 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 interprete 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 maquina.

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) esta en modo normal (**DISC0**), intercambiara datos solo 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 pasaran al manejador ***SLIP***.

El manejador de ***SLIP*** entiende por si mismo varias versiones del protocolo, incluyendo ***CSLIP***, que realiza la llamada compresión de cabeceras de Van Jacobson en los paquetes IP salientes. 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:

```
# slatt ach  /dev/cua3  &
```

Esto cambiará el modo de línea de cua3 a ***SLIPDISC***, y la enganchara a uno de los interfaces ***SLIP*** disponibles. Si es la única conexión ***SLIP*** se enganchara a la interfaz 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 c***SLIP***, ***SLIP***6, c***SLIP***6 (para la versión de 6 bits) y adaptive para ***SLIP*** adaptativo, que deja al núcleo averiguar que 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 **C*SLIP***, tendrá que usarlo Ud. también. El síntoma típico de una selección incorrecta es que la orden ping a una maquina remota no tendrá respuesta. Si la otra maquina le hace ping a Ud, 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 solo le permite activar ***SLIP***, sino también otros protocolos serie como ***PPP*** o **KISS** (protocolo que se usa en packet-radio). Para mas 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 maquina **vlager** hemos llamado al servidor **cow*SLIP***, se debería ejecutar:

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

El primer comando configura la interfase 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 maquina 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 interfase y enviar al proceso slattach la señal SIGHUP. Después de esto se deberá colgar el modem 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 solo tenga que invocar un comando. El programa *dip* hace esto. 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 interprete de un lenguaje script sencillo que puede manejar automáticamente el modem, 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 mas 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 to 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 aun 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* especifico. Estos programas pueden tener privilegios sin peligro.

7.3.1 Un script de ejemplo

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

```
# dip cowslip.dip
DIP: Dialup IP Protocol Driver versión 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
# Script de dip para conectarse al servidor cowslip

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

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

# 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.
```

Un script de ejemplo para *dip*

Después de conectar a ***cowslip*** y activar ***SLIP***, dip pasara 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 enviara una señal de colgar al proceso dip, cuyo numero se encontrara almacenado en el fichero */etc/dip.pid*

```
# dip -k
```

En el lenguaje que interpreta *dip* las palabras precedidas con un signo de dolar se corresponden con nombres de variables. *dip* tiene un conjunto predefinido de variables que se listara 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 maquinas local y remota han sido **vlager** y ***cowslip***, respectivamente.

Las cinco sentencias que siguen preparan la línea serie y el modem. La palabra *reset* envía una cadena de reinicio al modem; que será el comando ***ATZ*** para modems compatibles con Hayes. La siguiente sentencia limpia el buffer de salida del modem, para conseguir que el dialogo de entrada (login y password) funcione correctamente. Este dialogo es extremadamente simple: llama al número 41988, que es el numero 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 linea en modo ***SLIP*** y configura automáticamente el interfase y la tabla de encaminamiento.

7.3.2 Guía de Referencia de *dip*

Aunque se utiliza mucho, *dip* aun no esta 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 funcionara 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
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 como se introduciría el comando en modo prueba, y que 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 módem selecciona el tipo de módem. Actualmente solo esta soportado el tipo ***HAYES*** . Debe decirse el tipo, pues si no *dip* se negara 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 módems 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 dialogo con el módem podría fallar, porque lea la respuesta ***OK*** que proceda desordenes anteriormente enviadas al módem.

El comando *init* selecciona la cadena de inicialización enviada al módem antes de marcar, que para módems Hayes es, por defecto, la cadena "***ATE0 Q0 V1 X1*** ", que activa el eco de los comandos, hace que el módem de 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 módems Hayes es ***ATD***.

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 Ud. 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 maquina 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 asignara de un conjunto predeterminado y nos informara 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 solo 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 maquina y su dirección IP. Poniendo el nombre de la maquina, *dip* guardara dicho nombre en la variable $local, al tiempo que guardara la dirección **IP** en la variable *$locip*.

Las variables *$remote* y *$rmtip* hacen lo mismo pero con la maquina 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 delultimo 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 numero 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 dialogo. *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 caracter 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 fallara 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 mas, el tiempo se especificara en segundos.

Comandos *mode* y *default*

Se utilizan para cambiar el puerto entre modo ***SLIP*** o normal, y configurar la interfase. 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* ejecutara un comando *ifconfig* para configurar la interfase como enlace punto a punto y otro *route* para cambiar las tablas de encaminamiento apuntando a la maquina 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 maquina 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 sí se ejecuta como *diplogin*. Su configuración principal se encontrara en */etc/diphosts*, que asocia nombres de cuenta con direcciones de maquina asignadas. Alternativamente, puede usar *sli-plogin*, una utilidad procedente de ***BSD*** que proporciona un esquema de configuración que le permite ejecutar *shell scripts* cuando las maquinas se conectan y desconectan. Actualmente, su desarrollo esta 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* entrara en modo servidor. Para comprobar si esta autorizado para usar ***SLIP***, buscara 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 mas adelante). El tercero es el nombre o dirección IP de la maquina llamante. El siguiente es un campo informativo acerca del usuario, por el momento sin utilidad. Por ultimo, se describen separados por comas los parámetros de la conexión: el protocolo (***SLIP*** o C***SLIP***) seguido del valor de MTU.

Cuando **dent** entra en su cuenta*, diplogin* extrae la información acerca de el 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 **C*SLIP***, y preparara la interfaz y el encaminamiento. Esta conexión permanecerá hasta que el módem opuesto cuelgue, momento en que *diplogin* dejara la línea en modo normal y terminará.

d*iplogin* necesita privilegios de superusuario. Si no tiene puesto el bit setuid, deberá copiar el programa con el nombre *diplogin* y ponerle a este los privilegios. A *diplogin* se le pueden dar sin miedo, sin afectar al estado de *dip* en si 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 esta muy lejos de ser completa; si quiere saber mas sobre el ***PPP***, lea sus especificaciones en el **RFC 1548**, así como en la docena de **RFCs** que le acompañan.

En la parte mas baja del ***PPP*** esta el protocolo de Control de Conexión de Datos de *Alto-Nivel,* abreviadamente **HDLC**, que define los limites 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, es utilizado en la parte más alta del **HDLC** para negociar las opciones concernientes a la conexión de datos, tales como la Unidad Máxima de Recepción (**MRU**) que establece el tamaño máximo del datagrama que cada extremo de comunicación acepta recibir.

Un paso importante en la configuración del enlace ***PPP*** corresponde a la utentificación 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 si. Hay dos protocolos distintos, según el tipo de autentificación, los cuales discutiremos mas adelante. Se llaman el Protocolo de Autentificación por Contraseña, o **PAP**, y el Protocolo de Autentificación por Reto, o **CHAP.**

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 que direcciones **IP** van a utilizar. El protocolo de control utilizado para esto es el **IPCP**, 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 **C*SLIP***, 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*** esta dividida en dos partes, un controlador de DIC de bajo nivel situado en el kernel, y el demonio ***PPP****d* 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 modulo ***PPP*** para el kernel, el ***PPP****d*, y un programa llamado chat utilizado para llamar al sistema remoto.

El controlador del ***PPP*** para el kernel fue escrito por Michael Callahan. El ***PPP****d* fue escrito a partir de una implementación gratuita del ***PPP*** para maquinas 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]

Al igual que el ***SLIP***, el ***PPP*** esta 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 ***PPP****d*, 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 ***PPP****d* 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 ***PPP****d*, sino solamente darle una introducción. Para mas información, lea las paginas de manual y los ficheros ***README*** de la distribución con las fuentes del ***PPP****d*, 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 encontrara a la mayor parte de la gente envuelta en el desarrollo del ***PPP****d.*

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 mas información.

Como ejemplo introductorio de como establecer una conexión ***PPP*** con el ***PPP****d*, suponga que esta 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 utilizo para llamar, se ejecuta el siguiente comando:

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

Esto cambiara 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 ***PPP****d* 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 ***PPP****d* funcionara, así que no nos meteremos mas con este asunto. Volveremos a tratar el LCP con mas 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 ***PPP****d* negociará entonces los parámetros **IP** con su compañero usando **IPCP**, el protocolo de control **IP**. Al no especificar dirección **IP** alguna, el ***PPP****d* intentara usar la dirección que se obtiene al resolver el nombre del ordenador local. Decididas las direcciones, cada ***PPP****d* se lo comunicara al otro extremo.

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

Tras pasar por la fase de configuración **IPCP**, el ***PPP***d configurara la red de su ordenador para utilizar el enlace ***PPP***. En primer lugar, configurara el interfase de red ***PPP*** como un enlace punto-a-punto, utilizando el ***PPP****0* para el primer enlace ***PPP*** que este activo, ***PPP****1* para el segundo, y así sucesivamente. A continuación preparara una entrada de la tabla de encaminamiento que apunte al ordenador del otro extremo del enlace. En el ejemplo anterior, el ***PPP****d* hará que el encaminamiento de red por defecto apunte a **c3po**, debido a que lo especificamos con la opción *defaultroute*. 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 ***PPP****D*, y las cubriremos en mayor detalle mas adelante.

8.4 Los Ficheros de Opciones

Antes de que el ***PPP****d* 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 valido, distribuido a través de un cierto numero 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 ***PPP****d* 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 este en nuestras bases de datos para la autentificación.

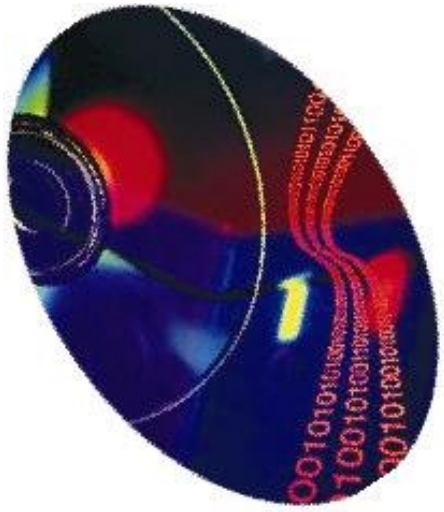
El otro fichero de opciones, que es leído después del */etc/**PPP**/options*, es el *.**PPP**rc* situado en el directorio *home* del usuario. Permite que cada usuario especifique su propio conjunto de opciones por defecto.

Un fichero */etc/**PPP**/options* de ejemplo puede parecerse a este:

```
# Opciones globales para el PPPd de vlager.vbrew.com

auth            # obligar a autentificación
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 continuacion. La expresión **lock** hace que el ***PPP****d* 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 esta siendo usado. Esto es necesario para evitar

que otros programas, como pueden ser el *minicom* o el *uucico*, abran el dispositivo de serie mientras este es usado por el ***PPP***.



La razón de poner estas opciones en el fichero de configuración global es que estas 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 mas 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 tenia que establecer la conexión manualmente antes de poder ejecutar el ***PPP****d*. Al contrario que el *dip*, el ***PPP****d* no tiene su propio lenguaje de *scripts* para llamar al sistema remoto y entrar en el, sino que confía en otro programa externo para que haga esto. El comando que tiene que ser ejecutado puede dársele al ***PPP****d* con la opción *connect* en la línea de comando. El ***PPP****d* 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 ***PPP****d* 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 envié el mensaje de petición de usuario y entonces le devuelve el nombre del usuario **b1ff**. Solo 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 envié 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 numero de teléfono del servidor es el 318714. En ese caso, la línea completa del chat para que pudiésemos establecer una conexión con **c3po** seria:

$ chat -v " ATZ OK ATDT318714 CONNECT " login: ***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 el, hacemos que el *chat* la ignore especificando una cadena vacía. Continuamos enviando **ATZ**, el comando de inicialización para los módems compatibles Hayes, y esperamos a que nos responda con **OK**. La siguiente cadena envía al chat el comando de marcado junto con el numero de teléfono, y espera a que aparezca el mensaje **CONNECT** como respuesta. Esto esta 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.

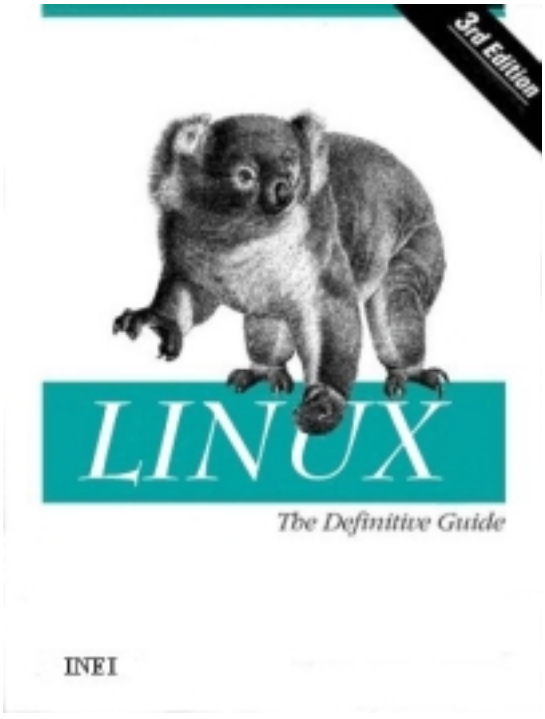
La opción -v hace que el chat capture todas las actividades hacia la facilidad local2 del demonio *syslog*.

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

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 ***PPP****d* tendrá ahora un aspecto como este:

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

Además de la opción *connect* que se refiere al script de llamada, hemos añadido dos opciones mas a la línea de comando: *-detach*, que le indica al ***PPP****d* que no se separe de la consola ni se vuelva proceso de segundo plano. La palabra *módem* activa algunas acciones especificas para módem sobre el dispositivo de serie, como colgar la línea antes y después de la llamada. Si no utiliza esta opción, el ***PPP****d* no se preocupara 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 numero 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 mas detalles, vea la pagina 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 envió 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 subscript enviando un **BREAK** y esperando de nuevo por el mensaje de login. Si ahora ya aparece, el script continua como usualmente y si no, termina con un error.

8.6 Depuración de la configuración *PPP*

Por defecto, el ***PPP***d registrara 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 desechara 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 que 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 ***PPP****d* 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 mas drástica es el activar la depuración a nivel de kernel llamando al ***PPP****d* 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 conexion. Normalmente, cada extremo de comunicación puede enviar un Paquete de petición de configuración IPCP, indicando que valores quiere cambiar de los que vienen por defecto, y a que valor. Tras la recepción, el extremo remoto inspecciona cada opción sucesivamente, y, o responde que la acepta, o la rechaza.

El ***PPP***d le da gran control sobre que opciones intentara 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 ***PPP***d 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 ***PPP***d soporta diferentes posibilidades sobre este aspecto.

Para pedir direcciones particulares, normalmente de al ***PPP***d la siguiente opción:

dir_local :dir_remota

donde **dir_local y dir_remota** pueden ser especificadas en notación de cuádruplas númericas o como nombres de ordenador. Esto hace al ***PPP***d 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**.

Si solo 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 unicamente, dejaría el campo de la ***dir_local*** en blanco. Por defecto, el ***PPP***d utilizara 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 estos servidores, debe asegurarse de que el ***PPP***d 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 interfase de red, el ***PPPd*** preparara un encaminamiento que solamente le sirve para comunicarse con el otro extremo. Si el ordenador remoto esta en una red de área local, seguramente usted deseara conectar también con los ordenadores que están "detrás" de el; para eso, se ha de configurar un encaminamiento de red adecuado.

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 solo 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 este conectando a **vlager** a través de ***PPP***, el utiliza una dirección de la subred de la Cervecera. Podremos dar al ***PPPd*** del ordenador **vlager** la opción **proxyarp**, que instalara 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 especifica, 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 numero 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** actuara como pasarela; la otra maquina se llama sub-etha y tiene una dirección **IP** de **191.72.3.1.**.

Cuando **sub-etha** conecta a **vlage**r, 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 interfase ***PPP*** ha sido configurado. Cuando esta presente, se le llama con los siguientes parámetros:

ip-up interface dispositivo velocidad dir local dir remota

donde **interfase** 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 aun no esta 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 este 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 pasara el nuevo encaminamiento a todos los ordenadores de las redes dependientes de esta.

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 mas importantes que pueden ser negociadas por el **LCP** son la unidad máxima de recepción (**MRU**) y el mapa de caracteres de control asincronos. También hay varias opciones de configuración **LCP** mas, pero son demasiado especificas como para comentarlas aquí. Eche un vistazo a la **RFC 1548** para ver una descripción de estas.

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 especifica 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 caracter 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.

El mapa asíncrono (*async map)* es un mapa de bits de 32 bits de ancho, y cuyo bit menos significativo corresponde al caracter **ASCII NUL**, y cuyo bit mas 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 solo de algunos, puede establecer un nuevo mapa asíncrono al ***PPPd*** utilizando la opción asyncmap. Por ejemplo, si solo **^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 esto puede que le recuerde al valor de la **MTU** (unidad máxima de transferencia), tienen poco en común. El **MTU** es un parámetro del dispositivo de red del kernel, y describe el tamaño máximo de la trama que el interfase es capaz de soportar. El **MRU** es mas bien un consejo al ordenador remoto para que no genere ninguna trama mas grande que la **MRU**; no obstante, el interfase ha de ser capaz de recibir tramas de hasta 1500 bytes.

Por lo tanto, elegir un **MRU** no es tanto una cuestión de que es capaz de transmitir la conexión, sino de como 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 **MRU**s pequeñas, de todas maneras, solo tienen sentido si no tiene la compresión de cabecera **VJ** desactivada (esta 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 numero de peticiones de configuración que pueden ser intercambiadas antes de que se corte la conexión. A menos que sepa exactamente lo que esta haciendo, deberá dejar este valor fijo.

Finalmente, hay dos opciones que se aplican a los mensajes de eco del **LCP**. El ***PPP*** define dos mensajes, "petición de Eco" y "Respuesta de Eco". El ***PPPd*** usa esta característica para comprobar si la conexión esta aun 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 ***PPP****d* 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 esta 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 maquina 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 ***PPP****d* 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 ***PPP****d* 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 tratata 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. Mas adelante, utilizare relajadamente 'cliente' y 'servidor' cuando quiera

distinguir entre el sistema autentificado y el autentificador. Un demonio ***PPP*** puede pedir a la otra maquina autentificación enviando otra petición mas 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 si 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 utilizara para autentificarnos con el y viceversa.

Por defecto, el ***PPPd*** no pide autentificación al ordenador remoto, pero aceptara el autentificarse a si mismo cuando se lo pida el ordenador remoto. Como el **CHAP** es mucho mas 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, renunciara a autentificarse. Como consecuencia de esto, se cerrara la conexión.

Este comportamiento puede ser modificado de varias formas. Por ejemplo, cuando se añade la palabra **auth**, el ***PPPd*** solicitara al otro ordenador que se autentifique. El ***PPPd*** aceptara 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 pagina de manual del ***PPPd****(8)* para mas 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 el 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 maquina 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**.

```
# claves CHAP para vlager.vbrew.com
#
# Cliente          Servidor          Clave                    Dirección
# 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 envió 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,** 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 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 esta en esta lista durante la negociación **IPCP**, la petición será rechazada, y **IPCP** se desconectara. En el fichero de ejemplo anterior, **c3po** esta limitado a poder usar solo su propia dirección. Si el campo de dirección esta vacío, se permitirá cualquier dirección; un valor de "-" evita el uso de una cierta dirección **IP** con un cliente.

La tercera línea del fichero *chap-secrets* de prueba, permite a cualquier ordenador establecer un enlace ***PPP*** con **vlager**, pues si aparece la expresión "*" en los campos de cliente o servidor, será valido cualquier nombre. El único requisito es que sepa la clave, y utiliza la dirección de **pub.vbrew.com.** Pueden aparecer perfectamente entradas con comodines en los nombres en cualquier lugar del fichero de claves, pues el ***PPPd*** siempre utilizara la entrada mas especifica 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 explico 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*** utilizara este como el nombre local. Para mas detalles, lea la pagina 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 si 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 este:

```
#  /etc/PPP/pap-secrets
#
# usuario        servidor     clave          dirección
  vlager-pap       c3po        cresspahl       vlager.vbrew.com
  c3po             vlager      DonaldGNUth     c3po.lucas.com
```

La primera línea se usa para autentificarnos a nosotros mismos cuando hablemos con **c3po**. La segunda línea describe como un usuario llamado **c3po** tiene que autentificarse con nosotros.

El nombre **vlager-pap** de la primera columna es el nombre de usuario que nosotros mandamos a **c3po.** Por defecto, el ***PPPd*** tomara 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*** :

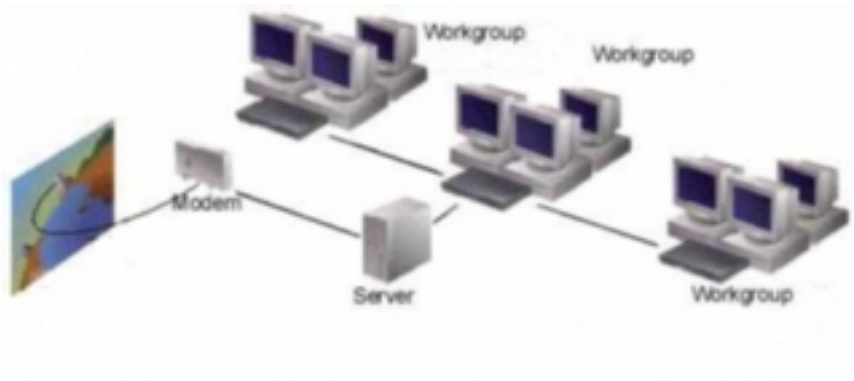
```
# PPPd ... remotename c3po user vlager-pap
```

En el cuarto campo (y todos los siguientes), puede especificar que direcciones **IP** están permitidas para ese ordenador particular, de la misma forma que en el fichero de claves del **CHAP**. El otro ordenador solo podrá pedir direcciones de esa lista. En el fichero de ejemplo, obligamos a **c3po** a usar su dirección **IP** autentica.

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 esta interesado en utilizar el **PAP**, encontrara algunas características. Mas de este comentadas en la pagina del manual del ***PPPd(8).***

8.11 Configuración de un Servidor *PPP*

Hacer funcionar el ***PPP****d* como servidor es solo 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 ***PPP***d con estas opciones. Por ejemplo, podría añadir la siguiente línea a */etc/passwd* :



PPP:*:500:200:Cuenta ***PPP*** Publica:/tmp:/etc/***PPP***/***PPP***login

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 ***PPP***login 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 envié le será devuelto a modo de eco. La opción del ***PPP****d* más importante de las incluidas en el script es **-detach**, porque evita que el ***PPP****d* 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 ***PPP****d* 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 **módem** hace al ***PPP****d* 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 ***PPP****d*, o en el fichero de opciones globales. La pagina del manual también habla sobre opciones mas especificas 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 capitulo trata la configuración de algunas sencillas aplicaciones de red, incluyendo el servidor *inetd*, y los programas de la familia *rlogin*. El interfase de *Llamada a Procedimiento Remoto* o **RPC**, 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**, también será tratado brevemente aquí. Las configuraciones de **NFS** y de **NIS**, sin embargo, ocupan mas 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, refierase 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 aceptara la conexión, mientras el padre continua escuchando a la espera de mas 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 unix corren un "super-servidor" que crea sockets para varios servicios, y escucha en todos ellos simultáneamente usando la llamada al sistema *select (2)*. Cuando al nodo remoto uno de los servicios requiere, el he/she del súperservidor recibe lo y he/she llama al sirviente especificado para ese puerto.

El súper-servidor mas 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 configuracion 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 numero de puerto consultando el fichero */etc/services*. Este fichero será descrito mas 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. Este debe ser un nombre de protocolo valido que se pueda encontrar en el fichero *protocols*, también descrito mas adelante.

Espera Esta opción se aplica solo a sockets de tipo *dgram*. Puede tomar los valores *wait* o *nowait*. Si se especifica *wait*, *inetd* ejecutara solo un servidor cada vez para el puerto especificado. De otro modo, continuara 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 mas, 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 numero ilimitado de instancias corriendo concurrentemente. Con estos servidores se deberá especificar *nowait*.

Para sockets de tipo stream se deberá especificar siempre *nowait*.

usuario Este es el identificador del usuario bajo el que se ejecutara el proceso. Por lo general,este 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 ejecutara 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

Esta es la línea de comando a pasar al servidor. Esto incluye el argumento 0, es decir, el nombre del comando. Normalmente, este 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 se muestra un ejemplo de fichero */etc/inetd.conf*. La línea del servicio *finger* esta comentada, de forma que no este 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 esta 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 estas aplicaciones, sin embargo, pueden ser reventadas (lo que quedo bastante demostrado con el RTM *Internet worm*), o pueden no distinguir entre un nodo seguro cuyas peticiones de un servicio

```
#
# 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
```

Un ejemplo de fichero */etc/inetd.conf*.

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* mas 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*, el denominado demonio *envoltorio*. Para los servicios **TCP** que quiera monitorizar o proteger, este es invocado en lugar del programa Servidor. *tcpd* informa de la petición al demonio *syslog*, chequea si el nodo remoto esta autorizado para usar ese servicio, y solo si la respuesta es satisfactoria, ejecutara 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 esta 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* (eneste 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** hace coincidir todos los nombres de nodos que no contengan un punto. *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** encajara con **biff.foobar.com.** También hay formas de especificar direcciones de red **IP** y números de subred. Por favor, refierase a la pagina del manual de *hosts_access(5)* para mas 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 "petición 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. Refierase a la pagina del manual de *hosts_access(5)* para mas 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". 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 que protocolo de transporte se usa. comúnmente, este es *udp* o *tcp*. Es posible que un servicio sea ofrecido a mas de un protocolo, lo mismo que es posible ofrecer distintos servicios por el mismo numero 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                  # Protoc. 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          # Notificación de correo
login       13/tcp                  # login remoto
```

```
who         513/udp  whod          # who  y  uptime  remotos
shell       514/tcp  cmd           # comando  remoto,  si  contraseña
syslog      514/udp               # registro  remoto  del  sistema
printer     515/tcp  spooler       # cola  de  impresión  remota
route       520/udp  router routed  # información  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 numero asociado. Necesitar modificar este fichero es todavía mas 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 transmisión
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 invocara 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 por el segundo, y convertidos otra vez a la representación de la maquina local por el receptor.

A veces, las mejoras en una aplicación **RPC** introducen cambios incompatibles en el interfase de llamada a procedimiento. Por supuesto, solo cambiando el servidor dejaría de funcionar cualquier aplicación que todavía espere el comportamiento original. Por ello, los programas **RPC** tienen números de versión asignados, normalmente empezando con 1, y con cada nueva versión del interfase **RPC** este contador se incrementara. A menudo, un servidor puede ofrecer varias versiones a la vez; entonces los clientes indicaran en sus peticiones mediante el número de versión que implementación del servicio desean usar.

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 deestos es llamado *programa*, y es identificado unívocamente por un numero de programa. En */etc/rpc* se suele mantener una lista que mapea nombres de servicios con números de programa, reproducimos un extracto de esto:

```
#
# /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
```

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 usaran **UDP** cuando envíen datos, y solo 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 que puerto mapea un numero de programa. Usando un fichero de configuración para esto seria 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 cualquien puerto que puedan utilizar, y lo registran con el denominado *demonio mapeador de puertos*. Este último actúa como un distribuidor de servicios para todos los servidores que corren en su maquina: un cliente que desee contactar con un servicio que tiene un numero de programa dado, preguntara primero al mapeador de puertos del nodo del servidor quien 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 estandar. De todas formas, este 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 maquina.

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 mas 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 indicara 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 maquinas de su **LAN**, tal vez desee ser admitido sin tener que escribir su password cada vez.

Deshabilitar autorizaciones sólo es aconsejable en un numero reducido de nodos cuyas bases de datos de passwords estén sincronizadas, o para un numero reducido de usuarios privilegiados que necesiten acceder a muchas maquinas 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 súper 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. Estos serian listados en el fichero *.rhosts* en el directorio *home* del usuario. Por

razones de seguridad, este fichero debe pertenecer al usuario o al súper usuario, y no debe ser un enlace simbólico, de otro modo será ignorado.

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 pretende entrar. Como ejemplo, asumamos que **janet** esta 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* si a Janet se le puede proporcionar acceso libre y, si esto falla, intentara 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 mas, 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 ultima 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 **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**.

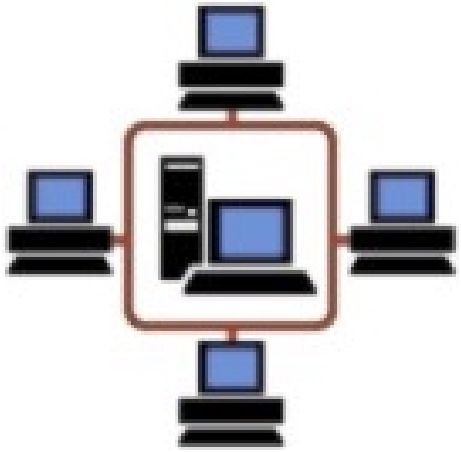
Observe que el nombre de nodo del cliente se obtiene mediante la resolución inversa de la dirección del que llama a un nombre, de forma que esta característica fallara con nodos desconocidos para el sistema de resolución. El nombre de nodo del cliente se considera que coincide con el nombre en los ficheros 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.

```
;
    ; información 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
;
; Física Teórica (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
...
```

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. Mas aun, si usted solo esta 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 desarrollo **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 maquinas de la red.

NIS esta basado en **RPC**, e incluye un servidor, una biblioteca para la parte del cliente, y varias herramientas de administración. Originalmente **NIS** se llamaba *Yellow Page* , 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** continua viviendo como prefijo en los nombres de la mayoría de los comandos relacionados con **NIS** como *ypserv, ypbind,* etc.

Hoy en día **NIS** esta disponible en prácticamente todos los sistemas unix, y hay incluso implementaciones gratuitas de el. Una de ellas es de la edición **BSD** Net-2, derivada de una implementación de referencia de dominio publico 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 Thummler. 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.*

Actualmente, el código de **NIS** esta siendo reescrito por completo por Peter Eriksson, se denominara **NYS** y soportara tanto el **NIS** normal como la revisión ampliada de Sun, el **NIS+**. **NYS** no solo 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 mas adelante.

Este capítulo se centrara en **NYS** mas 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 (vease [Stern92]).

Por el momento **NYS** esta 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 enlazara 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 esta ejecutando el servidor **NIS** y del que los clientes pueden obtener la información a través de varias llamadas **RPC**.

Muy frecuentemente, los mapas se almacenan en ficheros **DBM**.

Los mapas en si 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 el se derivan dos mapas **NIS**, llamados *hosts.byname* y *hosts.byaddr* respectivamente. La tabla 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	

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**. Estos pueden contener información sobre aplicaciones no tratadas en este libro, como el mapa *bootparams* 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 mas cortos y por lo tanto más fáciles 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"
        "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 maquinas 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 crearan solo en la maquina 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 solo una función puramente administrativa. Son además invisibles para los usuarios. Por ello el nombre dado a un dominio **NIS** es solo 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 si, 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 que servidor **NIS** preguntaran las aplicaciones. Por ejemplo, el programa *login* de un nodo de la Vinatera debería, por supuesto, pedir informacion de la contraseña de un usuario solo 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: como sabe un cliente a que servidor conectarse. La solución más simple seria 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*, que 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 mas que su nombre y un objetivo común. **NIS+** esta 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 numero de versión **RPC** de 2, mientras que **NIS+** tiene un numero de versión **RPC** de 3.

NIS+ no parece ser ampliamente usado todavía, y no conozco tanto sobre el realmente (bueno, prácticamente nada). Por esta razón, no lo trataremos aquí. Si esta interesado en aprender mas sobre el, por favor, refierase al manual de administración de **NIS+** de Sun ([NISPlus]).

10.3 El lado cliente de NIS

Si esta usted familiarizado con la escritura o el portado de aplicaciones de red, notara 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 realizaran la búsqueda requerida en el fichero estándar */etc/passwd.*

Una implementación de estas funciones que tenga en cuenta **NIS**, sin embargo, modificara este comportamiento, y realizara 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 el. 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 que mapas se sustituían y cuales 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 que orden. Esto será descrito en otra sección más adelante en este mismo capitulo.

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 seccion.

- 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 este 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 Tobías Reber, y el otro en el paquete *ypserv* de Peter Eriksson. No debería importar cual 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 mas 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*, debera 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 esta sirviendo un dominio **NIS** en particular comprobando si el directorio mapa esta 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 dificil 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 encontrara 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 solo uno o dos mapas, como los mapas *services.*.*

Tras editar el *Makefile,* y sin salir del directorio de mapas, teclee "**make**". Esto generara e instalara automaticamente 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 como 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**,este imprimirá mensajes de depuración en la consola sobre todas las peticiones **NIS** entrantes, y los resultados devueltos. Esto debería darle una idea sobre donde puede estar el problema. El servidor de Tobías no tiene esa opción.

10.5 Configurar un Cliente NIS con NYS

A lo largo de lo que queda de este capitulo, cubriremos la configuración de un cliente **NIS**.

Su primer paso debería ser indicarle a **NYS** que 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 seria algo así:

```
# yp.conf - configuración 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** usara el nombre de dominio que usted asigno 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 direccion **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 maquina 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 - configuración YP para la biblioteca NYS.
#
ypserver vbardolino vinatera
ypserver vstout     cervecera
```

Esto le permite mover su maquina 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 esta disponible. En este ultimo caso, asegúrese de que un *ping* a esa maquina da un resultado positivo, y de que esta 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 este seguro de que puede llegar al servidor **NIS**, debe decidir que ficheros de configuración sustituir o aumentar con mapas **NIS**. Normalmente deseara usar mapas **NIS** para las funciones de búsqueda 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 mas 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 este en el fichero *services* estándar.

Por lo general, usted deseara tener alguna libertad de elección acerca de cuando una función de búsqueda usara ficheros locales y cuando hará una petición al servidor **NIS**.

NIS 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* pero por supuesto no esta limitado a los servicios de nombres. Para cualquiera de las funciones de búsqueda de datos soportadas por **NIS**, 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; unicamente podría contener mas. Así que una buena elección seria consultar los ficheros locales primero, y probar con **NIS** solo 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 mas precisa, mientras que el fichero *hosts* local solo se mantiene como copia de respaldo por si el **DNS** y **NIS** fallasen. En este caso, habría que comprobar el fichero local en ultimo lugar.

El siguiente ejemplo muestra como configurar las funciones *gethostbyname(2), gethostbyaddr(2),* y *getservbyname(2)* de la forma descrita anteriormente. Probaran los servicios listados por orden; si una búsqueda es satisfactoria, se devuelve el resultado, si no, se intentara con el siguiente servicio.

```
# /etc/nsswitch.conf de ejemplo

#
hosts:      nis dns files
services:   files nis
```

Mas 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 esta configurada en el fichero *yp.conf* como se mostró en la seccion 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 solo es útil con la entrada *hosts.* Los servidores de nombres consultados siguen estando determinados por el fichero estándar *resolv.conf.*

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 mas entradas.

La figura muestra un ejemplo mas 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** continuara y buscara en los ficheros locales *solo* si las llamadas a los servidores **NIS** y **DNS** fallaron por alguna otra razón. Los ficheros locales serán usados entonces solo durante el arranque y como copia de respaldo cuando el servidor **NIS** se haya caido.

```
# /etc/nsswitch.conf
  #
  hosts:      nis dns [NOTFOUND=return] files
  networks:   nis [NOTFOUND=return] files

  services:   files nis


protocols:  files nis
  rpc:        files nis
```

Fichero *nsswitch.conf* de ejemplo.

10.7 Uso de los mapas passwd y group

Uno de las usos mas importantes de **NIS** es sincronizar usuarios e información de cuentas en todos los nodos de un dominio **NIS**. Para este fin, solo 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, solo 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 identificadotes 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 deseara 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 recien modificados, y finalmente cambiarles el propietario. Supongamos que **news** tenia un numero de usuario de 9, y que **okir** tenia un numero de usuario de 103, y que estos 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 usara 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 lineas 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 borrara la mayor parte de los usuarios de sus ficheros locales y sólo dejara 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 fallaran miserablemente durante un apagon 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 *útil-linux*. Los intrincados métodos para usar contraseñas shadow con **NIS** serán cubiertos mas 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* mostrara los nombres simbólicos del propietario y del grupo; esto es, por cada uid y gid que encuentre, debera hacer una peticion al servidor **NIS**. Esto ralentizará mucho todo el sistema cuando su red local se atasque, o, aun peor, cuando el servidor **NIS** no este 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 invocara el comando *passwd*, que leerá la nueva contraseña y actualizara el fichero *passwd* local. Esto es imposible con **NIS**, puesto que ese fichero no esta 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 analoga 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 ocultara 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*, publico 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 esta completa, y los ficheros no han sido añadidos todavía a la biblioteca C estándar. Por otro lado, publicar la información de */etc/shadow* vía **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* mirara si existe */etc/shadow* y lo intentara abrir. Si no existe, o si el usuario no tiene privilegios de **root**, volverá al comportamiento tradicional de mirar la información del usuario solo 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** funcionaran de forma transparente con una instalación *shadow* local.

10.9 Uso del Código NIS Tradicional

Si esta 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 mas arriba.

Recientemente 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 mas recientes de las utilidades **NIS** y recompilarlas se debería solucionar este problema.

Del mismo modo, la forma en que el **NIS** tradicional decide si hay que mezclar información **NIS**, y como, 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* :

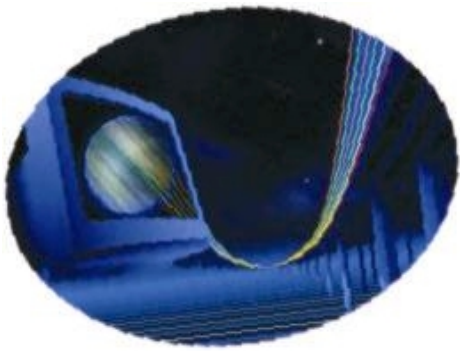
```
+:*:0:0:::
```

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 linea por:

```
order yp bind hosts
```

La implementación **NIS** tradicional no soporta ningún otro mapa mas por el momento.

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 mas 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 maquina, 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 maquinas.

- 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 maquinas distintas.

El **NFS** de Linux es, principalmente, obra de Rick Sladkey,, 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.

Veamos ahora un poco como 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**:

```
# mount -t nfs vlager:/home /users
```

mount intentara conectar con el emonio remoto *mountd* mediante **RPC**. El servidor comprobara si la maquina **vale** tiene permiso para montar el directorio pedido, y si es así retornara un descriptor de fichero. Este descriptor se utilizara 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 maquinas deberían ser los mismos.

En varias implementaciones de unix, 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 maquina. 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 esta 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 originaria.

El mayor problema con el código **NFS** de Linux es que el núcleo 1.0 no puede manejar bloques de memoria de mas 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 perdidas de rendimiento en ciertas circunstancias. Esta limitación desapareció en los núcleos posteriores al 1.1, rescribiéndose el código del cliente para aprovechar la nueva situación.

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 como configurar el núcleo hablamos en la sección "configuración del núcleo" del capitulo 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 volumenes NFS 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
```

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 incomodas. Por ejemplo, puede que tenga que teclear cosas como esta, para especificar que se límite 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 *útillinux* de Rik Faith. Las opciones mas 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 limites 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 centesimas 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 solo 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**, esperara 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 envié 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 caché, hace que no se sepa realmente si una operación de escritura ha tenido éxito o no, a menos que el volumen este montado de forma física.

En general, se recomienda el montaje físico salvo en caso de tratarse de información no critica, 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 envió de se"añiles para interrumpir las esperas en caso de necesidad.

Normalmente, el demonio *mountd* llevara de alguna forma un registro de que directorios están montados desde que maquinas. 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 maquinas, deberá ejecutar en el servidor los programas *nfsd* y *mountd*. Son programas basados en **RPC**, por lo que no son arrancados desde el i*netd*, 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 lineas 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 resultara útil si clientes y servidores tienen el mismo mapa de usuarios y rupos, 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 indicara 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 maquina. Para permitir que algún nodo monte un directorio, este debe estar *exportado*, es decir, especificado en el fichero de exportación. Un ejemplo de dicho fichero es el siguiente:

```
# Fichero de exportación 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 maquinas que pueden acceder a el por **NFS**. Un nombre de maquina 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 maquina 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 exportacion. 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:

Insecurep 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 esta activa por defecto.

secure-rpc Requerir autentificación **RPC** segura para este nodo. Aun no esta implementado. Se sugiere ver la documentación de Sun al respecto (vease, "*Secure RPC*").

kerberos Requerir autentificación Kerberos. Tampoco se ha implementado aun. Se sugiere consultar la documentación del **MIT**.

root_squash Se trata de una opción de seguridad que deniega acceso a nivel de superu suario, 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 solo lectura. Es una opción por defecto.

rw Monta el directorio con permiso para leer y escribir en él.

link_relative Convierte enlaces simbólicos absolutos (que empiezan con una barra de directorio, "/") en enlaces relativos colocando los prefijos ".../ " que sean necesáreos 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 automaticamente 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 auto montado 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 capitulo. 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 mas recientes de la versión 2 ofrecen muchas de las características de los tipos mas nuevos de **UUCP**.

La segunda clase de **UUCP** se desarrollo 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 Version 2. Por ejemplo, se añadieron 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, que es la versión en la que se basa este capitulo. La versión 1.04 de Taylor **UUCP** esta disponible desde Febrero de 1993. Además de los tradicionales ficheros de configuración, Taylor **UUCP**

también 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 usara, 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 más fácil de entender que los usualmente oscuros ficheros de configuración de **BNU**, describiré aquí el estilo Taylor.

El propósito de este capitulo 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 como poner en marcha un nodo de **UUCP**. La primera sección presenta una introducción de como **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 esta familiarizado con los programas de usuario del paquete **UUCP**. Estos son *uchú* y *uux.* Si no los conoce suficientemente, consulte las correspondientes paginas de manual.

Aparte de los programas de usuario, *uux* y *uucp*, el paquete **UUCP** contiene algunos comandos mas, usados solamente para fines administrativos. Se usan para observar él trafico 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**. Estas son **uucico** (donde "cico" significa "copy-in copy-out"), y *uuxqt*, que ejecuta trabajos enviados desde sistemas remotos.

12.1.1 Mas información Sobre UUCP

Aquellos que no encuentren todo lo que necesiten en este capitulo pueden leer la documentacion 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 especificas 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 Transferencias 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 árbol 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 cosas.

Además 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 **UUCP**s que son compatibles con **HDB** generalmente guardan los ficheros de cola en un directorio llamado */var/spool/uucp/**maquina***, donde **maquina** es el nombre del ordenador remoto. Si fue compilado para usar ficheros de configuración de Taylor, **UUCP** crea subdirectorios bajo el directorio de cola especifico 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

después de que la conexión haya finalizado. Esto lo hace *uuxqt,* que también se ocupa de reenviar cualquier trabajo que este 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 digito, 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 mas 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 solo si es *igual* o *mayor* que el máximo nivel de cola.

12.2.2 El Funcionamiento Interno de*uucico*

Para comprender por que *uucico* necesita saber ciertas cosas, hemos de revisar como 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 numero de teléfono. Sobre **TCP**, tiene que llamar *gethostbyname(3)* para convertir el nombre a una direccion de red, averiguar que puerto abrir, y conectar la direccion 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 *dialogo 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 inicio la conexión se denomina *maestro*, y la copia remota se denomina *esclavo*.

A continuación viene la fase de *negociacion*: El maestro envia su nombre, además de varias opciones. El esclavo comprueba el nombre para ver si tiene permiso para conectarse, para enviar y recibir ficheros, etc. Las opciones describen (entre otras cosas) el nivel máximo de ficheros de cola que hay que transferir. Si esta opción esta activada, tiene lugar una cuenta de conversación, o *comprobación de la secuencia de llamada.* Con esta característica, ambos ordenadores mantienen una cuenta de conexiones exitosas, que se comparan. Si las cuentas no son iguales, la negociación de protocolos no tendrá lugar. Esto es útil para protegerse de impostores.

Finalmente los dos *uucico* tratan de ponerse de acuerdo en un *protocolo de transferencia* común. Este protocolo gobierna la manera en que los datos se transfieren, la manera en que se comprueba la consistencia de los datos, y la manera en que se retransmiten en caso de error. Hacen falta protocolos diferentes debido a los diferentes tipos de conexiones que se soportan. Por ejemplo, las líneas de teléfono precisan un protocolo "seguro" que es pesimista respecto a errores, mientras que una transmisión de **TCP** es fiable y puede usar un protocolo mas 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 especifica 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 *uucicos* intercambian mensajes de terminación, y cierran la comunicación.

No vamos a profundizar en mas detalle: para esto, diríjase a las fuentes o a cualquier buen libro sobre **UUCP**. también existe un articulo muy antiguo en la red, escrito por David A. Novitz, que da una descripción detallada del protocolo **UUCP**. La **FAQ** de Taylor **UUCP** también trata algunos detalles de como **UUCP** esta implementado. Se puede encontrar regularmente en **comp.mail.uucp.**

12.2.3 Opciones de la línea de comandos de*uucico*

En esta sección de describen las opciones mas importantes de la línea de comandos del programa *uucico*. Para obtener una lista completa, consulte la pagina del manual de *uucico(1).*

-s sistema Llamar al mencionado sistema si no esta 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 usan **-s** o **-S**.\ Por si sola, la opción **-**r1 hace que *uucico* intente llamar todos los sistemas en *sys*, a no ser que este 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 suponenconectadas a un puerto serie, o al puerto de **TCP** especificado por la opción -p si se usa esta.

-x tipo , -X tipo

Activar la información para resolver problemas del tipo especificado. Se pue den especificar varios tipos en una lista separada por comas. Los siguientes son tipos validos: *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 numero en vez del tipo, lo cual activa las **n** primeras opciones de la anterior lista.

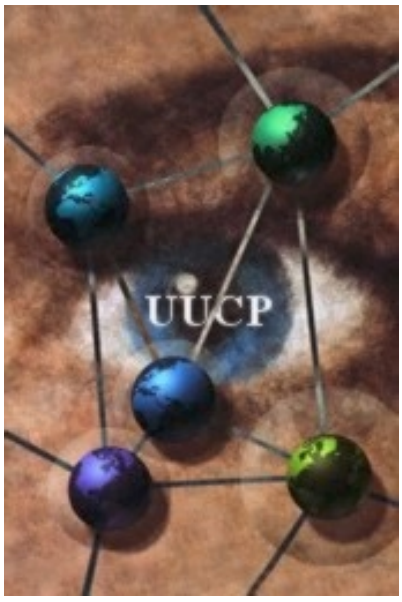
Los mensajes generados se registran en el fichero Debug bajo */var/spool/uucp.*

12.3 Ficheros de configuración de UUCP

Al contrario que programas de transferencia de ficheros mas simples, **UUCP** fue diseñado para ser capaz de llevar a cabo todas las tranferencias automaticamente. Una vez que esta 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 seria 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 mas fáciles (aunque el formato de Taylor es casi fácil de leer comparado con los formatos mas antiguos en **HDB** o versión 2).



Para darle una idea de como se interactúa con estos ficheros, le introduciremos los más importantes, y echaremos un vistazo a algunos ejemplos. No explicaremos ahora todo en detalle; una explicacion 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 este. 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 mas 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 configuración de UUCP
hostname        vstout
```

El siguiente fichero de configuración en importancia es el fichero *sys*. Este contiene toda la información especifica 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 seria:

```
# /usr/lib/uucp/sys - Vecinos UUCP
# system: pablo
  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 dialogo de entrada - la secuencia de caracteres que hay que intercambiar para permitir a *uucico* que conecte con **pablo**. Volveremos a las macros mas 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*.

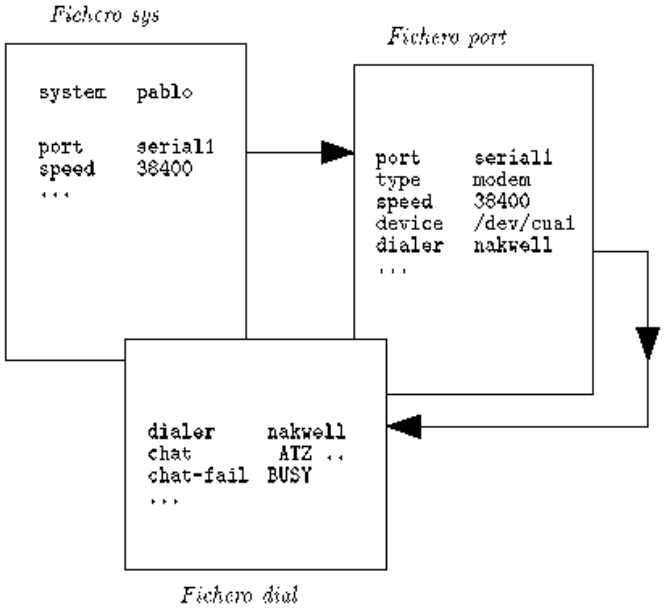
El fichero *port* contiene información especifica a la propia conexión. Para conexiones por módem, describe el fichero de dispositivo a usar, el conjunto de velocidades soportadas, y el tipo de equipo de marcación conectado al puerto. La entrada a continuación describe */dev/cua1* (o sea, el puerto COM 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 numero de teléfono. Una vez mas, esto se especifica como una macro de dialogo. Por ejemplo, la entrada para el anterior NakWell puede parecerse a esta:

```
# /usr/lib/uucp/dial - información por marcador.
# NakWell modems
 dialer         nakwell
 chat           "" ATZ OK ATDT\T CONNECT
```

La línea que empieza con chat especifica el dialogo del módem, que no es sino la secuencia de comandos enviados y recibidos del módem para inicializarlo, y para hacerle marcar el numero deseado. La secuencia "\T " será reemplazada con el numero de teléfono por el programa *uucico*.



Interacciones de los Ficheros de configuración de Taylor UUCP.

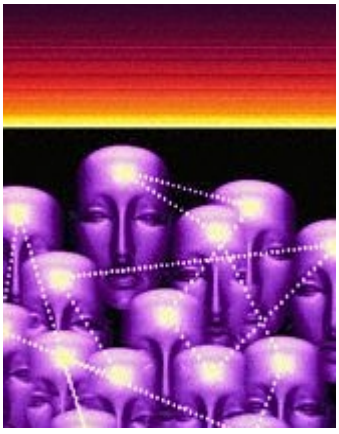
Para darle una idea a grandes rasgos de como uucico utiliza estos ficheros de configuracion, suponga que utiliza el comando $ uucico -s pablo en la linea de comandos. Lo primero que uucico hace es buscar pablo en el fichero sys. A partir de la entrada en el fichero sys para pablo, el programa ve que 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 dialogo de marcación. Esto es, envía "ATZ", espera a que el módem responda con "OK", etc. Cuando se encuentre los caracteres *"\T ",* sustituye el numero de teléfono (123-456) obtenido del fichero *sys.*

Cuando el módem devuelve **CONNECT,** la conexión se ha establecido, y el dialogo de marcación se ha completado. *uucico* ahora vuelve al fichero *sys* y ejecuta el dialogo 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 de que el sistema remoto ejecuta su propio *uucico*. Los dos entraran entonces en la fase de negociación de protocolos descrita en la sección anterior.

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 que dispositivo serie esta 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 con figuran 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 cuando un puerto serie esta 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 mas detalles sobre como configurar sus puertos serie, lea el capitulo 4.

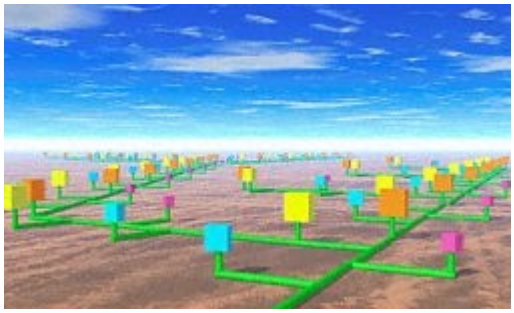
A continuación tiene que averiguar a que velocidad se comunicaran su módem y Linux.Tendrá que ajustar este valor a la velocidad de transferencia efectiva máxima que espere obtener. La velocidad efectiva puede ser mucho mayor que la velocidad física de transferencia de su módem. Por ejemplo, muchos módems 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, necesitara el numero de teléfono al que llamar. también necesitara un nombre de usuario valido y probablemente una clave en el sistema remoto.

Si solo quiere probar **UUCP**, obtenga el numero 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 necesitara saber *exactamente* como 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 dialogo*, que es un script que le dice a *uucico* como entrar. Si no lo sabe, o si la macro de dialogo 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 maquinas necesitan tener un nombre para la red de **UUCP**. Mientras solo 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.

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 capitulo 13. Incluso si usted participa en un dominio, podria 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 ano ser que lo haya registrado como su nombre de **UUCP** oficial. Lo mejor que puede pasares 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 esta usando, este correo será dirigido aese sitio, y le causara al administrador del correo un sin fin 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 dio 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.

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 que horas llamarlo, que numero marcar, que tipo de dispositivo usar, y como 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 conexión telefónica.

dial codee 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 solo 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 si 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

estos usted podrá configurar una conexión de **UUCP** por módem. Otras secciones describiran 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 usara este fichero para otra cosa que describir su nombre de nodo **UUCP**. Por defecto, **UUCP** usara 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 configuración 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 ultimo se describirá en una sección posterior.

12.3.6 Como 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 parámetros 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 verificara 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 configuraciónes para un mismo sistema (por ejemplo, números de teléfono diferentes que *uucico* puede usar alternativamente), puede especificar *alternasvias*, que se describen mas adelante.

Numero de teléfono

Si para conectar con el sistema remoto hace falta una línea de teléfono, el campo *phone* especifica el numero 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 numero de teléfono.

Cualquier lista de caracteres se puede usar para esconder información que depende decada 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 - traducción de códigos de marcación
    Bogoham        024881
    Coxton         035119
```

Con estas traducciones, se puede usar un numero 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. Una entrada *system* puede usar cualquiera de las dos opciones solas, o ambas. Cuando se busca un dispositivo apropiado en el fichero *port*, solo se eligen aquellos puertos cuyo nombre ido 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 solo tiene que especificar la velocidad deseada. Si tiene varios módems 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 esta en uso.

El diálogo de entrada

Antes ya nos encontramos con la macro del dialogo de entrada, que le dice a uucico como entrar en el sistema remoto. Consiste de una lista de palabras clave, que especifican el texto que se espera y el que se envía por el proceso local de *uucico*. El objetivo es hacer que *uucico* espere hasta que la máquina remota envíe una línea pidiendo el nombre de usuario, y entonces enviar el nombre de usuario, luego esperar a que pida la palabra 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 dialogo sencilla seria parecida a esta:

login: vstout ssword: catch22

Dese 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 razón, usted puede añadir un sub-diálogo a un texto de espera, separado con un guión. El sub-dialogo se ejecuta solo 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 dialogo 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 inmediatamente.
\t	Un caracter de tabulador.
\r	Un caracter de retorno de línea.
\s	El caracter de espacio. Lo necesitamos para incluir espacios en un dialogo.

\n Caracter de nueva línea.

\\ Caracter de barra invertida.

En tiras de caracteres de envio se pueden incluir, ademas de los mencionados anteriormente, los siguientes caracteres:

EOT Caracter de fin de la trasmision (^D).

BREAK Caracter Break.

\c Suprime el envio del caracter nueva linea al final de cada tira de caracteres.

\d Retrasar el envio 1 segundo.

\E Activar la comprobacion de eco. De esta forma, *uucico* esperara a leer el eco de todo lo que escribe en el dispositivo antes de que continue con el dialogo. Se usa principalmente en dialogos de modems (que veremos mas adelante). La comprobacion de eco esta desactivada por defecto.

\e Desactivar la comprobacion de eco.

\K Lo mismo que **BREAK**.

\p Pausa de una fraccion de un segundo.

Alternativas

A veces es deseable tener multiples entradas para un mismo sistema, por ejemplo si se puede acceder al sistema en diferentes lineas de modem. Con Taylor **UUCP** se puede hacer esto definiendo una *alternativa*.

Una entrada alternativa mantiene todas las caracteristicas de la entrada principal, y especifica solamente aquellos valores que tienen que ser cambiados, o añadidos. Una alternativa esta separada de la entrada principal por una linea que contiene la palabre clave *alternate*.

Para usar dos numeros de telefono para **pablo**, habria 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* marcara primero el 123-456, y si no funciona, probara la alternativa. La entrada alternativa retiene toda la otra informacion de la entrada de sistema principal, y altera solo el numero de telefono.

Restringir horas de llamada

Taylor **UUCP** proporciona varios metodos para restringir las horas a las que se pueden efectuar llamadas a un sistema remoto. Una razon para hacer esto seria por las limitaciones que el sistema remoto impone en sus servicios durante horas de oficina, o simplemente para evitar las horas mas caras. Siempre se pueden desactivar las restricciones con la opcion **–S** o **-f** en el programa *uucico*.

Por defecto, Taylor **UUCP** no permite conexiones a ninguna hora, asi que usted *tiene que* especificar algun horario en el fichero *sys.* Si no le importan las restricciones, puede especificar la opcion time con un valor de Any en su fichero *sys*.

La manera mas sencilla de restringir horas de llamada es con la entrada time, seguida de una tira de caracteres que consta de dos campos, dia y hora. El dia puede ser cualquiera de los siguientes: *Mo, Tu, We, Th, Fr, Sa, Su* (que corresponden a Lunes, Martes, Miercoles, Jueves, Viernes, Sabado y Domingo, respectivamente) combinados, *Any* (cualquiera), *Never* (nunca), o *Wk* para los dias laborables. La hora consiste en dos numeros de un reloj de 24 horas, separados por un guion. Especifican el grupo de horas durante las que se pueden efectuar llamadas. La combinacion de los simbolos se escribe sin ningun espacio en blanco entre ellos. Se pueden especificar varios grupos de dia-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 conexion falla, *uucico* no permitira otro intento de llamar al ordenador remoto hasta que transcurra un cierto tiempo. Por defecto, uucico usa un algoritmo de espera exponencial, segun 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 aceptara llamar otra vez en los 5 minutos despues del ultimo intento fallido.

El comando *timegrade* le permite añadir un rango maximo 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 comuni-cacion, mientras que las noticias (news) (normalmente con rango N) seran 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 observacion: la opcion *timegrade* solo se aplica a lo que *su* sistema envia; el sistema remoto puede transferir todo lo que le plazca. Usted puede usar la opcion *call-timegrade* para forzar explicitamente que envie solamente trabajos sobre cierto rango de cola; pero no hay ninguna garantia de que obedecera esta peticion.

Igualmente, el campo *timegrade* no se comprueba cuando un sistema remoto hace una llamada aeste, 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 si 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 que serunico; 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 esta describiendo. Tipos validos 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 *módem*.

En esta seccion solo hablaremos de puertos de modem; los puertos de TCP y las lineas directas seran tratados en una seccion posterior.

Para puertos directos y de modem, 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*.

En el caso de un dispositivo de módem, la directiva *port* tambien determina que tipo de módem esta conectado al puerto. Cada tipo de modem tiene que configurarse de manera diferente. Incluso los modems que dicen ser compatibles con Hayes no tienen por que ser realmente compatibles entre si 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 estos, 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 esta llamando. Por ejemplo, algunos modems antiguos no entienden cuándo un modem rapido trata de conectar a 14400bps; simplemente desconectan la linea en vez de negociar la conexion a 9600bps por ejemplo. Si sabe que el ordenador **plasta** usa un modem tan tonto, usted tiene que configurar su modem 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 situacion anterior pueden ser asi:

```
# 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 usaria ahora serial1 como el nombre del puerto, pero pediria usar la velocidad de 9600bps solamente. uucico usara automáticamente la segunda entrada de port. Todos los otros ordenadores con velocidad de 38400bps en la entrada de sistema seran llamados usando la primera entrada de port.

12.3.8 Cómo marcar un numero - el fichero*dial*

El fichero *dial* describe como 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 marcacion automatico (que era caro) sirviese un banco entero de modems. Hoy, la mayoria de los modems tienen soporte para marcar incluido, asi que la distinción tiende a desaparecer.

De cualquier modo, cada marcador o modem puede necesitar una configuracion 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 mas importante, aparte de esta, es el dialogo del modem, especificado por el comando *chat*. Similar al dialogo de entrada (login), consta de una secuencia de caracteres que *uucico* envia al marcador y de la secuencia que espera recibir como respuesta. Se usa normalmente para reiniciar el modem a un estado conocido, y marcar el numero. El siguiente ejemplo de una entrada de un marcador muestra un dialogo tipico 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 dialogo del modem comienza con "", es decir, que espera una cadena vacia. *Uucico* entonces envia el primer comando (**ATZ**) de inmediato. **ATZ** es el comando de Hayes para reiniciar el modem. Entonces espera hasta que el modem envie **OK**, y a continuacion envia el siguiente comando que desactiva el eco local, y cosas parecidas. Despues de que el módem envie **OK** otra vez, uucico envia el comando de marcacion (**ATDT**). La secuencia de escape **\T** en esta cadena es reemplazada con el numero de telefono obtenido de la entrada de sistema en el fichero *sys*. *uucico* espera a que el modem devuelva la cadena **CONNECT**, que indica que se ha establecido una conexion exitosa con el modem remoto.

A menudo el modem no puede conectarse con el sistema remoto, por ejemplo si el otro sistema esta conectado con otro ordenador y la linea esta ocupada. En este caso, el modem devuelve algun mensaje de error indicando la razon. Los dialogos de módem no pueden detectar estos mensajes; uucico seguira esperando la cadena esperada hasta que un temporizador se agote. El fichero de recopilacion de informacion (log) de **UUCP** mostrara solamente el mensaje "timed out in chat script" (tiempo agotado en la macro de dialogo) en vez de la razon 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 dialogo mientras lo ejecuta, interrumpe la llamada y anota el error en el fichero log de **UUCP**.

El ultimo comando en el ejemplo anterior indica a **UUCP** que cambie la linea **DTR** antes de empezar el dialogo de modem. La mayoria de los modems se pueden configurar para conectarse cuando detectan un cambio en la linea **DTR**, y entrar en modo de comando.

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, segun el cual los articulos se piden y se transmiten individualmente, sin compresion ni ninguna otra optimizacion. Aunque es una tecnica adecuada para ordenadores grandes con varias fuentes de grupos de noticias simultaneas, esta tecnica no es favorable para pequeños sistemas que reciben los grupos a traves de una conexion lenta, como RDSI. Estos ordenadores normalmente desean combinar las cualidades de TCP con las ventajas de enviar articulos en grandes lotes, que se pueden comprimir y por lo tanto transferir con muy poco gasto. Un metodo estandar 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 direccion de internet (IP) del ordenador, o su nombre de dominio completo (FQDN). La entrada correspondiente en el fichero port seria asi:

```
port        tcp-conn
type        tcp
service     540
```

Esta entrada indica que hay que usar una conexion de **TCP** cuando una entrada en el fichero *sys* hace referencia a *tcp-conn*, y que el programa *uucico* debera tratar de conectarse al puerto **TCP 540** en el sistema remoto. Este es el puerto por defecto del servicio **UUCP**. En vez del numero de puerto, tambien se puede especificar un nombre de puerto simbolico. El numero de puerto correspondiente sera buscado en el fichero */etc/services*.

12.3.10 Uso de una conexion directa

Supongamos que usted usa una linea directa que conecta su sistema **vstout** con el ordenador **tiny**. Al igual que en el caso del modem, tiene que escribir una entrada de sistema en el fichero *sys*. El comando port identifica el puerto serie en el que tiny esta 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 conexion directa. La entrada *dialer* no hace falta porque no hay que marcar ningun numero.

```
port        direct1
type        direct
speed       38400
```

12.4 Los sies y noes de UUCP - Ajuste de Permisos

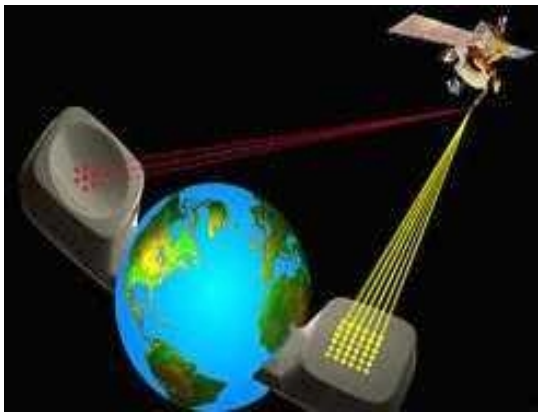
12.4.1 Ejecucion de comandos

La tarea de **UUCP** es copiar ficheros de un sistema a otro, y pedir la ejecucion de ciertos comandos en sistemas remotos. Por supuesto, usted como administrador querra 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 unicos comandos que Taylor **UUCP** permite a otros sistemas ejecutar en su ordenador son *rmail* y *rnews*, que se usan comunmente para intercambiar correo y noticias de Usenet sobre UUCP. El directorio en el que *uuxqt* busca es una opcion 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 busqueda se puede cambiar con el comando *command-path*. Por ejemplo, usted puede querer dar acceso al sistema **pablo** para que ejecute el comando *rsmtp* ademas de *mail* y *rnews*:

```
system          pablo
...
commands        rmail  rnews  rsmtp
```

12.4.2 Transferencias de Ficheros



Taylor **UUCP** tambien 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 sera 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.

Ademas, 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 asi se permite a los usuarios locales que envien ficheros de sus directorios. Comunmente, a los usuarios remotos se les permite que reciban ficheros solo del directorio publico de **UUCP**, */var/spool/uucppublic*. Este es el lugar tradicional para poner los ficheros disponibles publicamente; muy parecido a un servidor de **FTP** en Internet. Normalmente se refiere a este directorio con el caracter tilde.

Por lo tanto, Taylor **UUCP** provee cuatro comandos diferentes para configurar los directorios para enviar y recibir ficheros. Estos son *local-send*, que especifica la lista de directorios desde los que un usuario puede pedir a **UUCP** que envie 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 ~
local-receive    /home ~/recibir
remote-send     ~ !~/entrada !~/recibir
remote-receive  ~/entrada
```

El comando *local-send* permite a los usuarios de su sistema que envien cualquier fichero bajo */home* y en el directorio publico 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 exclamacion delante de los nombres de los directorios. Finalmente, laultima linea permite que **pablo** ponga ficheros en *entrada*

Uno de los mayores problemas con la transferencia de ficheros usando **UUCP** es que solo 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 desactivacion total de la transferencia de ficheros por **UUCP**.

12.4.3 Reenvio

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 envie a su sistema. El siguiente comando haria esto:

```
$ uucp -r seci!uchile!~/find-ls.gz ~/uchile.files.gz
```

Esta tecnica de pasar un trabajo a traves de varios sistemas se llama *forwarding* (reenvio). En el ejemplo anterior, la razon para usar el reenvio 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 reenvio a unos pocos sistemas en que usted confia, para que no se le acumule una factura telefonica horrenda cuando alguien use su sistema para obtener la ultima version de **X11R6**.

Por defecto, Taylor **UUCP** no permite el reenvio. 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 reenvie trabajos. Por ejemplo, el administrador de **UUCP** del sistema **seci** tendria que añadir las siguientes lineas 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 el sea en efecto pasado a **pablo**. De otro modo, **UUCP** se desharia del fichero. Esta entrada usa una variacion del comando forward que permite que **uchile** solo envie ficheros a **pablo** a traves de **seci**, no al reves.

Para permitir el reenvio a cualquier sistema, use el comando especial ***ANY*** (tiene que estar en mayusculas).

12.5 Configuracion de su sistema para ser llamado

Si quiere configurar su sistema para que otros se conecten aeste llamandole, tiene que permitir conexiones en su puerto serie, y modificar ciertos ficheros del sistema para proveer cuentas de **UUCP**. Este es el tema de esta seccion.

12.5.1 Configuracion de getty

Si quiere usar una linea serie como un puerto de entrada, tiene que activar un proceso *getty* en ese puerto. Sin embargo, algunas implementaciones de *getty* no son validas 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 facsimiles. También puede obtener la ultima versión de estos programas en **sunsite.unc.edu**, tanto en binario como en código fuente.

La explicación de las diferencias de como *uugetty* y *mgetty* manejan la entrada al sistema esta mas allá del alcance de esta pequeña sección; para mas información, vea el **HOWTO** Serial de Greg Hankins, así como la documentación que viene con *getty ps* y *mgetty*.

12.5.2 Proveer Cuentas de UUCP

A continuacion tiene que configurar las cuentas de usuarios que permiten a sistemas re- motos entrar en su sistema y establecer una conexion de **UUCP**. Generalmente tendra 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 traves 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 raiz de cada cuenta de **UUCP** tiene que ser el directorio publico */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 aplicacion de claves ocultas, probablemente tendra que editar */etc/passwd* a mano, añadiendo una linea como la siguiente, donde 5000 y 150 son el numero de iden-tificacion de usuario (uid) y el numero 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 asignandole 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 linea al fichero */etc/inetd.conf* :

```
uucp stream tcp nowait root /usr/sbin/tcpd /usr/lib/uucp/uucico -l
```

La opcion -l hace que uucico haga su propia autorizacion de entrada. Pedira un nom-bre de usuario y una clave, igual que el programa estandar *login*, pero usara 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:

```
Upablo IslaNegra
Ulorca cordoba
```

Por supuesto, este fichero tiene que pertenecer al usuario **uucp** y tener permiso 600.

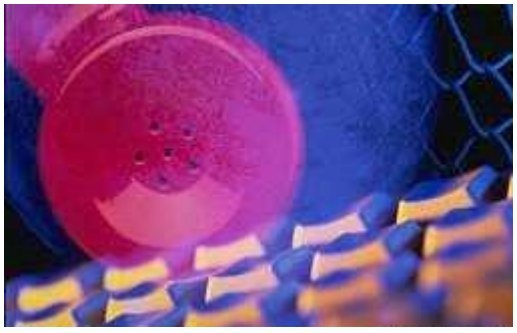
Si esta base de datos parece una idea tan buena que le gustaria usarla en verificacion normal de entrada (login) por serie tambien, se desilusionara al saber que esto no es posible por el momento de manera sencilla. Para empezar, necesita Taylor **UUCP 1.05** para hacer esto, porque permite a getty que pase el nombre del usuario que llama al programa uucico usando la opcion -u. Luego tiene que engañar al programa *getty* que este usando para que llame a *uucico* en vez del usual *login*. Con *getty ps*, esto se puede hacer poniendo la opcion *LOGIN* en el fichero de configuracion. Sin embargo, esto desactiva los logins interactivos por completo. *mgetty*, por otro lado, tiene una caracteristica atractiva que le permite invocar diferentes comandos de entrada (login) segun 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 mayuscula, pero dejar que todos los demas usen el comando estandar *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 seccion siguiente.

12.5.3 Proteccion 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 despues de entrar, pero el servidor no tiene manera de comprobarlo. Por consiguiente, un atacante podria 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** anonimo, que tiene una clave publica.

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 podria ser asi:



```
system                  pablo
... opciones usuales    ...
called-login            Upablo
```

La ventaja de este metodo es que cuando un sistema entra y pretende ser **pablo**, el programa *uucico* comprobara que haya entrado como usuario **Upablo**. Si no es asi, el sistema que nos llama sera desconectado. Deberia 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 llamaran a su sistema o no. Para aquellos sistemas que nunca le llamaran, usted puede indicar en *called-login* un mombre ficticio, como **nuncallama**.

12.5.4 Vuelvase Loco - Comprobacion de Secuencia de Llamadas

Otra manera de defenderse de impostores es usando la comprobacion de secuencia de lla- madas. La comprobacion 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 comprobacion de secuencia de llamadas, ambas maquinas mantienen una cuenta del numero de conexiones establecidas hasta el momento. Se incrementa con cada conexion. Despues de entrar, el llamador envia su numero de secuencia de llamadas y el sistema llamado lo comprueba con su propio numero. Si no son iguales, el intento de conexion es rechazado. Si el numero inicial se elige aleatoriamente, los atacantes lo tendran mas dificil para adivinar el numero de secuencia de llamadas correcto.

Pero la comprobacion de la secuencia de llamada sirve para mas que esto: aunque una persona muy inteligente descubriese su numero de secuencia de llamada asi como su clave, usted sabra 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 numero 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 rechazara, porque los numeros de secuencia ya no son iguales.

Si usted activa la comprobacion de secuencia de llamadas, deberia comprobar los fiche-ros historicos regularmente para buscar mensajes de error que puedan significar posibles ataques. Si su sistema rechaza el numero de secuencia de llamada que el sistema remoto le ofrece, *uucico* pondra un mensaje en el fichero historico que dira algo como “Out of sequen- ce call rejected” (“Llamada fuera de secuencia rechazada”). Si su sistema es rechazado por el proveedor de correo porque los numero de secuencia no estan sincronizados, pondra un mensaje en el fichero historico que dice “Handshake failed (RBADSEQ)” (“Negociacion fallida (RBADSEQ)”).

Para activar la comprobacion del numero 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 numero de secuencia. Taylor **UUCP** mantiene el numero 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 solo por **uucp**). Lo mejor es inicializar este fichero con un valor arbitrario que ambas partes hayan acordado. De otro modo el atacante podria apañarselas 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 tambien tiene que activar la comprobacion del numero de secuencia, y empezar usando el mismo numero que usted.

12.5.5 UUCP Anonimo

Si quiere ofrecer aceso anonimo de **UUCP** a su sistema, primero tiene que establecer una cuenta especial como se describio anteriormente. Es practica comun darle a esta cuenta el nombre y la clave **uucp**.

Ademas, tiene que especificar algunas opciones de seguridad para sistemas desconocidos. Por ejemplo, usted podria querer prohibirles que ejecuten comandos en su sistema. Sin embargo, estos parametros 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 linea hace que *uucico* ignore cualquier peticion del sistema remoto de activar la comprobacion de errores (debugging) localmente. Las dosultimas lineas permiten que los sistemas desconocidos ejecuten *rmail*; pero el camino de busqueda de comandos especificado hace que *uucico* busque el comando *rmai*l solamente en un directorio privado llamado *anon-bin*. Esto le permite a usted suminitrar algun programa *rmail* especial que, por ejemplo, reenvie todo el correo al superusuario para examinarlo. Esto permite a los usuarios anonimos contactar con el administrador del sistema, pero al mismo tiempo evita que ellos manden correo a otros sistemas.

Para activar UUCP anonimo, tiene que especificar por lo menos un comando *unknown* en el fichero *config*. Si no, *uucico* rechazara cualquier sistema desconocido.

12.6 Protocolos de bajo nivel de UUCP

Para negociar el control de la sesion y las transferencias de ficheros con el sistema remoto, *uucico* usa un grupo de mensajes estandar. Esto es lo que se llama normalmente *protocolo de alto nivel*. Durante la fase de inicializacion y la fase de desconexionestos se envian simplemente como tiras de caracteres. Sin embargo, durante la fase de transferencia, se usa tambien un *protocolo de bajo nivel*, que resulta transparente para los niveles superiores. De esta manera es posible comprobar errores cuando se usan lineas poco fiables, por ejemplo.

12.6.1 Resumen del protocolo

Dado que **UUCP** se usa sobre diferentes tipos de conexiones, como lineas serie, **TCP**, o incluso **X.25**, es preciso usar protocolos de bajo nivel especificos. Ademas, varias implementaciones de **UUCP** han introducido diferentes protocolos para hacer lo mismo.

Los protocolos se pueden dividir en dos categorias: de corriente o flujo (streaming) y por paquetes. La primera clase de protocolos transfiere un fichero entero, posiblemente calculando una suma de comprobacion (checksum). Esto apenas supone un gasto extra de tiempo, pero precisa una conexion fiable, porque cualquier error causaria que todo el fichero tenga que volver a ser enviado. Estos protocolos se suelen usar sobre conexiones de **TCP**, pero no sobre lineas telefonicas. Aunque los modems modernos hacen un buen trabajo corrigiendo errores, no son perfectos, y tampoco lo es la deteccion de errores entre el ordenador y el modem.

Por su lado, los protocolos por paquetes parten el fichero en varias partes de igual tamaño. Cada paquete se envia y recibe por separado, se realiza una suma de comprobacion, y se devuelve al origen un paquete de confirmacion. Para que sea mas eficiente, se inventaron protocolos de ventanas deslizantes, que permiten un numero limitado (una ventana) de paquetes sin esperar confirmacion en un momento dado. Esto reduce considerablemente la cantidad de tiempo que *uucico* tiene que esperar durante una transmision. Aun asi, todos los calculos extra necesarios en comparacion a un protocolo de flujo hace que los protocolos de paquetes sean ineficientes sobre **TCP**.

El ancho de los datos tambien supone una diferencia. A veces, el envio de caracteres de ocho bits sobre una conexion serie es imposible, por ejemplo si la conexion atraviesa un estupido servidor de terminales. En este caso, los caracteres con el octavo bit igual a uno tienen que ser especialmente tratados. Cuando se envian caracteres de ocho bits sobre una conexion de siete bits, tienen que estar bajo la suposicion del peor caso posible. Esto duplica la cantidad de datos a transmitir, aunque la compresion que se hace por hardware puede

compensar esto. Las lineas que pueden transmitir caracteres de ocho bits se llaman preparadas para ocho bits. Este es el caso de todas las conexiones **TCP**, asi como la mayoria de los modems.

Existen los siguientes protocolos con Taylor **UUCP 1.04**:

g Este es el protocolo mas comun y deberia ser entendido por practicamente todos los *uucico's*. Hace comprobacion de errores en profundidad y es, por lo tanto, apropiado para las ruidosas conexiones telefonicas. *g* require una conexion preparada para ocho bits. Es un protocolo orientado a paquetes que usa una tecnica de ventana deslizante.

i Este es un protocolo bidireccional de paquetes que puede enviar y recibir ficheros al mismo tiempo. Requiere una conexion que permita comunicación bidireccional simultanea (full-duplex) y preparada para ocho bits. Actual- mente solo es usado por Taylor **UUCP**.

t Este es un protocolo diseñado para usarse sobre una conexion de **TCP**, u otras redes libres de errores. Usa paquetes de 1024 bytes y requiere una conexion de ocho bits.

e Este protocolo basicamente hace lo mismo que *t*. La principal diferencia es que e es un protocolo de flujo.

f Este protocolo esta pensado para usarse sobre conexiones fiables X.25. Es un protocolo de flujo y espera una conexion de siete bits. Los caracteres de ocho bits son codificados, lo cual lo hace muy ineficiente.

G Esta es la version del *Unix* SVR4 del protocolo *g*. Tambien se utiliza en algunas otras versiones de **UUCP**.

a Este protocolo es similar a **ZMODEM**. Requiere una conexion de ocho bits, pero codifica ciertos caracteres como **XON** y **XOFF**.

12.6.2 Ajuste del protocolo de transmision

Todos los protocolos permiten alguna variacion en el tamaño de los paquetes, el cronometro y similares. Usualmente, los valores por defecto funcionan bien, pero puede no seroptimo para su configuracion. El protocolo *g*, por ejemplo, usa tamaños de ventanas de 1 a 7, y tamaños de paquetes en potencias de 2 desde 64 a 4096. Si su linea telefonica es tan ruidosa que ignora el 5 por ciento de los paquetes, probablemente deberia disminuir el tamaño de los paquetes y de la ventana. Sin embargo, en lineas de telefono muy buenas el hecho de enviar acuses de recibo por cada 128 bytes puede resultar un desperdicio, asi que podria 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 parametros 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 parametros ajustables y sus nombres varian entre protocolos. Para ver una lista completa deestos puede consultar la documentacion que acompaña al codigo fuente de Taylor **UUCP**.

12.6.3 Seleccion de protocolos especificos

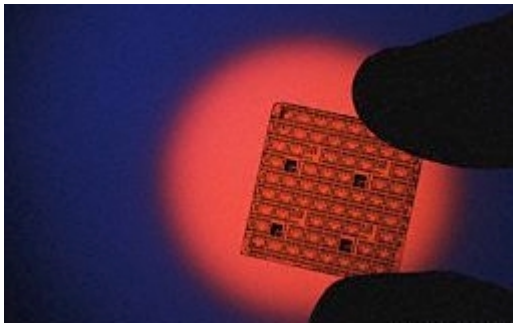
No todas las implementaciones de *uucico* hablan y entienden cada protocolo, de modo que durante la fase de negociacion de protocolos, ambos procesos tienen que ponerse de acuerdo en uno comum. El *uucico* maestro ofrece al esclavo una lista de protocolos soportados enviando **P*protlist*** , de la cual el esclavo elige uno.

Segun el tipo de puerto usado (modem, **TCP** o directo), *uucico* crea una lista por defecto de protocolos. Para modem 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 podria modificar la entrada del fichero port para su puerto de modem de esta manera:

```
port            serial1
...
protocols       igG
```

Esto requiere que cualquier conexion entrante o saliente en este puerto use *i, g* o *G*. Si el sistema remoto no soporta ninguno deestos, la negociacion fallara.

12.7 Solucion de problemas



Esta seccion describe lo que puede ir mal con su conexion de **UUCP** y sugiere donde buscar el error. Sin embargo, solo he puesto las preguntas que se me han ocurrido, por lo que pueden surgir otras muchas cosas que no diga aqui.

En cualquier caso, active la opcion de encontrar errores con **-xall**, y observe el resultado en el fichero *Debug* del directorio de cola. Esto deberia

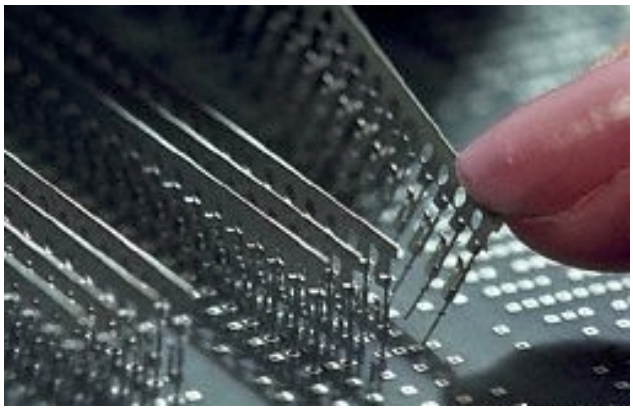
ayudarle rapidamente a reconocer donde reside el problema. Siempre me ha servido de ayuda el activar el altavoz del modem cuando no se conecta. Con un modem compatible Hayes, esto se consigue añadiendo "**ATL1M1 OK**" en el dialogo del modem en el fichero *dial*.

La primera cosa a comprobar siempre deberia ser si todos los permisos de los ficheros estan ajustados correctamente. *uucico* debe tener identificacion de usuario **uucp**, y todos los ficheros en */usr/lib/uucp*, */var/spool/uucp* y */var/spool/uucppublic* tienen que pertenecer a **uucp**. Tambien hay algunos ficheros ocultos 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 especifico el comando *time* que determina a que horas se puede llamar al sistema remoto, o bien especifico unas horas que en realidad prohiben llamar en este momento. Si no se especifica cuando se puede llamar, *uucico* asume que no se puede llamar nunca a ese sistema.

***uucico* se queja de que el sistema ya esta en uso** Esto significa que *uucico* detecto 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, tambien es posible que haya otro *uucico* ejecutandose en el sistema que este intentando llamar al sistema remoto y se atasco en una macro de dialogo, etc. Si este *uucico* no consigue conectarse al sistema remoto, matelo 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 dialogo falla Mire el texto que recibe del sistema remoto. Si esta salteado, esto puede ser un problema relacionado con la velocidad. Si no, confirme que realmente envia lo que su macro de dialogo espera recibir. Recuerde, la macro de dialogo empieza con una cadena de caracteres esperada. Si usted recibe la invitacion de entrada al sistema (login), despues envia 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 rapido para su modem.



Mi modem no marca: Si su modem no indica que la linea **DTR** ha sido elevada cuando *uucico* hace una llamada, posiblemente no le ha dado el dispositivo correcto a *uucico*. Si su modem 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 dialogo del modem. Si esto no produce un eco de sus comandos durante el dialogo del modem, compruebe si la velocidad de la linea es demasiado alta o demasiado baja para su modem. Si si que ve el eco, compruebe si ha desactivado las respuestas del modem, o las ha

configurado como codigos numericos. Verifique que la macro de dialogo en si misma es valida. Recuerde que tiene que poner dos barras invertidas para enviar una al modem.

Mi modem intenta marcar, pero la llamada no sale Inserte un retraso en el numero de telefono. Esto es especialmenteutil cuando se llama fuera de la red interna de una compañia. Para la gente en Europa, que normalmente marca con pulsos (pulse-tone), pruebe con tonos (touch-tone). En ciertos paises, los servicios telefonicos 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 cone- xion entre su ordenador y su modem 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 limite es 9600. Tambien tiene que asegurarse de que la negociacion hardware esta incluida en la linea serie.

Otra posible causa es que la negociacion hardware no este activada en el puerto. Taylor **UUCP 1.04** no tiene mecanismos para activar la negociacion de **RTS/CTS**. Tiene que activarla explicitamente en el fichero *rc.serial* usando el siguiente comando:

```
$ stty crtscts < /dev/cua3
```

Puedo entrar en el otro sistema, pero la negociacion falla Bien, puede ser debido a muchas causas. Los mensajes en el fichero de registro deberian decirle un monton de cosas. Mire que protocolos ofrece el sistema remoto (envia un **P*protlist*** durante la negociacion). A lo mejor no tienen nada en comum (>seleciono algun protocolo en *sys* o *port* ?).

Si el sistema remoto envia **RLCK**, hay un fichero de bloqueo (lock) para su sistema en el sistema remoto. Si no es porque usted ya esta conectado al sistema remoto en otra linea, pida que lo borren.

Si envia **RBADSEQ**, el otro sistema tiene la comprobacion de la cuenta de conversacion activada para su sistema, pero los numeros no se corresponden. Si envia **RLOGIN**, no le fue permitido entrar con ese nombre de usuario.

12.8 Archivos de registro historico (Log Files)

Cuando se compila el paquete de **UUCP** para usar ficheros de registro al estilo Taylor-**UUCP**, se tendran tres ficheros historicos globales, y todos residiran en el directorio de cola. El fichero principal se llama Log y contiene toda la informacion sobre las conexiones establecidas y los ficheros transferidos. Un extracto tipico podria ser como el siguiente (despues de formatearlo para que quepa en la pagina):

```
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 estadisticas de transferencias de ficheros. La seccion de *Stats* que corresponde a la transferencia anterior se muestra aqui:

```
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 lineas han sido partidas para que quepan en la pagina.

El tercer fichero es *Debug*. Este es el sitio donde se incluye toda la informacion para buscar errores. Si usted usa deteccion de errores (debugging), tiene que asegurarse de que este fichero tenga modo de proteccion de 600. Dependiendo del modo de busqueda 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 historico de **HDB**. **HDB UUCP** usa muchos ficheros de registro archivados bajo */var/spool/uucp/.Log*. Este directorio contiene tres directorios mas, llamados *uucico*, *uuxqt* y *uux*. Estos contienen el resultado de informacion historica 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** acabara en *.Log/uucico/pablo*, mientras que el *uuxqt* correspondiente escribira en *.Log/uuxqt/pablo*. Las lineas escritas a cada uno de estos ficheros son sin embargo iguales que en Taylor **UUCP**.

Cuando active la opcion de busqueda de errores con el estilo **HDB**, la informacion sera escrita en el directorio *.Admin* bajo */var/spool/uucp*. Durante llamadas salientes, la informacion se envia al fichero *.Admin/audit.local*, mientras que la salida de *uucico* cuando alguien nos llama se graba en *.Admin/audit*.

Capítulo 13 Correo Electónico

Uno de los usos mas comunes de las redes informaticas desde sus origenes ha sido el correo electronico. Empezo siendo un simple servicio que copiaba un fichero de una maquina a otra, y lo añadia al fichero *mailbox* (buzon de correo) del destinatario. Basicamente, en esto sigue consistiendo el *e-mail* (correo electronico), aunque el crecimiento continuo de la red y, consiguientemente, el aumento de la complejidad de encaminado, ha hecho necesario un esquema mas elaborado.

Se han diseñado varios estandares de intercambio de correo. Los nodos conectados a la Internet cumplen uno recogido en el **RFC 822**, complementado en algunos **RFCs** que describen un metodo independiente de la maquina para transferir caracteres especiales, y similares. Mucho se ha discutido recientemente sobre el "correo multi-media", que tiene que ver con incluir imagenes y sonido en los mensajes de correo. Otro estandar, X.400, ha sido definido por el **CCITT**.

Hay ya una gran cantidad de programas de transporte de correo para sistemas UNIX. Uno de los mas 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 describira en el capitulo 15. La version de *sendmail* actualmente en desarrollo es la 8.6.5.

El agente de correo de uso mas comun 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 mayoria 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 aqui.

Comparado con *sendmail*, *smail* es bastante joven aun. 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 metodo de configuracion es mucho mas flexible.

Ambos *smail* y *sendmail* admiten un conjunto de ficheros de configuracion que deben ser adaptados a cada caso particular. Aparte de la informacion que se necesita para hacer funcionar el subsistema de correo (como puede ser el nombre del ordenador local), hay muchos mas parametros que pueden ajustarse. El fichero principal de configuracion de sendmail es muy dificil de entender a la primera. Parece como si el gato se hubiese echado una siesta sobre el teclado con la tecla de mayusculas pulsada. Los ficheros de configuracion de smail estan mas estructurados y son mas faciles 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 practicamente el mismo.

En este capitulo trataremos sobre que es el 'e-mail' y que temas tendra que abordar usted como administrador del sistema. Los capitulos 14 y 15 daran instrucciones para poner a punto *smail* y *sendmail* por primera vez. La informacion que se suministra debe bastar para poner en marcha pequeños nodos, pero hay muchas mas opciones y usted podra pasar muchas horas felices frente a su ordenador configurando las caracteristicas mas superficiales.

Hacia el final de este capitulo nos ocuparemos brevemente de como poner a punto *elm*, un programa para usuario de correo muy comun en muchos sistemas **UNIX**, incluyendo Linux.

Para mas informacion sobre temas especificos de correo electronico 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 *send- mail* contienen tambien una docu mentacion muy extensa que debe solucionar la mayoria de sus dudas sobre instalacion y puesta a punto. Si busca informacion sobre correo electronico en general, hay varios RFCs que tratan especificamente este tema. Una lista de ellos se encuentra en la bibliografia al final del libro.

13.1 ¿Que 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 categorias; en la primera categoria estan los datos que son especificos 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 manipulacion del mensaje, que no es propio de ningun mecanismo de transporte, como es la linea del encabezado en la que indicamos el tema del mensaje (*Subject*), la lista de todos los destinatarios, y la fecha en la que se envio el mensaje. En muchas redes, se ha convertido en un estandar 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 linea en blanco.

La mayoria del software para transporte de correo que se usa en el mundo UNIX usa un formato de encabezado definido en el **RFC 822**. Su proposito original era especificar un estandar para usar en la **ARPANET**, pero dado que fue diseñado para ser independiente del entorno de uso, ha sido facilmente adaptado a otras redes, incluyendo muchas basadas en **UUCP**.

Pero **RFC 822** es solo el maximo denominador comun. Otros estandares mas recien- tes han sido concebidos para dar respuesta a las crecientes necesidades como pueden ser, por ejemplo, encriptacion de datos, soporte de conjuntos de caracteres internacionales, y extensiones de correo multimedia (multi-media mail extension, **MIME**).

En todos esos estandares, el encabezado consiste en varias lineas, separadas por caracteres de retorno de carro. Cada linea consiste en un nombre de campo, que comienza en la columna uno, y el campo en si, separados por dos puntos (:) o un espacio. El formato y la semantica de cada campo varia dependiendo del nombre del mismo. Un campo del encabezado se puede continuar mas alla de una linea, si la linea siguiente comienza con un caracter de tabulacion. Los campos pueden aparecer en cualquier orden.

Un encabezado de correo tipico puede ser algo asi:

```
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
```

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 continuacion:

From: Contiene la direccion de correo electronico del remitente, y posiblemente el "nombre real". Un verdadero zoologico de formatos distintos se usa aqui.

To: Esta es la direccion 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 direccion a la que el remitente desea que el destinatario le conteste. Esto puede serutil si se tienen varias direcciones, pero se desea recibir la mayor parte del correo solo en aquella que se usa mas a menudo. Este campo es opcional.

Organization:
La organizacion que posee la maquina desde la que se ha enviado el mensaje. Si la maquina 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. Esunica para cada mensaje.

Received: Cada nodo que procesa su correo (incluyendo las maquinas del remitente y el destinatario) insertan este campo en el encabezado, dando el nombre del nodo, una identificacion de mensaje, hora y fecha a la que lo recibieron, de que nodo procede, y que software de transporte ha sido usado. Esto se hace asi 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:
Ningun programa relacionado con el correo debe protestar sobre cualquier encabezado que comience con **X-**. Esto se usa para implementar caracteristicas adicionales que aun no han sido incluidas en un **RFC**, o que no lo seran 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 unica excepcion a esta estructura es la primera linea. 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 mas adelante), la hora y la fecha en que fue recibido por laultima maquina que lo ha procesado, y una parte opcional que especifica desde que maquina 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** continua 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 enel para marcar el comienzo de un mensaje en el buzon del usuario. Para evitar problemas potenciales con lineas del contenido del mensaje que comiencen tambien con “From“, se ha convertido en practica comun distinguir este ultimo caso precediendolo de un “>”.

13.2 ¿Como se reparte el correo?

Generalmente, usted escribira su correo usando un interface de correo como *mail* o *mailx*; u otros mas sofisticado s como *elm*, *mush*, o *pine*. Estos programas se denominan *agentes de usuario de correo (mail user agents*), o **MUAs** para abreviar. Si usted envia un mensaje de correo, el programa interface en la mayoria de los casos se lo pasara 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 envios locales o lejanos; en otros hay solo un **MTA**. El comando para envios lejanos se denomina usualmente rmail, el otro se denomina *lmail* (si existe).

Un envio local de correo es, por supuesto, algo mas que añadir el mensaje al buzon del destinatario. Usualmente el **MTA** local entendera como usar alias (definir direcciones locales de destinatarios que dirigen a otras direcciones) y como usar redirecciones, es decir, dirigir el correo de un usuario a otra direccion). Tambien, los mensajes que no pudieron ser enviados deben ser normalmente *devueltos* (*bounced*) al remitente junto con algun mensaje de error.

Para envios lejanos, el software de transporte usado depende del tipo de enlace. Si el correo debe enviarse a traves 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 maqui- na del destinatario directamente, negociando la transferencia del mensaje con el demonio **SMTP** del otro lado.

En redes tipo **UUCP**, el correo no suele ser enviado directamente, sino que es redirigido hasta su destino a traves de un conjunto de maquinas intermedias. Para enviar un mensaje a

traves de un enlace **UUCP**, el **MTA** remitente ejecutara usualmente *rmail* en la maquina intermedia usando uux, y suministrandole el mensaje en la entrada estandar.

Dado que esto se hace para cada mensaje por separado, puede producir una carga considerable de trabajo en un nodo procesador de correo grande, ademas de inundar las colas UUCP con cientos de pequeños mensajes que ocupan una cantidad de disco desproporcionada. 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 ejecutaria normalmente si usara una conexion **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 procesara la entrada como si una conexion **SMTP** normal hubiese ocurrido.

13.3 Direcciones de correo electrónico

Para el correo electronico, una direccion consiste en, al menos, el nombre de la maquina que maneja el correo del destinatario, y una identificacion de usuario reconocida por ese sistema. Puede ser el nombre de acceso del destinatario, pero puede ser tambien cualquier otra cosa. Otros esquemas de direcciones, como el X.400, usan un conjunto mas general de "atributos" que se utilizan para buscar la maquina del destinatario en un servidor de directorio X.500.

La forma en que se interpreta un nombre de maquina, es decir, a que nodo va a llegar finalmente nuestro mensaje, y como 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 estandar RFC 822, que requiere una notacion **usuario@maquina.dominio**, donde **maquina.dominio** es el nombre de dominio totalmente cualificado (Fully Qualified Domain Name, o **FQDN**) de la maquina. El signo de *arroba* que aparece entre medias se suele denominar signo "at". Dado que esta notacion no indica la ruta hasta la maquina de destino, sino que da el nombre (unico) de dicha maquina, a esto se le suele llamar una direccion *absoluta*.

En el entorno **UUCP** original, la forma predominante era **ruta!maquina!usuario**, donde **ruta** describia una secuencia de maquinas a traves de las cuales debia viajar el mensaje para llegar a **maquina**, su destino final. Esta notacion se llama la *ruta bang*, porque un sig -no de exclamacion se denomina coloquialmente "bang". Hoy en dia muchas redes basadas en **UUCP** han adoptado el **RFC 822**, y entenderan ese tipo de direccion.

Estos dos tipos de direcciones no se mezclan muy bien. Supongamos una direccion **maquinaA!usuario@maquinaB**. No queda claro si el signo '**@**' tiene precedencia sobre la ruta o viceversa: >Hemos de enviar el mensaje a **maquinaB**, que lo enviara a **maqui- naA!usuario**, o debe ser enviado **maquina**, que lo redirigira a **usuario@maquinaB**?

Las direcciones que mezclan diferentes tipos de operadores de direccion se denominan *direcciones hibridas*. El mas notorio es el ejemplo anterior. Se resuelve usualmente dandole precedencia al signo '**@**' sobre la ruta. En el ejemplo anterior, esto significa enviar el mensaje a **maquinaB** primero.

De todos modos, hay una forma de especificar rutas acorde con RFC 822: <**@maquinaA,@maquinaB:usuario@maquinaC**> denota la direccion de **usuario** en **maquinaC**, indicando que se debe llegar a **maquinaC** a traves de **maquinaA** y **maquinaB** (en ese orden). Este tipo de direccion se suele llamar una direccion route-addr (de route, ruta y address, direccion).

Y tambien existe el operador de direccion '**%**': **usuario%maquinaB@maquinaA** sera enviado primero a **maquinaA**, que sustituira el signo de tanto por ciento que se encuentre mas a la derecha en la expresion (en este caso elunico) por un signo '**@**'. La direccion quedara ahora **usuario@maquinaB**, y el gestor de correo redirigira alegremente el mensaje a la **maquinaB** que lo entregara a **usuario**. Este tipo de direccion se suele denominar a veces como "Ye Olde ARPANET Kludge" ("La Vieja Chapuza de ARPANET") y su uso esta desaconsejado. Aun asi muchos agentes de transporte de correo generan este tipo de direcciones.

Otras redes tienen mas 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 ***maquina::usuario***. Finalmente, el estandar X.400 usa un esquema totalmente distinto, describiendo a un destinatario por un conjunto de pares atributo-valor, como pais u organizacion.

En FidoNet, cada usuario se identifica por un codigo como **2:320/204.9**, que consiste en cuatro numeros que denotan la zona (2 es Europa), red (320 es Paris 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 escribiria: **Thomas.Quinot@p9.f204.n320.z2.fidonet.org** ¿No he dicho antes que los nombres de dominio son faciles de recordar?

Hay algunas implicaciones al usar esos tipos diferentes de direcciones que seran descritas a lo largo de las proximas secciones. De todos modos, en un entorno RFC 822 raramente.

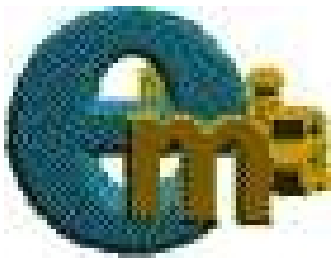
13.4 ¿Como funciona el encaminado del correo?

El proceso de dirigir un mensaje a la maquina del destinatario se denomina *encaminado*. Ademas de encontrar una ruta desde el nodo emisor al receptor, este proceso incluye tambien chequeo de errores asi como optimizacion 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 direccion **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 algun encaminado especifico de correo. El comportamiento por defecto consiste en enviar el mensaje al nodo de destino buscando su direccion **IP**, y dejando el encaminado de los datos en si a la capa **IP** de transporte.

Generalmente, la mayoria de los nodos querran que todo el correo entrante se dirija a un servidor de correo facilmente accesible que sea capaz de procesar todo ese trafico, y que distribuira 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 basicamente quiere decir que el servidor va a actuar como un redistribuidor de correo para todas las maquinas de este dominio. Los campos **MX** tambien pueden usarse para manipular el trafico dirigido a maquinas que no estan ellas mismas conectadas a la Internet, como redes **UUCP** o redes corporativas que contienen informacion confidencial.

Los campos **MX** tambien tienen una *preferencia* asociada. Es un entero positivo. Si existen varios intercambiadores de correo para una maquina, el agente de transporte de correo intentara enviar el mensaje al intercambiador con menor valor de preferencia, y solo sieste falla probara uno con mayor valor. Si el nodo local esel mismo un intercambiador de correo para la direccion de destino, no redirigira los mensajes a cualquier maquina MX que tenga un valor de preferencia mayor que el suyo propio: esta es una forma segura de evitar bucles de correo.

Supongamos que una organizacion, digamos la Sociedad **ACME**, quiere que todo su correo sea manipulado por su maquina llamada **mailhub**. Entonces tendran 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 maquina que desee enviar un mensaje a **joe@greenhouse.acme**.com buscara el registro DNS de **acme.com**, y encontrara el campo MX apuntando hacia **mailhub**. Si no hay ningun MX con un valor de preferencia menor que 5, el mensaje sera enviado a **mailhub**, que lo entregara a **greenhouse**.

Lo anterior es solo un esbozo de como funcionan los campos MX. Para mas informacion 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 mas complicado que en la Internet, porque el software de transporte no realiza ningun encaminado. Al principio, todo el correo tenia que ser dirigido usando rutas bang. Las rutas bang especifican, como hemos dicho ya, una lista de nodos a traves de los cuales enviar el mensaje, separados por signos de admiracion, y seguida del nombre del usuario. Para dirigir una carta a Juanita, usuaria de una maquina llamada **moria**, deberiamos usar la ruta **eel!swim!moria!juanita**. Esto enviaria el correo desde nuestro nodo a **eek**, desde alli a **swim** y finalmente a **moria**.

El inconveniente obvio de esta tecnica es que obliga a que recordemos mucho sobre la topologia de la red, enlaces rapidos, etc. Aun peor, cualquier cambio en la topologia 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 maquinas 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 ambig"uedades en nombres de nodos fue la fundacion del *Proyecto de Cartografia UUCP* (*The UUCP Mapping Project*). Se encuentra en la Universi- dad de Rutgers, y lleva el registro de todos los nombres de nodos oficiales **UUCP**, junto con informacion de sus vecinos **UUCP** y su localizacion geografica, asegurandose de que ningun nombre se repite. La informacion recogida por el Proyecto de Cartografia se publica como los *Mapas de Usenet*, que se distribuyen regularmente en Usenet. Una entrada tipica para un sistema en un mapa (despues de eliminar los comentarios) seria algo asi:

```
moria
        bert(DAILY/2),
        swim(WEEKLY)
```

Esta entrada dice que **moria** tiene un enlace con **bert**, al que llama dos veces al dia (DAILY=diariamente), y con **swim**, al que llama una vez a la semana (WEEKLY). Volveremos de nuevo al formato del fichero mapa mas adelante.

Usando la informacion sobre conectividad proporcionada en los mapas, se pueden gene-rar automaticamente las rutas completas de nuestra maquina a cualquier nodo de destino. Esta informacion se almacena usualmente en el fichero *paths*, tambien llamado *base de da-tos de alias de rutas* (*pathalias database*). Suponiendo que los mapas indican que se puede llegar a **bert** a traves de **ernie**, entonces una entrada de alias de ruta para **moria** generada a partir del fragmento de mapa anterior quedaria tal que asi:

```
moria           ernie!bert!moria!%s
```

Si ahora damos una direccion de destino **de juanita@moria.uucp**, nuestro MTA esco-gera la ruta mostrada anteriormente, y enviara el mensaje a **ernie** con una direccion 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 informacion proporcionada en ellos normalmente esta bastante distorsionada, y a veces es obsoleta. Asi, solo unos pocos nodos grandes usan los mapas mundiales completos de **UUCP** para construir su fichero *paths*. La mayoria de los nodos solo mantienen informacion de encaminado hacia los nodos en sus cercanias, y envian cualquier correo dirigido a nodos que no encuentran en sus bases de datos a un nodo mas inteligente, con una informacion de encaminado mas completa. Este esquema se denomina *encaminado por nodo inteligente* (*smart-host routing*). Las maquinas que tienen un solo enlace de correo **UUCP** (*llamadas nodos hoja*) no realizan ningun encaminado propio; confian 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 adopcion del sistema de nombres de dominio en ellas. Por supuesto, no se pueden hacer peticiones a un servidor de nombres usando **UUCP**. Aun asi, muchos nodos **UUCP** han formado pe queños 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 informacion de encaminado global se encuentra solo 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 que incluir informacion de encaminado hacia cada una de las maquinas **UUCP** en el mundo. Aparte de la informacion completa de encaminado para el dominio al que sirven, ahora solo necesitan tener en sus bases de datos rutas a dominios completos. Por ejemplo, la siguiente entrada alias de ruta encaminara todo el correo para nodos del dominio **sub.org** a **smurf** :

```
.sub.org        swim!smurf!%s
```

Cualquier correo dirigido a **claire@jones.sub.org** sera enviado a **swim** con una direccion en el sobre tal como **smurf !jones!claire**.

La organización jerarquica del espacio de nombres de dominio permite a los servidores de correo mezclar rutas mas especificas con otras menos especificas. Por ejemplo, un sistema en Francia puede tener rutas especificas hacia subdominios de **fr**, pero encaminar cualquier correo dirigido a maquinas del dominio **us** hacia algun sistema en los EE.UU. De esta forma, el encaminado basado en dominios (que es como se denomina esta tecnica) reduce enormemente el tamaño de las bases de datos de encaminado, asi como las tareas de administracion 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 facilmente usar pasarelas entre redes **UUCP** y la Internet. Hoy en dia, muchos dominios **UUCP** tienen un enlace con una pasarela a Internet que actua como su nodo inteligente. Enviar mensajes a traves de la Internet es mas rapido, y la informacion de encaminado es mucho mas 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** actua como su pasarela a Internet. Entonces **moria** usaria **gcc2** como su nodo inteligente, de modo que todo el correo hacia dominios remotos seria enviado a traves de la Internet. Por otro lado, **gcc2** anunciaria una entrada MX para **orcnet.org**, y enviaria todo el correo entrante para nodos de **orcnet** a **moria**.

Elunico problema que queda es que los programas de transporte **UUCP** no pueden manejar nombres de dominio totalmente cualificados. La mayoria de los paquetes integrados de **UUCP** fueron diseñados para tratar con nombres de nodos de ocho caracteres como maximo, algunos incluso menos, y usar caracteres no alfanumericos, como puntos, esta totalmente fuera de lugar para la mayoria.

Asi, 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 implementacion. Una manera comun de relacionar **FQDNs** con nombres **UUCP** es usar el fichero de alias de ruta para esto:

```
moria.orcnet.org   ernie!bert!moria!%s
```

Esto producira una ruta bang al estilo **UUCP** puro a partir de una direccion 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 transformacion inversa (llamada coloquialmente *dominizar*) se necesita a veces cuando se envia correo desde una red **UUCP** a la Internet. Mientras el remitente use el nombre de dominio totalmente cualificado en la direccion de destino, este problema se puede evitar no eliminando el nombre del dominio de la direccion del sobre cuando se redirige el mensaje al nodo inteligente. De todos modos, siguen existiendo algunos nodos **UUCP** que no

son parte de ningun dominio. Estos 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 informacion de encaminado principal en redes basadas en **UUCP**. Una entrada tipica sera 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 via **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 maquinas dentro de algun dominio a su concentrador de correo, se debe tambien 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 maquinas de **sub.org** pueden ser alcanzadas a traves 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 solo cuando se esta manejando un nodo que no necesita hacer demasiados encaminados. Si se tienen que hacer encaminados hacia un gran numero de maquinas, 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 mas facilidad, porque solo 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 Cartografia de Usenet ya no se usan demasiado para encaminar, las redes **UUCP** pequeñas pueden suministrar informacion 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 lineas, comenzando cada nueva linea con un tabulador. Cada enlace consiste en el nombre del nodo enlazado, seguido del coste, entre corchetes. El coste es una expresion aritmetica, consistente en numeros y costes simbolicos. Las lineas que comienzan con un signo hash (#) se ignoran.

Como ejemplo, consideremos **moria**, que registra a **swim.twobirds.com** dos veces al dia, y **a bert.sesame.com** una vez a la semana. Ademas el enlace con **bert** usa solo un modem lento de 2400bps. **moria** publicaria la siguiente entrada en los mapas:

```
moria.orcnet.org
        swim.twobirds.com(DAILY/2),
        bert.sesame.com(WEEKLY+LOW)

moria.orcnet.org = moria
```

La ultima linea lo haria ser conocido bajo su nombre **UUCP** tambien. Observese que debe ser *DAILY/2*, porque llamar dos veces al dia reduce a la mitad el coste de ese enlace.

Usando la informacion de dichos ficheros de mapa, *pathalias* puede calcular rutasoptimas para cualquier destino que aparezca en el fichero de rutas, y producir una base de datos de alias de ruta a partir deeste, que se puede usar para encaminar hacia esos nodos.

pathalias proporciona varias posibilidades mas, como esconder nodos (es decir, hacerlos accesibles solo a traves de una pasarela), etc. Vease la pagina del manual sobre *pathalias* para mas detalles, ademas de una lista completa de costes de enlace.

Los comentarios en el fichero de mapa contienen generalmente informacion adicional sobre los nodos descritos en el. Existe un formato rigido para especificaresta, 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 informacion de una manera elegante.

Cuando se registra un nodo con una organizacion que distribuye ficheros de mapa a sus miembros, generalmente se debe rellenar una de esas entradas de mapa.

A continuacion 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 detras de los primeros dos caracteres es un tabulador. El significado de la mayoria de los campos es bastante obvio; recibiremos una descripcion detallada del dominio con el que vayamos a registrarnos. El campo *L* es el mas divertido de averiguar: da nuestra posicion geografica en latitud/longitud y se usa para dibujar los mapas *postscript* que muestran todos los nodos de cada pais, asi como los del mundo entero.

13.6 Configuracion de elm

elm significa "electronic mail" (correo electronico) y es una de las utilidades UNIX mas razonablemente bautizadas. Proporciona una interface de pantalla completa con una buena utilidad de ayuda. No vamos a explicar aqui como se usa *elm*, solo nos detendremos en sus opciones de configuracion.

Teoricamente, se puede usar *elm* sin configurar, y todo funciona bien _ con suerte. Pero hay algunas opciones que deben definirse, aunque solo se necesitan en contadas ocasiones.

Cuando comienza, elm lee un conjunto de variables de configuracion desde el fichero *elm.rc* en */usr/lib/elm*. Entonces, intentara 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 menu de opciones de *elm*.

El conjunto de opciones para el fichero privado *elmrc* tambien esta disponible en el fichero global *elm.rc*. La mayoria de las definiciones en el fichero privado elmrc sustituiran 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 maquina. Por ejemplo, en la Cervecera Virtual, el fichero para **vlager** contendra 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 maquina local. Aunque esta informacion se usa raramente, debemos definir estas opciones. Observese

que estas opciones solo tienen efecto cuando se dan en el fichero global de configuracion; cuando se encuentran en nuestro *elmrc* privado, seran ignoradas.

13.6.2 Conjuntos de Caracteres Nacionales

Recientemente, han habido propuestas para corregir el estandar **RFC 822** para soportar varios tipos de mensajes, como texto simple, datos binarios, ficheros Postscript, etc. El conjunto de estandares y **RFCs** que cubren estos aspectos se suelen conocer como **MIME**, o Extensiones de Correo Internet Multiproposito (Multipurpose Internet Mail Extensions). Entre otras cosas, esto permite al destinatario saber si un conjunto de caracteres distinto del estandar **ASCII** ha sido usado al escribir el mensaje, por ejemplo usando los acentos o dieresis 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 estandar que cumple. Tambien se conoce como Latin-1. Cualquier mensaje que use caracteres de ese conjunto debe llevar la siguiente linea 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 como 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 mensaj e 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 usara *metamail* sino que mostrara el mensaje directamente. Esto se puede hacer definiendo la siguiente opcion en el *elm.rc* global:

```
displaycharset = iso-8859-1
```

Observese que se puede definir esta opcion incluso cuando nunca vayamos a enviar o recibir mensajes que realmente contengan caracteres distintos del **ASCII**. Esto es asi porque la gente que envia 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 solo **ASCII**.

De todos modos, definir esta opcion en *elm.rc* no es suficiente. El problema es que cuando muestra los mensajes con el paginador incorporado, *elm* llama a una funcion de

biblioteca por cada caracter para determinar si es mostrable o no. Por defecto, esta funcion solo reconoce caracteres **ASCII** como mostrables, y muestra todos los demas 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 caracteristicas esta 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 mas 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 envie 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 tambien en el fichero *elmrc* privado en lugar de en el global.

Capítulo 14 Cómo configurar y poner en marcha smail

Este capitulo es una breve introduccion a la forma de configurar *smail* y ademas dara una idea general de la funcionalidad que este programa provee. Aunque *smail* es muy similar en comportamiento a *sendmail*, sus archivos de configuracion son totalmente diferentes.

El archivo de configuracion principal es */usr/lib/smail /config*. Este archivo es el que se debe editar para ajustar los valores especificos al sistema que se esta configurando. Si unicamente es un ordenador terminal de **UUCP**, seran relativamente pocas las opciones a cambiar. Hay ademas otros archivos que configuran las opciones de encaminamiento y transporte que se pueden modificar; se hablara brevemente sobre la forma de hacerlo.

La forma de operacion normal de *smail* hace que procese y entregue todo el correo de entrada inmediatamente. Si se tiene un trafico 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 maquina, se invoca desde el archivo *rc.inet2*, y se coloca en segundo plano, desde donde espera que haya una conexion **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 trafico, pues *smail* no se lanza por separado para cada conexion 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 conexion 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 estandar de operacion que se habilitan cuando es invocado con un nombre especifico tal como *rmail* o *smtpd*. Es comun que estos nombres especificos sean enlaces simbolicos al binario de *smail*. Se veran mas deestos cuando se discutan algunas otras caracteristicas de *smail*.

Hay dos enlaces a *smail* que deben existir siempre: */usr/bin/rmail* y */usr/sbin/sendmail*.

Cuando se crea y se envia un mensaje de correo con un agente de usuario tal como *elm*, el mensaje se pasara a *rmail* para su entrega, con la lista de destinatarios dada en la linea de comandos. Lo mismo sucede con el correo que entra via **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* esta en */usr/local/bin*, se debe escribir lo siguiente en la linea de comandos:

```
# ln -s /usr/local/bin/smail /usr/bin/rmail
# ln -s /usr/local/bin/smail /usr/sbin/sendmail
```

Si se quiere investigar mas sobre los detalles de configuracion de *smail*, se debe buscar en las paginas del manual *smail*(1) y *smail*(5). Si no estuviesen incluidas en su distribucion preferida del Linux, se pueden obtener junto con el codigo fuente de *smail*.

14.1 Configuracion de UUCP

Para usar *smail* en un entorno que solo tiene **UUCP**, la instalacion basica es muy sencilla. Primero se debe asegurar de que esten los dos enlaces simbolicos a *rmail* y *sendmail* men- cionados anteriormente. Si se espera recibir conexiones de **SMTP** de otros sitios, tambien se debe hacer un enlace de *rsmtp* a *smail*.

En la distribucion de *smail* de Vince Skahan, se encuentra un archivo muestra de con- figuracion. Su nombre es *config.sample* y esta en */usr/lib/smail*. Se debe copiar a *config* y editarlo para ajustar los valores especificos de su sistema.

Suponiendo que su maquina se llama *swim.twobirds.com*, y esta registrado en los mapas **UUCP** como *swim* y su relevo **UUCP** es *ulysses*, entonces el archivo *config* podria ser como el siguiente:

```
#
# Los nombres de nuestros dominios
visible_domain=two.birds:uucp
#
# 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 instruccion le indica a *smail* los dominios a los que su sistema pertenece. Se deben insertar sus nombres aqui, separados con signos de punto y coma. Si el nombre de su sistema esta registrado en los mapas de **UUCP**, sera necesario agregar ademas 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 direccion del destinatario con respecto al nombre del nodo, revisando cada uno de los nombres de la lista. Si la direccion coincide con cualquiera de estos nombres, o el nombre del sistema no esta 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 maquina que se desea utilizar para los mensajes que se envian hacia afuera. Este nombre se usa cuando se genera la direccion de quien envia 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 rebotaran hacia afuera del nodo local.

La ultima instruccion pone la ruta utilizada para el encaminamiento del relevo **UUCP** (descrito en la seccion 13.4). Con este cambio mostrado, *smail* enviara cualquier correo dirigido hacia direcciones remotas al relevo. Como los mensajes seran entregados a traves de **UUCP**, el atributo debe especificar un sistema conocido para los programas **UUCP** que corran en su sistema. Consulte el capitulo 12 sobre el tema de como hacer que su nodo sea conocido por **UUCP**.

Hay una opcion que se utiliza en el archivo anterior que aun no ha sido explicada; esta es *uucp name*. La razon para utilizar esta opcion 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 esta registrado en el mapa de **UUCP**, es necesario indicar a *smail* que en vez deeste utilice el nombre de dominio completamente calificado. Esto se puede hacer agregando la opcion *uucp name* al archivo de configuracion *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 mas 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 describira mas adelante, en este mismo capitulo.

14.2 Configuración para una red local

Si esta funcionando una instalacion con dos o mas nodos conectados por medio de una red local, es necesario designar a uno de ellos para que maneje la conexion **UUCP** con el mundo exterior. Entre las maquinas 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 demas maquinas. Esto permite a los usuarios desplazarse de maquina en maquina sin tener que mover su correo por todos lados (o peor, revisar tres o cuatro ordenadores para ver su correo recien recibido cada mañana). Asi mismo, es deseable hacer que las direcciones de los usuarios sean independientes del ordenador en la cual el correo se almacena. Es una practica comun utilizar el nombre del dominio como la direccion de quien envia el correo en vez de utilizar el nombre de la maquina servidora del correo. El usuario Janet, por ejemplo, podria especificar su direccion como **janet@vbrew.com** en vez de **janet@vale.vbrew.com**. A continuacion se explicara como hacer que el servidor reconozca el nombre del dominio como un nombre valido para su instalacion.

Otra forma de mantener todos los apartados postales en un anfitrion central es utilizar **POP** o **IMAP**. **POP** quiere decir, por sus siglas en ingles *Post Office Protocol*, es decir, *Protocolo de Oficina Postal* y permite a los usuarios tener acceso a sus archivos de correo a traves de una conexion **TCP/IP**. **IMAP**, o *Protocolo de Acceso Interactivo al Correo* por sus siglas en ingles de *Interactive Mail Access Protocol*, es similar a **POP**, excepto que es mas general. Ambos clientes y servidores para **IMAP** y **POP** han sido portados a Linux, y estan disponibles en **sunsite.unc.edu** bajo el directorio */pub/Linux/system/Network*.

14.2.1 Como escribir los archivos de configuracion

La configuracion para la Cervecera funciona de la siguiente forma: todos los nodos, con excepcion del servidor de correo **vstout**, encaminan todo el correo que va hacia el exterior hacia este servidor, utilizando la tecnica 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, envia todo el correo de la Cervecera; esteultimo nodo se llama **moria**.

El archivo estandar *config* para todas las maquinas con la excepcion 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 solo 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 configuracion, *config*, utiliza un esquema diferente para indicar a *smail* como 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 explicitamente. La lista de arriba contiene tanto el dominio completamente calificado como el del sistema no calificado, y el nombre del dominio completo en si mismo. Esto hace que *smail* reconozca a **janet@vbrew.com** como una direccion 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 direccion hacia **host .vbrew.com** en donde ***host*** no corresponde a ninguna maquina existente, se rechaza el mensaje y se devuelve al remitente del mismo. Si esta linea no esta, cualquier mensaje rechazado sera enviado nuevamente al relevo de correo, quien lo mandara a **vstout**, y asi sucesivamente hasta que se descarte por exceder la cuenta maxima de saltos.

14.2.2 Cómo ejecutar *smail*

La primera cosa que se debe hacer es decidir si se ejecutara *smail* como un demonio inde- pendiente, o si se permitira que *inetd* administre el puerto **SMTP** e invoque a *smail* cuando un cliente solicite una conexion **SMTP**. Normalmente es preferible la operacion 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 conexion indivi- dual. Cuando un servidor de correo entrega casi todo el correo que recibe directamente a los usuarios, es preferible optar por la operacion con *inetd*.

Independientemente del modo de operacion que se haya elegido para cada anfitrion in-dividual, es importante asegurarse que se tiene la siguiente linea en el archivo */etc/services*:

smtp 25/tcp # Simple Mail Transfer Protocol

Esto define el numero del puerto **TCP** que smail utilizara para las conexiones **SMTP. 25** es el puerto estandar definido por el **RFC** de Numeros de Puerto Asignados.

Cuando se ejecuta como demonio, *smail* se coloca a si mismo en segundo plano, y esperara a que ocurra una conexion en el puerto **SMTP**. Cuando haya una conexion, lanza un proceso y conduce una conversacion **SMTP** en dicho puerto. El demonio *smail* se lanza normalmente invocandolo desde el *script rc.inet2* con la siguiente instruccion:

/usr/local/bin/smail -bd -q15m

El modificador **-bd** indica que se funcionara 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* debera contener una linea como la siguiente:

```
smtp    stream  tcp  nowait  root  /usr/sbin/smtpd  smtpd
```

smtpd debe ser un enlace simbolico al binario de *smail*. Recuerde que tiene que forzar a que *inetd* relea *inetd.conf* enviandole una señal **HUP** despues de hacer estos cambios.

El modo demonio y el modo *inetd* son mutuamente excluyentes. Si se ejecuta *smail* como demonio, asegurese de que este comentada cualquier linea en *inetd.conf* para el servicio *smtp*. De manera similar, cuando se tenga a *inetd* como administrador de *smail*, asegurese de que *rc.inet2* no lanza al demonio *smail*.

14.3 Si no logra pasar. . .

Si algo va mal con la instalacion, hay algunas herramientas que pueden ayudar a encontrar cual es la raiz del problema. El primer lugar que se debe revisar es el conjundo de archivos de registro de *smail*. Estan en */var/spool/smail /log*, y se llaman *logfile* y *paniclog*, respecti- vamente. El primero lista todas las transacciones, mientras que elultimo solo se usa cuando haya mensajes de error relacionados con errores en la configuracion y similares.

Un ejemplo tipico de una linea 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 correcta- mente entregado al sistema **vstout** a traves de **SMTP**.

Los mensajes que *smail* no pudo entregar generan una linea 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 tipico para una situacion en la cual *smail* reconoce correctamente que el mensaje deberia ser entregado a **vstout** pero que no fue posible establecer la conexion al servicio **SMTP** en **vstout**. Si esto sucede, es posible que tenga un problema de configuracion o bien que el soporte **TCP** este 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 maquina. Una conexion exitosa al servidor **SMTP** se muestra a continuacion (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 linea que comienza con el codigo 220), debe asegurarse de que su configuracion es *verdaderamente* correcta antes de recompilar *smail*, como se describira a continuacion.

Si hay algun problema con *smail* que no se pueda localizar con el mensaje de error que *smail* genera, se pueden activar los mensajes de depuracion. Para hacer esto, se debe utilizar el modificador **-d**, seguido de un numero opcional que especifique el nivel de detalle de la informacion (no se debe dejar ningun espacio entre el modificador y el argumento numerico). Entonces, *smail* mostrara un informe de su operacion en la pantalla que dara mas pistas acerca de lo que puede estar mal.

14.3.1 Como compilar smail

Si esta seguro de que su *smail* carece de soporte de red **TCP**, es necesario obtener el codigo fuente. Es posible que ya este incluido en su distribucion si la obtuvo en **CD-ROM**, si no la mejor forma de compilar *smail*, es comenzar con el conjunto de archivos de confi- guracion de la distribucion de Vince Skahan. Para incluir el controlador de **TCP** dentro de la compilacion, se debe poner la macro ***DRIVER CONFIGURATION*** en el archivo *conf/**EDITME*** con el parametro *bsd-network* o *arpa-network*. El primero se utiliza para las instalaciones de red local, pero cuando se esta 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 seleccionar 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* est a 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

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 solo sé 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 mostrar_ a 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 este

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 booleana se pone, cualquier error generar_ a un mensaje al administrador de correo. Normalmente esto solo 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 valida, entonces esta se utiliza como ultimo 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 modulo de transporte.

El modulo 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

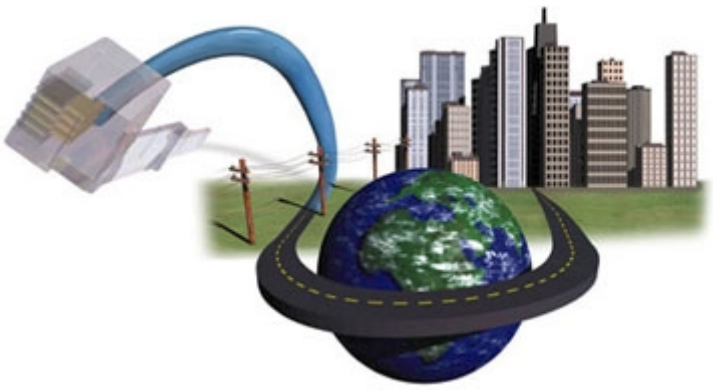
entrega local. Normalmente, la acción a realizar será entregar a un archivo de correo, pero los mensajes también pueden ser pasados a la entrada de un comando (por ejemplo, un filtro de correo que el usuario quiera establecer) o agregados a un archivo arbitrario cualquiera.

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 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*, 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.

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 que maquina 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 especifica 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 especifico, es elegido sobre el primero. Si hubiese dos encaminadores que proveen una solución correcta e igual de especifica, se elige al primero que esté en el archivo *routers*.

A continuación, el encaminador elegido especifica que 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 esta 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 encontrar_ a 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. Estos 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 v 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 deberá~ nadir 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). Estas 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 unix 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 simb_olico 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 ser_ a 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 él archivo especificado, y sustituir_ a 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 entregar_ a ning_ un 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 generar_ a 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 seguridad6. 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.

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 deber_ a 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 ser_ a 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* deber_ a ser un enlace al programa *smail*.

Cuando se maneja un mensaje con el transporte **UUCP**, *smail* convierte la dirección destino una trayectoria **UUCP** con símbolos de admiración. Por ejemplo, **user@host** se transformará **enhost!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 env__o 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. Es capaz de entregar un mensaje a cualquier número de direcciones de una maquina, 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 *smpt* como está listado en */etc/services*. Si no puede ser alcanzado, o expira el tiempo máximo de espera, la entrega del correo se reintentar_ a 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. *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 sería 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 deben ser mapeados al dominio **uccp** 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 carácter 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 especifica, 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 serán extremadamente difíciles, son ahora simples de crear y mantener.

Cuando se escribió este manual, la versión actual de *sendmail5.67b*+IDA1.5 estaban 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 .*

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 **DEC**net.

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 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.

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 están
# 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 #---------------------------------------------------------
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 # 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 # incluye el codigo 'real' que hace que todo funcione
dnl # (provisto con el codigo fuente)
dnl #
include(Sendmail.mc)dnl # LINEA INDISPENSABLE !!!
dnl #------------ FIN DEL ARCHIVO EJEMPLO DE SENDMAIL.M4 -------
```

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* 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.
```

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 auto disciplinen (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.uucpvbrew.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 aceptar_ a el correo.

DEFAULT HOST especifica la dirección de sistema que aparecer_ a 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           # dirección genérica 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 deberán configurar el ***DEFAULT HOST*** y las macros ***PSEUDONYMS***, detal 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 especificas 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 vía 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* será la siguiente:

```
# (en mailertable)
#
# redirige todo el correo para el dominio cs.groucho.edu vía UUCP a ada
UUCP-A,ada .cs.groucho.edu
#
# redirige todo el correo para el dominio groucho.edu vía 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 tendrá 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 va formato Internet (**SMTP**) utilizando un servidor de **DNS**, o mediante un sistema de reenvío. El archivo *uccpxtable* 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 " 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 " 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 u*ccpxtable*.

Supongamos que usted proporciona servicio de reenvío de correo a un sistema llamado **sesame.com** en **DNS** y **sesame** en los mapas **UUCP**. Necesitará la siguiente línea en *uccpxtable* 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*

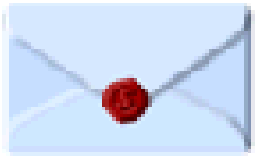
El archivo *pathtable* se utiliza para definir el encaminamiento explícito a sistemas o redes remotas. El archivo *pathtable* debe escribirse en orden alfabético con una sintaxis similar al estilo de *pathalias*. Los dos campos de cada línea deben estar separados por un **TAB** real; si no es así *dbm* 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 través 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 tabulación reales en cada línea o
# m4 podría 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 través 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 termino 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 # línea 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 *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 **DEC**net 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 distribución de Slackware reciente, probablemente baste con un tar -zxvf fichero.tgz y se obtendrán los mismos resultados.

Al desempaquetar el archivo se genera un directorio llamado *sendmail5.65b+IDA1.5+mailx5.3b*. En este directorio encontrar_ a 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 vía **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 gestionará 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 termino 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
>
```

Deberá 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, deber_ a 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 *uucptable* 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** deber_ a 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, también 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 módems 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 origino, 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, recuerdelo. . .) 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 caracteristicas, 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 ´*. 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* 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 deberá estar haciendo), y añadir una línea al fichero */etc/syslog.conf(5)* que tiene el siguiente aspecto:

```
mail.debug /var/log/syslog.mail
```

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 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 (*rmai l*), 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.

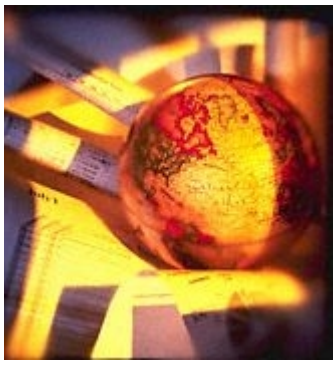
15.10 Dónde obtener más información

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 unix. 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 continuo 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 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.

Los artículos son enviados a uno o más grupos de noticias. Podría n 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 ujo de noticias se muestra en la figura 16.1.

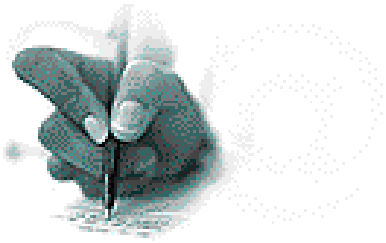
Las etiquetas en las echas que parten de **brewhq** pueden requerir ciertas explicaciones. Por defecto, **brewhq** quiere que todas las noticias generadas localmente sean enviadas a **groucho.edu**. Sin embargo, ya que **groucho.edu** no lleva los grupos **fj**, no hay razón para enviar ningún artículo 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. Por supuesto esto requiere mucho más que enredar con unos cuantos ficheros. Veamos cómo se las apañan la mayoría de sistemas unix 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 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 articulo 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*.

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 servidores conectados a Internet generalmente se basan en programas bajo **TCP/IP** que usan el protocolo **NNTP**, 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, 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.

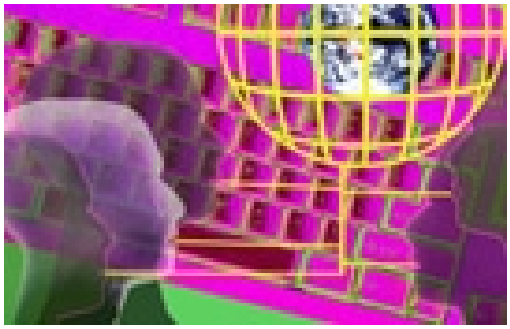
Capítulo 17 C-News

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 área 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**. *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, lo que sitúa a los artículos en uno o más ficheros, opcionalmente los comprime, y los manda a *rnews* en el servidor remoto.

La figura 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 está permitido.

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 propósitos 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*, pero Ud. debería crearlos tanto si planea usar *ihave/sendme* como si no. Después, sustituya todos los números 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 unix 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 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**, así 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 (cómo **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 una 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. 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 propósito 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:.

u Este flag indica a C-News que procese por lotes sólo los artículos de los grupos no moderados.

m Este flaag 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 esdtándar.
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 cierto número de guiones del intérprete de comandos cuyos nombres empiezan por vía; 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 artículos 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 tienen 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 recoge la lista de artículos del fichero *site/togo*, y los pone en varios lotes de noticias. Deberá 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 **UUCP**s compatibles con **BNU**. Si usa una clase diferente de directorios de cola, por ejemplo **UUCP** de Taylor, tendría que escribir el suyo propio.

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. 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 *vía. 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 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 entregado. 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 *script*s, 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


Mejoran la eficiencia en las cervecerías

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, añaadiendo 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 mantener 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 donde. Si no se desea archivarlo, deberá 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 *doex-pire* 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 núumero 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*.

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. 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 lite- ralmente.

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).
Hay dos entradas especiales que son proporcionadas por defecto. Estas 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 explícitamente. Si el nombre del grupo de noticias es **alt.sewer**, y laentrada *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 mis- mas. 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
mugnetsólo 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 re- gistro antiguos se guardan en *log.o, log.oo*, etc.

errlog Este es un registro de todos los mensajes de error creados por C-News. Estos 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 Este 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. Estos son los llamados mensajes de *control*. Se reconocen por la presencia de un campo **Control**: en la cabecera del artículo, el cual contiene el nombre de la operación de control a realizar. Existen varios tipos 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 men-sajes *newgroup* y *rmgroup*. Los grupos de noticias bajo las jerarquías "usuales" sólo pue-de ser creados después de que se haya mantenido una discusión y voto entre los lecto-res de Usenet. Las reglas aplicadas a la jerarquía **alt** permiten algo similar a la anar-quía. Para más información, ver los mensajes regulares de **news.announce.newusers** y 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 servido-res 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 opera-ción 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. Estos son sendsys, versión, 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 versión; 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.

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 noti- cias normalmente llamará ainews para inyectar el artículo en el sistema de noticias. Este comando ejecuta algunas comprobaciones sobre el artículo, completa la cabecera, y com-prueba 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 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 Este 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

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. 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.

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. 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.

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 aquellos 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. 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 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.

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 Ud. 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 Ud. nntpd como servidor independiente, asegúrese de que la línea anterior está comentada en inetd.conf. En cualquier caso, tendrá Ud. 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 Ud. 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 elPOST, lo enviará arnews 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.

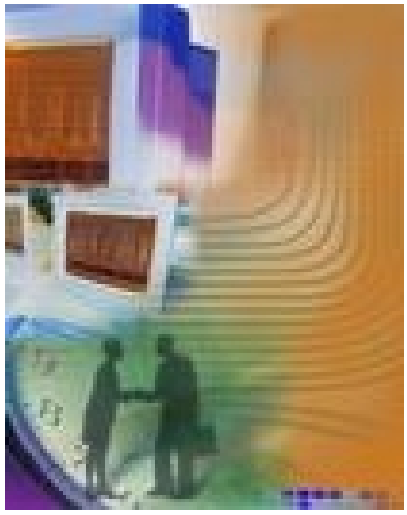
Para llevar a cabo con éxito el protocolo ihave/sendme, nntpd tiene que poder acceder al fichero histórico history. Por tanto, antes de la compilación hay que asegurarse de que la ruta a dicho fichero está correctamente especificada. También hay que asegurarse de que Cnewsy nntpd usen el mismo formato de fichero histórico. Cnewsusa funciones dbm para acceder al fichero; sin embargo hay bastantes implementaciones ligeramente incompatibles de la librería dbm. Si Cnews está 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 it" indica que ya tiene dicho artículo. En cambio, si se obtuviese un mensaje como 35 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 re ejarse 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, ynn.

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. Una de las tareas al escribir un lector de noticias es diseñar un algoritmo eficiente para ordenar los artículos, ya que el tiempo requerido para ello es proporcional al cuadrado del número de artículos.

No discutiremos aquí cómo se construyen 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. 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. 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 Ud. 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. 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.

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 Ud. 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.

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). La “t” en su nombre significa threaded . 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 Ud. 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 Ud. 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.

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", 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.

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. Los indicadores permitidos y sus efectos también vienen detallados en la página de manual de nnmaster.

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, yambos 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 búsqueda
```

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 con-tacta 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 esotra 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 especi-ficado 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
```

Glosario

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.

ACU Automatic Call Unit Unidad de Llamada Automática. Un módem.

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 protoco- lo que sirve para intercambiar información de encaminamiento entre los sistemas autonómos.

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 aplicacio- nes 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 Universi- dad de Berkeley.

CCITT — Comiteé Consultatif International de Télégraphique et Téléphonique, Comité Con- sultivo 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, a~ nadiendo 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 proto- colo 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.

FQDN — Fully Qualified Domain Name, Nombre deNodo Totalmente Calificado. Es un nom- bre 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 pro- tocolos de TCP/IP más conocido, pues es el que se usa para enviar ficheros de un lugar a otro de la red.

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 Funda- ción del Software Libre (FSF) que pretende proporcionar un conjunto de utilidades tipo Unix , 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 otante 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 Estandariza- ción.

ISDN — Integrated Services Digital Network, Red Digital de Servicios Integrados. Nueva tecnología para comunicaciones que utiliza circuitos digitales, y está llamada a sus-tituir a la telefonía tradicional.

HTML — Hiper Text 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.

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 unservicio 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-
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 ujo 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 (Ro-bert 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.

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 Unix .

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. 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.
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 Telara~ na 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, ynamed, 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 deayuda 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 Unix . 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 extra~ nas 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 entra~ nas de los protocolos de Internet.

[Tanen89] Andrew S. Tanenbaum: Computer Networks. Prentice-Hall International, 1989. ISBN 0-13-166836-6.

Este libro trata de temas sobre redes en general. A través del Modelo de Referencia OSI, explica los problemas de dise~ no 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.

[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 peque~ nos 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.

Indice de HOWTOs

A continuación hay una relación de los HOWTOs disponibles actualmente.

- 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). Indice 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.
- 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 " 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.

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@ins ug.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.

1340 Números Asignados, Postel, J., yReynolds, 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., yLapsley, 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).

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, Cerf, V. ¿Quién dice que los heroes de las redes no son poetas?