La Vida en el Nirvana

Como vivir en gnu/Linux

21 de septiembre de 2006

Índice de Contenidos

Capítulo 1. Introducción

1.1. Contenido

Este documento no es un curso de GNU/Linux. Tampoco es una descripción del sistema operativo ni una Guía de Uso. Pretende ser una Guía de Referencia para usuarios de carácter técnico (administradores de sistemas, programadores, usuarios avanzados, etc.)

Como Guía de Referencia, no contiene toda la información que necesitarás para vivir cómodamente en tu sistema GNU/Linux. Simplemente, repasa las principales áreas del sistema operativo, explica los conceptos más importantes y ... te aportará un montón de referencias, casi todas ellas en Internet. Es en estas referencias donde encontrarás todos los detalles, datos e información técnica que necesitarás.

Esta Guía, simplemente te ayudará a encontrar la información que necesitarás y a entenderla.

Naturalmente, este documento contiene errores (y supongo que muchos). Yo ya sé que no soy perfecto, y si tu creías que lo era, lamento defraudarte. Pero puedes ayudarme a mejorarlo enviándome un correo con las erratas, pifias y sugerencias (mjaque@ilkebenson.com). Te lo agradezco de antemano.

Vamos a lo técnico.

Este documento está basado en la distribución Debian (hoy, la última versión estable es la 3.1 Sarge). Supongo que podrá utilizarse, con leves variaciones para otras distribuciones, pero si estás empezando, ¿porqué no pruebas con Debian?

Una última cosita. He incluido las referencias bibliográficas en cada capítulo (también hay referencias generales al final). Estas referencias pueden ser de cinco tipos:

- Libros. Se indican de la forma habitual (título, autor/es, editorial, etc.).
- Páginas Web. En muchas ocasiones, la información se encuentra en un sitio web, pero no puedo darte su ubicación concreta porque cambia al reestructurar las páginas. En estos casos, te indico la URL principal.
- Documentos. Casi siempre de Internet. Podrás encontrarlos en la dirección web que te indico. Si no indico ninguna, es que son generales, están en muchos sitios y puedes encontrarlos buscando por su título en cualquier buscador.
- Documentos HOWTO. Son un caso especial. Aunque pueden encontrarse en Internet, también los tendrás instalados en tu sistema operativo. En el capítulo 3 te explico como acceder a ellos.
- Páginas de Manual. Son las páginas de manual de GNU/Linux. Se accede a ellas desde el sistema operativo, tal y como te explico en el capítulo 3.

1.2. Licencia

1.3. Responsabilidad

No se asume ninguna responsabilidad por los contenidos de este documento. El lector asume el riesgo derivado del uso de los conceptos, ejemplos y cualquier otro contenido. Al tratarse de una nueva edición, este documento puede contener errores e imprecisiones.

1.4. Acerca del Autor

Miguel Jaque Barbero nació en Barcelona en 1968. Es Ingeniero Superior de Telecomunicación por la Universidad Politécnica de Madrid y Máster en Administración de Empresas por el Instituto de Empresa de Madrid.

Ha desarrollado toda su carrera profesional en el sector de la ingeniería de software y, desde 1999 centrado exclusivamente en tecnologías de software libre a través de Ilke Benson (www.ilkebenson.com).

Su actividad profesional se centra desde entonces en el desarrollo de proyectos, formación y consultoría, utilizando exclusivamente estas tecnologías.

Para contactar con el autor: mjaque@ilkebenson.com

Capítulo 2. Conceptos Fundamentales de Software Libre

No puedes meterte en el Nirvana, que es un sistema GNU/Linux, sin tener muy claros algunos conceptos fundamentales. Además, el trabajo fundamental de un informático es... ¡hablar de informática! Así que si quieres "aclarar" algunos conceptos a tus colegas microsoftizados, necesitarás leerte este capítulo.

2.1. ¿Qué es el Software Libre?

2.1.1. Las Cuatro Libertades

El término software libre se utiliza como contraposición al de software propietario para diferenciar un software que se entrega con más derechos. Según la Free Software Foundation el software libre tiene cuatro libertades adicionales y una obligación:

- La libertad de Uso: Entendida como la libertad de utilizar el software sin limitaciones ni restricciones. Es decir, el software libre puede utilizarse en cuantas instalaciones se precise, y para cualquier propósito.
- La libertad de Ayudarse a Uno Mismo: Entendida como la libertad de modificar el software, bien sea para mejorarlo o para adaptarlo a una necesidad concreta. Esta libertad lleva implícito el derecho a disponer del código fuente, sin el cual, no es posible realizar las modificaciones.
- La libertad de Ayudar a los Demás: Entendida como la libertad para distribuir el software, de forma ilimitada y sin restricciones. En el software libre no existe el pirateo. Al contrario, se fomenta la distribución y la difusión del software.
- La libertad de Ayudar a Toda la Sociedad: Entendida como la libertad de publicar el software y sus mejoras, de forma ilimitada y sin restricciones.

Sin embargo, existe la posibilidad de que alguien reciba un código como software libre, lo modifique, y luego lo distribuya restringiendo estos derechos. Esto ocurrió, por ejemplo, con el código del servidor X. Para evitarlo, a las cuatro libertades anteriores se les añade una obligación, el *CopyLeft*.

2.1.2. CopyLeft

El término "CopyLeft" (traducido por Richard Stallman como "izquierdo de copia"[1]), implica la imposibilidad de alguien que haya recibido software libre, se atribuya derechos de propiedad sobre él.

El Copyleft consiste en la obligación de mantener las cuatro libertades del software libre en cualquier distribución o publicación del mismo, ya sea del código original o de las modificaciones realizadas sobre él.

1 ¿Quien le mandará a Stallman hacer traducciones del inglés al castellano?

De esta forma, se garantiza que el software libre SIEMPRE permanecerá libre. Y que a su vez, fomentará la expansión del software libre. Quizás por eso, Microsoft lo denomine *software vírico*.

2.1.3. GPL - General Public License

Se trata de un modelo de licencia que recoge las libertades del software libre y el copyleft. Ha sido diseñada por la Free Software Foundation para facilitar su inclusión en los paquetes de software libre.

Si desarrollas software y quieres que sea software libre, pon una copia de esta licencia o una referencia a ella.

La licencia GPL es muy abierta. Para algunas organizaciones, demasiado, ya que permite incluso la eliminación de las referencias al autor original del código. Por este motivo, algunas compañías han diseñado sus propios modelos de licencia. Suelen ser algo más restrictivos, por ejemplo, porque obligan a mantener la referencia de la empresa que originalmente desarrolló el software, o a que se les comunique cualquier mejora realizada sobre el código, etc. Pero muchas de ellas se consideran igualmente software libre porque no restringen las cuatro libertades básicas. Entre estas organizaciones están Apache, NetScape ... Todas ellas tienen su propio modelo de licencia.

2.1.4. GNU - GNU is Not Unix

GNU es un proyecto que inició Richard Stallman para desarrollar un sistema operativo íntegramente software libre. Este proyecto dio lugar a la ya mencionada Free Software Foundation y el código de sus aplicaciones puede encontrarse allí.

Hoy, el proyecto GNU está completo, aunque sigue desarrollándose y evolucionando. Sin embargo, en 1991, le faltaba una pieza fundamental, el núcleo.

¿Qué es el núcleo?

El núcleo de un sistema operativo (kernel en inglés) es la aplicación que gestiona el acceso del resto de aplicaciones a los recursos principales del sistema (CPU, memoria RAM, buses, memoria de swap, ...).

2.1.5. ¿Qué es Linux?

Aunque habitualmente el término "Linux" designa al sistema operativo GNU, técnicamente, Linux es un núcleo.

Desarrollado en 1991 por Linus Torvalds, profesor de Universidad en Helsinki, para probar el nuevo modo protegido del procesador 386, Linux se convirtió en un proyecto de software libre gracias a su difusión por Internet. Hoy, el núcleo se encuentra en su versión 2.6. y, además de ser totalmente estable, está disponible para un sinfín de arquitecturas e incorpora soporte para prácticamente todo tipo de hardware[2].

Todo lo referente al núcleo de Linux, incluyendo naturalmente el código fuente, está disponible en www.kernel.org.

¿Hay otros núcleos?

Sí. Cada sistema operativo tiene el suyo propio.

Incluso Richard Stallman, quien no se lleva muy bien con Linus Torvalds por cuestiones filosóficas del software libre (Linus representa la corriente Open Source, y no comparte la filosofía "hippie" de Stallman y la Free Software Foundation), ha completado su proyecto GNU desarrollando un núcleo denominado Hurd que se basa en el concepto de microkernels de Andrew Tannenbaum.

Por lo tanto, el sistema operativo GNU puede funcionar con un núcleo Linux o con un núcleo Hurd. Lo que habitualmente entendemos por Linux es realmente GNU/Linux.

También es posible, por ejemplo, ejecutar aplicaciones GNU sobre un sistema operarativo Windows (que tiene su propio núcleo).

2.1.6. ¿Qué es eso de Open Source?

Al referirnos al software libre podemos utilizar dos términos: Software Libre o Código Abierto.

Aunque parezcan iguales, cada uno representa una orientación distinta, con unas ideas y organizaciones diferentes tras ellos.

El término software libre es el defendido por la Free Software Foundation tal y como hemos visto. La idea que hay tras él es que el software debe ser libre, porque los seres humanos tenemos derecho a ayudarnos los unos a los otros sin que nadie (y mucho menos una licencia de software) pueda impedirlo. Conviene señalar que la Free Software Foundation no apoya ni fomenta el pirateo. Al contrario, simplemente señala que, lo razonable es crear software libre, sin negar el derecho de quienes no comparten su filosofía.

La cara visible del Software Libre es, sin duda, Richard Stallman.

2 Por ejemplo... Recientemente mi mujer se compró una cámara digital (no diré marcas). Pues a pesar de ser de los últimos modelos, ya tenía soporte en el núcleo. Claro, que a mi mujer le dije que nuestro ordenador podía descargar sus fotos gracias al sensacional trabajo de su marido.

El término Open Source, defendido por Linus Torvalds, refleja que técnicamente el software libre es mejor que el propietario. Open Source no reconoce un derecho natural en el software, sino que considera el modelo de software libre como el que produce mejores resultados. Open Source fundamenta su posición en las Ventajas del Software Libre que veremos en breve.

Como resulta evidente, no son tendencias opuestas, simplemente puntos de vista radicalmente diferentes.

2.2. Las Distribuciones

En la difusión del software libre, y en especial del sistema operativo GNU/Linux, han jugado un papel primordial las distribuciones.

Como distribución, entendemos una recopilación de programas (normalmente software libre) que funcionan de forma integrada.

Hay muchas organizaciones que realizan distribuciones de GNU/Linux. Algunas son empresas privadas, con ánimo de lucro, otras son fundaciones, y otras... administraciones públicas. Cada una tiene su objetivo. Veamos algunas de las principales.

2.2.1. Debian

www.debian.org

Se trata de una de las distribuciones más características de GNU/Linux y que cuenta con gran número de usuarios. La distribuye la fundación Debian, que no tiene ánimo de lucro y basa su actividad en su Contrato Social.

La Fundación Debian está formada por programadores (hackers), y la inició Ian Murdock en 1993. Hoy sigue siendo dirigida por esta comunidad, que elige entre sus miembros a los dirigentes de forma democrática. Su Contrato Social establece que:

- Debian permanecerá libre al 100%.
- Debian recompensará a la comunidad del software libre, publicando como software libre sus propios desarrollos.
- Debian no esconderá los problemas.
- Las prioridades de Debian son los usuarios y el software libre.
- Debian no pondrá dificultades al software no libre.

2.2.2. gnuLinEx

www.linex.org

gnuLinEx es la distribución desarrollada por la Dirección General de Sociedad de la Información de la Junta de Extremadura para su Red Tecnológica Educativa.

Esta basada en Debian, pero orientada al usuario final, por lo que simplifica notablemente los procedimientos de instalación, configuración y administración del sistema a cambio de perder parte de su flexibilidad.

2.2.3. Otras Distribuciones

- **Red Hat**: *www.redhat.com* - Sin duda, otras de las grandes distribuciones de GNU/Linux. Red Hat es una empresa comercial, y su distribución se caracteriza por la facilidad de instalación y su rápida evolución.
- **SuSe**: *www.suse.com* - El competidor europeo de Red Hat. SuSe, que también es una empresa comercial, dispone de una distribución muy extendida, también resulta sencilla de instalar y utilizar, incorporando las últimas novedades del software libre. Recientemente, la empresa Novell compró a SuSe (compró la empresa, no el código).
- **Ubuntu**: *www.ubuntu.com* – Una nueva distribución, basada en Debian, pero promovida por una empresa comercial.
- **esWare**: *www.esware.com* - Una distribución realizada íntegramente en castellano.
- **Knoppix**: *www.knoppix.org* - Una distribución especial, que permite arrancar "cualquier" ordenador con GNU/Linux, desde un cdrom, sin necesidad de instalar nada de software.

2.3. Ventajas e Inconvenientes del Software Libre

Indudablemente, un software que se entrega con más derechos solo puede tener ventajas. Sin embargo, hay quien considera que el nivel de madurez de este tipo de software no es todavía suficiente. Veamos algunas de sus ventajas e inconvenientes:

2.3.1. Ventajas

- Aprendizaje: De cara a los profesionales informáticos es sin duda la ventaja principal. El software libre permite aprender e investigar de forma mucho más fácil, sencilla e ilimitada. Las posibilidades de elección que ofrece (sin duda un inconveniente para los usuarios finales inexpertos) permiten probar nuevos entornos, analizar posibilidades, arquitecturas, etc. Y todo eso, sin coste y sin limitación.
- Independencia: De cara al cliente corporativo, la ventaja principal. Al utilizar software libre, desaparece la dependencia de un proveedor único (el propietario del software). De esta forma, se incrementa la competencia y se reducen costes.
- Mejora Continua: El software propietario sólo es mejorado por su dueño en función del beneficio que le reportará la mejora. Sin embargo, el software libre "mejora por sí solo". Por este motivo, la velocidad de evolución del software libre, al no estar sometido a consideraciones comerciales, es mucho mayor.
- Reducción de Costes: Aunque el software libre no es necesariamente gratuito, sí es cierto que muchos de sus elementos lo son (sistemas operativos, bases de datos, aplicaciones ofimáticas...). Con él, además de reducir el coste de mantenimiento gracias a la competencia, se reducen también los costes de adquisición. No sólo eso, al no tener que pagar licencias, los presupuestos se pueden invertir en mejoras y adaptaciones que, habitualmente son realizadas por empresas más cercanas, contribuyendo al desarrollo empresarial local (no al de Estados Unidos).

2.3.2. Inconvenientes

Intentando ser imparcial, el software libre también tiene inconvenientes. Personalmente, no considero que ninguno de ellos haga la más mínima sombra a las ventajas anteriores, pero depende de los casos.

- Soporte de Hardware: Casi todos los fabricantes garantizan el funcionamiento de sus equipos en entornos propietarios. Son pocos los que lo hacen en entornos de software libre, dejando que sean comunidades de programadores quienes desarrollen sus drivers. Por este motivo, en ocasiones, puede resultar complicado hacer funcionar componentes poco comunes sobre GNU/Linux, especialmente si el fabricante mantiene en secreto sus especificaciones.
- Miedo al Cambio: A todos nos cuesta cambiar. Cambiar de un entorno conocido a otro nuevo siempre será un problema.
- Complicado out-of-the-box: La gran flexibilidad del software libre hace que pueda resultar complicado a los que se inician en él. Conviene elegir una distribución fácil de instalar y utilizar (como gnuLinEx) y no optar por las más complicadas y potentes (como Debian).

2.4. La Calidad del Software

Una última puntualización. Habitualmente algunos colegas informáticos argumentan que es imposible que un software desarrollado caóticamente por una comunidad de... "colgaos" pueda tener una calidad comparable al desarrollado por una gran empresa comercial.

Bien, la respuesta es obvia. ¡Sí, es imposible! La calidad del software de una empresa comercial... siempre será menor.

El concepto es demoledoramente sencillo:

- Pregunta: *¿Cuándo corrige un error una empresa de software?*
- Respuesta: *Cuando el beneficio esperado de la corrección supera el coste de la misma.*
- Pregunta: *¿Cuándo corrige un error una Comunidad de Desarrolladores Libres?*
- Respuesta: *Tan pronto como pueden.*

Además, por si te quedan dudas, hay un documento archiconocido de Eric S. Raymond, "La Catedral y el Bazar". En él, Raymond explica cómo se organizan los desarrollos libres y cómo consiguen calidades superiores a las comerciales comparándolo con la construcción de grandes catedrales en la Edad Media y la construcción de bazares. No dejes de echarle un vistazo (si es que te quedan dudas).

2.5. Bibliografía y Referencias

- Free Software Foundation. www.fsf.org.
- Comunidad de Desarrollo del Núcleo de Linux. www.kernel.org.
- "La Catedral y el Bazar". Eric S. Raymond. Publicado en Internet.

Capítulo 3. Instalación de GNU/Linux

El primer paso para iniciarse en esta nueva vida es... la instalación de GNU/Linux. Desgraciadamente, este es uno de los más complicados. Pero nadie dijo que el camino hacia la Libertad fuera fácil.

3.1. Preparación

Instalar el sistema operativo suele ser lo primero que hay que hacer, pues apenas hay fabricantes que distribuyan ordenadores con GNU/Linux preinstalado. Y la instalación es uno de los procedimientos más complicados. Así que, si no tienes necesidades especiales, conviene utilizar una distribución sencilla (como LinEx).

La preparación de la instalación tiene dos pasos:

3.1.1. Conseguir el Código

Que GNU/Linux sea software libre, no significa que sea fácil encontrarlo. Si dispones de una buena conexión a Internet puedes conectarte al web de la distribución que elijas y bajarte un montón de megas. Sino, lo más conveniente es acudir al kiosko y comprar alguna revista especializada que incluya una distribución (aunque entonces, no podemos elegir). Por último, podemos comprarla. En las tiendas especializadas se distribuyen versiones comerciales.

La mejor opción: Pídesela a alguien. GNU/Linux puede distribuirse sin problemas. Cualquiera puede dejarte los cds de instalación sin cometer un delito. Después, quématelos, instálate el sistema y pásaselos a otro colega.

¿Qué Necesito?

En función del ordenador donde vayas a instalar necesitaras una versión de instalación u otra. Lo normal es que necesites la típica instalación para arquitectura i386 (esto incluye Pentiums y AMDs) y con acceso al cdrom por bus IDE. Pero si tienes una arquitectura distinta (Motorola, Sparc, PowerPc... o acceso al cdrom por SCSI o sin cdrom...), tenlo en cuenta.

3.1.2. Configurar el Arranque

Tendrás que configurar tu ordenador para que arranque desde el medio de instalación del que dispongas. Normalmente, bastará con configurar la BIOS para que arranque desde el cdrom. Pero, si tienes un cdrom SCSI, también tendrás que configurar la tarjeta de SCSI.

Si ni siquiera tienes cdrom, siempre puedes instalar desde floppy, aunque eso conlleva disponer de unos cuantos. Algunas distribuciones (Debian, por ejemplo) te permiten arrancar la instalación con sólo dos disquetes y después, bajarte el resto del código desde Internet.

Por último, si tienes problemas para configurar el arranque, y tienes Güindos instalado,

puedes bajarte (de tu distribución favorita) el floppy de arranque de instalación desde ese "sistema operativo". Consiste en un .bat y un floppy con la imagen del núcleo (linux.bin).

RECUERDA Si tienes problemas siempre puedes preguntar en las listas de correo y los foros de tu distribución. Todo el mundo está acostumbrado a pedir y ofrecer su ayuda en la comunidad del software libre. Pero no dejes de consultar antes la documentación.

3.2. Instalación de LinEx

LinEx es, sin duda, una de las distribuciones más sencillas de instalar y utilizar. Está orientada al usuario final, y eso se nota en su proceso de instalación. Sin embargo, sólo podrás instalarla para arquitectura i386 y desde un cdrom IDE (lo habitual).

El proceso de instalación hace las siguientes preguntas:

3.2.1. Inicio

Lo primero que vemos al iniciar la instalación es una pantalla de texto de bienvenida. En ella tenemos tres opciones:

- <ENTER> - Para iniciar la instalación en modo gráfico (opción por defecto).
- text install - Para iniciar la instalación en modo texto (útil si la instalación por defecto no detecta nuestro configuración gráfica).
- boot rescue - Para iniciar el arranque de rescate en situaciones desesperadas.

Tras pulsar ENTER, se inicia la instalación.

3.2.2. Configuración de Particiones

El proceso de instalación nos preguntará si queremos instalar LinEx en todo el disco duro (la opción que te recomiendo), en el espacio libre (opción por defecto) o diseñar las particiones a medida. En el último caso, entraremos en una aplicación para el diseño de particiones de disco. Tenemos que tener en cuenta que LinEx (y cualquier versión de GNU/Linux) necesita al menos dos particiones: la raíz, formateada en ext2 (o ext3) y la de swap (no requiere formato). Esta última deberá tener aproximadamente el mismo tamaño que la memoria RAM de nuestro ordenador.

Si optamos por la instalación de LinEx en el espacio libre, hay que tener en cuenta que deberemos disponer de al menos 1,5GB no asignados a otras particiones (ni siquiera Güindos). Si no tienes particiones libres, la instalación de LinEx te permitirá “hacerte hueco” utilizando el programa nparted.

Por último, si utilizamos la opción de "todo el disco", LinEx creará dos particiones lógicas, llamadas /dev/hda5 y /dev/hda6. La primera será la raíz y la segunda la de swap.

Nombres de Particiones en GNU/Linux

Los nombres de las particiones en GNU/Linux pueden resultar curiosos. Todas ellas estarán en el directorio /dev (dispositivos) de nuestro sistema. Las

que empiecen por hd se referirán a nuestros discos IDE, siendo hda el primero y hdb el segundo. Y dentro de ellos, cada partición numerada.

Si tenemos SCSI, el bus se llama sd (sda el primero y sdb el segundo). Y las particiones sda1, sda2, sdb1, sdb2, etc.

También podemos tener cosas más raras.

3.2.3. Instalación del Gestor de Arranque

El gestor de arranque permitirá que nuestro ordenador arranque en el nuevo sistema operativo. Si tienes otros sistemas instalados, no te preocupes, LinEx configurará el gestor (que se llama GRUB) para que puedas también arrancar desde ellos. Lo mejor es utilizar la opción por defecto (instalar gestor de arranque).

3.2.4. Instalación del Sistema Base

Después del último paso, el proceso de instalación se pondrá a copiar y configurar el sistema base desde el cdrom. No hay nada que elegir. Simplemente esperar.

3.2.5. Configuración Inicial

Hay algunas cosas que el proceso de instalación no puede hacer solo. Así que preguntará por la contraseña del administrador (root) y el nombre y contraseña de un usuario normal. Esto último es opcional, pero muy recomendable.

3.2.6. Instalación de Paquetes de Software

A continuación el proceso de instalación completará el sistema base con los paquetes que incluye LinEx, tales como GNOME, Galeón, OpenOffice... Tampoco hay que tomar ninguna decisión.

3.2.7. Rearranque del Sistema

Una vez concluido (media hora en total) el sistema estará listo para arrancar con tu "nuevo y flamante LinEx". Con LinEx, instalar es realmente sencillo.

3.3. Instalación de Debian GNU/Linux

Instalar Debian es mucho más complicado, pero también mucho más potente. Eso significa que podemos instalar GNU/Linux (o GNU/Hurd) en un sinfín de configuraciones distintas. Así que lo primero será asegurarnos de que tenemos los cds adecuados a la arquitectura de nuestro sistema.

3.3.1. Inicio

La instalación de Debian no se realiza en entorno gráfico, así que solo tendremos las opciones de:

- <ENTER> - Para iniciar la instalación en modo texto (opción por defecto).
- boot rescue - Para iniciar el arranque de rescate en situaciones desesperadas.

Realmente, hay muchas más opciones (distintos núcleos, soporte para arquitecturas

variopintas, etc.). Pulsando <F1> obtenemos información de ayuda sobre todas las opciones.

Tras pulsar ENTER, se inicia la instalación.

3.3.2. Configuración Inicial

La instalación de Debian hace unas preguntas iniciales, tales como el idioma del propio proceso, tipo de teclado, etc.

Conviene tener en cuenta que el proceso de instalación nos va a mostrar una lista de opciones y una opción por defecto, que depende del resultado del hardware detectado (no marcará por defecto la configuración de particiones si detecta que son compatibles), y el estado de la instalación. Casi siempre nos convendrá la opción por defecto[3].

3.3.3. Configuración de Particiones

La configuración de particiones se realiza con el programa cfdisk. Necesitamos disponer de al menos una partición raíz, formateada en ext2 (Debian Woody no incluye ext3 en su proceso de instalación, todavía) y otra de swap.

Una vez definidas y creadas las particiones, el proceso de instalación nos preguntará cual será la raíz y cual la de swap. Opcionalmente, podremos montar otras particiones en el sistema.

3.3.4. Instalación del Gestor de Arranque

El gestor de arranque que instala Debian es LILO, y nos ofrece la posibilidad de hacerlo en el MBR (opción por defecto) o en la partición, en cuyo caso, deberemos disponer de otro gestor de arranque ya instalado en el MBR.

3.3.5. Instalación del Sistema Base

La instalación nos preguntará desde donde vamos a instalar el sistema base. Si disponemos de tarjeta de red y esta ha sido detectada por el proceso de instalación, podremos incluso instalar el sistema base desde Internet o desde un disco de red. Sino, tendremos que tirar de cdrom.

3.3.6. Configuración Inicial

La configuración inicial en Debian es mucho más complicada. Entre otras cosas, al configurar la red, nos preguntará si queremos hacerlo mediante un servidor DHCP o BOOTP. Y, ¡cuidado! la respuesta por defecto a esta pregunta es Sí. Claro, que sólo nos llevará perder algo de tiempo si no tenemos ningún servidor DHCP a mano.

3.3.7. Instalación de Paquetes de Software

Otra diferencia notable es la manera en la que Debian determina que paquetes de software debe instalar. Para ello, utiliza la caché de paquetes que crea durante la instalación y que

3 Algunos fanáticos de Debian (como yo) no comparten la opinión de que instalar Debian es complicado. Argumentan que se trata de una "instalación para pollos" porque basta con pulsar siempre el ENTER eligiendo las opciones por defecto.
Esto es cierto. Pero no siempre. Lo malo es detectar estas salvedades y entender cómo tenemos que continuar. Pero conozco informáticos cuya inteligencia era inferior a la de un pollo y que han instalado Debian sin problemas.

dependerá del medio utilizado (Internet, cdrom ...). Además, utilizará los programas tasksel y dselect para permitir que el usuario señale las funciones básicas que tendrá el sistema y los paquetes adicionales que desea.

Así como tasksel es muy fácil de usar, y basta con señalar las funciones del sistema (desktop, servidor de correo...) dselect es muy complicado, y requiere una lectura cuidadosa de las teclas que utiliza.

3.3.8. Configuración de Paquetes de Software

Por si fuera poco, cada paquete que se instala en el sistema puede llevar asociado un proceso de configuración. Esto se traduce en un montón de preguntas, algunas difíciles de entender. Normalmente, las opciones por defecto nos sacarán del apuro (pero no siempre). Léetelo todo, así además aprenderás bastante sobre GNU/Linux.

3.3.9. Rearranque del Sistema

Una vez concluido (puede que mas de una hora) el sistema estará listo para reiniciarse.

3.4. Y... si no funciona

Bien, no hay porque negarlo. Puede que el proceso de instalación no se complete, que cometamos errores o que tras concluirlo, el sistema no arranque. Además de la opción obvia de intentarlo de nuevo podemos utilizar lo siguiente:

- Disco de Rescate: Tanto LinEx como Debian nos ofrecerán la posibilidad de crear un disco de rescate adaptado a nuestro sistema. No la despreciemos. Tras arrancar con el disco de rescate podremos acceder al sistema y configurarlo adecuadamente (aunque para ello tengamos que leernos entero este documento y varias de sus referencias).
- Consola Virtual: Si no hemos creado el disco de rescate, todavía tenemos una opción. El proceso de instalación no es más que un núcleo Linux en ejecución. Podemos por tanto, detenerlo (por ejemplo, sin contestar a una pregunta) y abrir una consola virtual para acceder al sistema. Para ello, bastará con pulsar CTRL-ALT-F1.

3.5. Bibliografía y Referencias

- Installation-HOWTO.
- Install-Strategies-HOWTO.
- Debian Installation Manual.

Capítulo 4. El Intérprete de Comandos (`bash`)

4.1. ¿Qué es un intérprete de comandos?

Es la aplicación que atiende a un usuario tras su proceso de login.

En los sistemas GNU/Linux, el intérprete más utilizado es bash, que son las siglas de "Bourne Again Shell".

4.2. Ayudas a la Productividad de `bash`

En GNU/Linux, el uso de la consola está mucho más extendido que en otros sistemas operativos, especialmente para labores de administración. Por este motivo, los programas de interpretación de comandos incorporan funciones que permiten a los usuarios trabajar de forma rápida y efectiva con ellos.

Bash dispone de dos ayudas:

- Función Autocompletar: Pulsando la tecla TAB bash completará el comando o el nombre del fichero que encuentre en la línea de comando. Si no existiera un único resultado coincidente, bash avisa con un pitido. En ese caso, si el usuario pulsa dos veces la tecla TAB, bash mostrará una lista de coincidencias.
- Histórico: Pulsando las flechas Arriba y Abajo, se accede al histórico de comandos. De esta forma, resulta muy rápido repetir comandos.

4.3. Sintaxis de un comando

La sintaxis de un comando es la siguiente:

```
nombre_del_comado -opciones_cortas --opciones_largar argumentos
```

Hay que subrayar que GNU/Linux distingue entre mAyúScuLas y miNÚsCulaS[4].

4.4. Comandos de Ayuda

Estos son los comandos fundamentales. Y siguen siéndolo incluso para los usuarios más avanzados. Poca gente conoce completamente las opciones de comandos tan utilizados como, por ejemplo `ls`. Afortunadamente, la documentación de GNU/Linux es completa.

- `man` - Acceder a las páginas de manual de un comando. Por ejemplo `man ls`.
- `info` - Muestra información más ampliada sobre un comando.

4 ¿Te he dicho que GNU/Linux distingue entre mayúsculas y minúsculas?

- HOWTO – No es un comando. Son documentos que explican como hacer y configurar determinadas cosas con GNU/Linux. Suelen instalarse en /usr/share/doc/HOWTO, y hay que utilizar algún editor para verlos.

4.5. Comandos de Navegación

Son los comandos que se emplean para recorrer la estructura de directorios:

- `cd` - Cambia de directorio. Por ejemplo `cd /home/usuarios/`.
- `pwd` - Indica en qué directorio nos encontramos.
- `ls` - Muestra los archivos de un directorio.

4.6. Comandos de Gestión de Archivos

Son los comandos que se emplean para modificar la estructura de directorios:

- `mkdir` - Crea un directorio.
- `rmdir` - Borra un directorio.
- `cp` - Copia un archivo.
- `rm` - Borra un archivo.
- `mv` - Mueve un archivo.
- `touch` - Cambia la fecha de acceso de un archivo o lo crea si no existe.

4.7. Comandos de Edición

Son los comandos que llaman a aplicaciones de edición de texto.

- `less` - Es un editor de "solo lectura". Además de utilizarse para visualizar textos, se emplea para controlar la salida de comandos demasiado extensas. Por ejemplo: ls * | less. La ventaja de less es que permite recorrer el buffer de salida en ambos sentidos.
- `vim` - El mejor editor de textos. Lo veremos con detalle (y cuidado).
- `emacs` - Otro editor de textos, pero requiere que el usuario tenga más de once dedos (en cada mano).
- `sed` - Un editor de textos programable mediante scripts.

4.8. Comandos de Búsqueda

- `locate` - Busca nombres de archivo coincidentes en la base de datos del sistema.
- `updatedb` - Actualiza la base de dats de archivos del sistema.
- `find` - Busca archivos con nombres coincidentes, directamente en el disco.
- `grep` - Busca textos coincidentes con la expresión de búsqueda dentro de los archivos.

4.9. Otros Comandos

- `su` - Permite que nos convirtamos en otro usuario.
- `whoami` - Nos indica qué usuario somos.
- `date` - Indica o cambia la fecha del sistema.
- `lpr` - Envía un archivo a la impresora.

4.10. Bibliografía y Referencias

- Toda la información sobre cada comando puede encontrarse en sus páginas de manual (`man comando`).
- La información sobre bash también (`man bash`).
- Los documentos HOWTO puede encontrarse en Internet. También están disponibles en todas las distribuciones. En el caso de Debian, se encuentran en el paquete doc-linux-text (versión de texto en inglés). También están disponibles en otros idiomas, en html, en pdf ...
- Para saber más de bash, puedes consultar su manual en www.gnu.org. Y, en internet y el los Howto encontrarás la Advanced Bash Scripting Guide, por si quisieras programar scripts de bash (duro y difícil).

Capítulo 5. El Editor de Textos (`vim`)

5.1. ¿Porqué `vim`?

Vim es un editor de textos compatible con el famoso vi. Sin embargo, es mucho más humano y tiene funciones muy avanzadas. Vim resulta especialmente útil en la edición de ficheros de programación.

Ciertamente existen otros editores. Pero, al contrario que vim, no resultan fáciles de manejar ni cuando se dominan. En la comunidad GNU/Linux, como ocurre con casi todas sus aplicaciones, también hay dos tendencias: unos usuarios utilizan vim para la edición de sus archivos de texto; otros, prefieren emacs.

La diferencia es que los primeros son, en su mayoría, seres humanos, mientras que los usuarios de emacs suelen pertenecer a razas con más de once dedos en cada mano (o pata).

Lo mejor es que no te dejes influir. ¡ Prueba los dos y elije tú !

5.2. Entrando y Saliendo de `vim`

Arrancar vim es fácil basta con teclear `vim` o `vim fichero` en la línea de comandos. Pero salir ... no es tan fácil

Para salir de vim, basta con pulsar `<ESC>:q<ENTER>` (es decir, la tecla Escape, los dos puntos, la letra q y pulsar el ENTER). En seguida veremos el lógico significado de este comando.

5.3. Modos de Operación

Un programa de edición de textos debe distinguir si el usuario está introduciendo texto para que se incorpore en el fichero o si está indicándole algún comando que debe realizar (guardar, salir, borrar...). En una aplicación de ventanas, es fácil. El texto se introduce por el teclado y los comandos se seleccionan de un menú visual. Pero, vim está diseñado para que se pueda utilizar desde una consola. Y eso, es más difícil.

Para distinguir lo que le pide el usuario, vim utiliza modos. Así, existe un modo de inserción de texto y otro de comandos.

Inicialmente, vim arranca en modo comando. Basta con pulsar `i` o la tecla de `<INSERT>` para cambiar al modo de inserción. Vim indica el modo en que se encuentra en la última línea. Si no aparece nada, el modo es comando.

Las siguientes teclas cambian de un modo a otro:

- `<ESC>` - Pasa a modo comando.
- `i`, `<INS>` - Pasa a modo inserción desde el modo comando.
- `V`, `v` - Pasa a modo visual desde el modo comando.
- `r`, `<INS><INS>` - Pasa a modo reemplazar desde el modo comando.

Vim tiene otros modos, pero estos son los principales.

5.4. Tipos de Comandos

En modo comando, o en cualquier modo que no sea de edición, vim acepta dos tipos de comandos:

- Comandos Ex: Se preceden de dos puntos y necesitan que se pulse la tecla `<ENTER>` para su ejecución. Admiten parámetros y se visualizan mientras se escriben en la línea de comandos.
- Comandos Vi: Basta con pulsar la combinación de teclas para que se ejecuten.

Por ejemplo, si en modo comando tecleamos:

```
:saveas fichero.txt
```

y pulsamos `<ENTER>`, estaremos guardando el fichero (“Elemental querido Watson”).

Sin embargo, si pulsamos `x` borraremos el carácter bajo el cursor.

5.5. Edición Básica

Los comando básicos de edición, además de los necesarios para cambiar de modo, son:

- `:q` - (quit) Cierra la ventana. Si es la última, sale de vim.
- `:w` - (write) Guardar el fichero.
- `:saveas nombre` - Guarda el fichero con el nombre indicado.

- `:wq` - (write and quit) Guarda el fichero y cierra la ventana.
- `:edit fichero` - Abre el fichero indicado en la ventana.

Además, añadiendo `!` a cualquier comando, se fuerza su ejecución. Por ejemplo, `:q!` cierra la ventana aunque no se haya guardado el contenido del fichero.

Por último, cualquier comando precedido de un número (incluso los comandos vi) se repetirán el número de veces indicado.

5.6. Buscando Ayuda

Vim tiene un magnífico sistema de ayuda que incluye documentación completa y tutoriales. Sin embargo, está en inglés.

Para acceder a la ayuda basta con pulsar `<F1>`. Y si quieres información sobre un tema concreto, puedes utilizar el comando help (`:help concepto`).

5.7. Gestión de Ventanas

Vim permite abrir varias ventanas de edición en una única consola de texto (impresionante ¿verdad?).

Todos los comandos de gestión de ventanas está precedidos de `<CTRL-W>` (pero no se deja pulsado, solo tenemos 5 dedos en cada mano).

Algunos de estos comandos son:

- `n` - Abre una nueva ventana (es decir `<CTRL-W >n` abre la ventana).
- `Arriba, Abajo` - Cambia de ventana.
- `+, -` - Cambia el tamaño de una ventana.

En cada ventana, se pueden utilizar los comandos de siempre (`:q`, `:w`...).

5.8. Copiar y Pegar

Copiar y pegar requiere algo de práctica en vim. Pero no es muy difícil.

Pasar a modo visual: Mediante `<CTRL-V>` o `<CTRL-v>` según queramos marcar líneas completas o caracteres, seleccionamos el texto a copiar.

Copiar: En vim se llama *yank* (en inglés, dar un tirón) y se realiza con el comando `y`. Yank envía las líneas marcadas al buffer, pero conviene no hacer ninguna operación entre el yankeado y el pegado, pues operaciones como borrar o modificar cambiarán el contenido del buffer.

Pegar: En vim se denomina *put* y se realiza con la tecla `p`.

5.9. Buscando y Reemplazando

Vim tiene comandos de búsqueda muy potentes. Son:

- `/texto` - Busca un texto en sentido descendente.
- `?texto` - Busca un texto en sentido ascendente.
- `n` - Va a la siguiente coincidencia (en el sentido indicado).
- `N` - Va a la coincidencia anterior (en el sentido indicado).

Buscar y reemplazar es algo más complicado, pues el comando `:search` admite muchas opciones. Su forma más común, que permite reemplazar todas las coincidencias de texto en el fichero editado es:

:%s/texto1/texto2/g

Que significa...

- `:` - Inicia un comando Ex.
- `%` - que se aplicará a todo el documento.
- `s` – el comando es search.
- `texto1` – su primer argumento, el texto a buscar.
- `texto2` – su segundo argumento, el texto de reemplazo.
- `g` – un tercer argumento, indica que la sustitución se hará incluso varias veces por línea.

Bueno, todo es cuestión de mirar en la ayuda, probar y coger soltura... Ánimo.

5.10. Bibliografía y Referencias

Sin lugar a dudas, el mejor sitio para buscar información sobre vim es su página web www.vim.org. Allí, además de toda la información y las referencias, pueden encontrarse trucos y scripts para incrementar la potencia y ergonomía de vim.

Santiago Romero ha escrito un manual de VIM en castellano. Puedes consultarlo en www.sromero.org

Capítulo 6. Instalación de Paquetes de Software con `apt`

6.1. ¿Qué es `apt`?

`apt` es el sistema de Debian para la instalación de paquetes.

Dicho en clarito, es el conjunto de programas que facilitan la instalación de programas en una máquina Debian.

Todos los usuarios de Debian están orgullosos de su sistema de actualización de software. Y, lo que es más importante, los usuarios de otras distribuciones echan en falta un sistema de actualización con la potencia, flexibilidad y comodidad de `apt`.

6.2. ¿En qué se diferencia?

El "otro" sistema de actualización de paquetes es el rpm de Red Hat, que actualmente también es utilizado por otras distribuciones (Suse, por ejemplo).

La diferencia fundamental es la resolución de dependencias.

Como sabes, el GNU/Linux es habitual que cada programa utilice un montón de librerías y programas de base. De esta forma, con programas muy sencillos se consiguen resultados muy potentes.

Pero claro, eso exige que, al instalar o actualizar un nuevo programa, sus dependencias también se actualicen.

apt lo hace automáticamente. Es decir, si te instalas la nueva versión de ... por ejemplo, Mozilla, apt analizará los paquetes de los que depende el paquete mozilla, comprobará que están actualizados en tu sistema y, si no lo están, los buscará y los instalará. Naturalmente, hace el mismo proceso con cada paquete de dependencia.

El resultado es que puedes instalar y actualizar paquetes con un simple comando.

6.3. ¿Qué pasa con rpm?

rpm sólo analiza las dependencias, no las resuelve. Así, al instalarte la nueva versión de Mozilla, rpm detecta que las dependencias no están actualizadas, simplemente te dará un error indicándotelo. Tendrás que buscarte tú mismo el paquete dependiente e instalártelo para poder completar la instalación. Y deberás repetir el mismo proceso para cada dependencia y las dependencias de cada dependencia y las dependencias de cada dependencia de cada dependencia...

Pero... no tienes porqué elegir

La ventaja de rpm es que muchas empresas publican sus programas en ese formato. Hay muchísimas aplicaciones en rpm. No es que en deb (el formato del paquete de Debian)

haya precisamente pocos, pero en rpm, hay más.

Sin embargo, no tienes que renunciar a ninguno. Si tienes una distribución Debian, puedes instalarte el paquete rpm y utilizarlo para instalarte aplicaciones en ese formato.

Igualmente, en otras distribuciones, puedes instalarte el sistema apt e instalarte paquetes Debian. Al fin y al cabo, ambos son software libre.

Lo único que no puedes hacer, es instalar paquetes rpm con apt y viceversa.

Sin embargo, como en software libre nadie se conforma con las limitaciones, hay aplicaciones que migran de un formato a otro. En concreto, `alien` te permite transformar paquetes rpm en deb.

6.4. La Arquitectura del Sistema

Vamos con apt.

apt se basa en una arquitectura con los siguientes componentes:

6.4.1. Las Fuentes de Paquetes

Son las ubicaciones físicas de las colecciones de paquetes. Por ejemplo, Debian tiene publicadas en Internet sus fuentes con 9.000 paquetes. Están en ftp://ftp.debian.org y en cualquiera de sus espejos (ftp.es.debian.org, ftp.fi.debian.org, obelix.ucm.org ...).

Además, otras organizaciones publican colecciones de paquetes, pudiédose utilizar como fuentes adicionales para algunos sistemas. Por ejemplo, www.cica.es dispone de una colección de paquetes que forman un entorno de programación en Java.

Las Fuentes pueden ser de diversos tipos: servidores ftp, servidores http, cdroms, floppies, discos de red, subdirectorios....

6.4.2. Paquetes

Constituyen la unidad básica del sistema apt. Cada paquete se conforma como un archivo con extensión .deb, que realmente es un archivo comprimido (en un estándar abierto) que contiene, básicamente, lo siguiente:

- Descriptor: Indica las características del paquete (nombre, descripción, versión, mantenedor, paquetes de los que depende, conflictos, sustituciones, etc.)
- Código: Los ficheros que deben instalarse en el sistema para que el software funcione.
- Scripts: Programas de preinstalación, postinstalación, preeliminación y posteliminación que deben ejecutarse en cada momento.

6.4.3. Lista de Fuentes

En cada sistema Debian, se alberga una lista de fuentes. Serán las que utilice el sistema apt para la instalación y actualización. Reside en el fichero `/etc/apt/sources.list` y simplemente contiene una lista con cada una de las fuentes, la distribución, áreas que se utilizarán, etc.

6.4.4. Caché de Paquetes

En cada sistema Debian no se almacenan todos los paquetes referidos en la lista de fuentes. Sería imposible disponer de tanto espacio. En su lugar, el sistema apt dispone de una caché en la que recoge todos los descriptores de todos los paquetes de todas las fuentes. De esta forma, el administrador puede buscar un paquete sin tener que conectarse a Internet ni tener que ir de cdrom en cdrom. La caché de paquetes reside en `/var/cache/apt`.

6.5. Los Comandos de apt

Naturalmente, la gestión del sistema apt se realiza con una serie de comandos. Son los siguientes:

- `apt-setup`: Añade fuentes a la lista de fuentes
- `apt-cdrom`: Gestiona los cdrom indicados en la lista de fuentes. Por ejemplo, apt-cdrom add añade un cdrom a la lista y lo escanea para incluir sus descriptores en la caché de paquetes.
- `apt-cache`: Incluye diversas funciones para utilizar la caché de paquetes. Por ejemplo:
  - `apt-cache show paquete`: Muestra el descriptor de un paquete.
  - `apt-cache search texto`: Busca paquetes en cuyo descriptor esté el texto indicado.
  - `apt-cache dotty`: Genera un gráfico con las dependencias entre paquetes.

- `apt-get`: Incluye diversas utilidades para gestionar los paquetes. Por ejemplo:
  - `apt-get install paquete`: Instala el paquete.
  - `apt-get remove paquete`: Quita el paquete (y sus dependencias si no son necesarias para otros paquetes)
  - `apt-get upgrade`: Actualiza toda la distribución
  - `apt-get update`: Actualiza la caché de paquetes.
  - `apt-get source paquete`: Descarga el código fuente de un paquete.

Realmente, como casi todo en GNU/Linux, apt utiliza otros comando de nivel inferior (mucho más complicados). En concreto, apt utiliza el comando `dpkg`.

De él, conviene saber que el comando

```
dpkg-reconfigure paquete
```

Reconfigura el paquete, volviendo a ejecutar el programa de configuración y haciéndonos todas las preguntas necesarias.

Al mismo tiempo, apt es utilizado por otros programas de nivel superior. Por ejemplo, `dselect` en una aplicación en modo casi gráfico que facilita la instalación de paquetes (después de leerte cómo se maneja, pues utiliza unas teclas "un poco raritas").

También hay aplicaciones totalmente gráficas (ventanas, menús, botones, ratón y esas guarradas) que utilizan apt para instalar paquetes. Este es el caso de `synaptic`.

6.6. Bibliografía y Referencias

- Debian Policy Manual. En él se describe la estructura del directorio de Debian (todo eso de main, contrib, non-free) y todos los detalles de la estructura de paquetes de Debian.
- APT Howto y man apt, es la referencia completa con toda la información para utilizar esta herramienta.

Capítulo 7. Sistemas de Ficheros

7.1. La Estructura de Directorios en GNU/Linux

Cuando instalamos un sistema GNU/Linux por primera vez, puede sorprendernos la cantidad y el contenido de los directorios que el proceso de instalación crea en el sistema. No son tan raros, vamos a ver qué directorios hay cuando analizamos nuestra partición raiz (/).

- `bin` - Contiene comandos que pueden ejecutar todos los usuarios del sistema, tales como ls, cp, less...
- `boot` - Contiene los archivos necesarios para el arranque del sistema, tales como la imagen del núcleo, los stages de grub, el mapas del sistema, el fichero initrd...
- `cdrom` - Es el punto de montaje para la unidad de cdrom.
- `dev` - Directorio especial con los ficheros de dispositivos hardware.
- `etc` - Contiene los archivos de configuración.
- `floppy` - Punto de montaje de la unidad de discos.
- `home` - Contiene los directorios de trabajo de los usuarios, salvo root.
- `lib` - Librerías comunes del sistema que son utilizadas por varios comandos y programas, tanto de sistema como de usuario.
- `mnt` - Directorio para otros puntos de montaje.
- `proc` - Directorio especial con los ficheros de procesos del sistema.
- `root` - Directorio de trabajo del superusuario.
- `sbin` - Directorio de comandos solo para el superusuario.
- `tmp` - Directorio para archivos temporales.
- `usr` - Se utiliza para los ficheros que deben ser compartidos por varias máquinas en una misma red.
- `var` - Directorio para archivos de alta variación, tales como los buzones de correo, spool de impresoras, páginas web, ficheros de bases de datos, bloqueos de ficheros y logs del sistema.

7.2. El Sistema Virtual de Ficheros

GNU/Linux permite acceder a distintos sistemas de ficheros. Es decir, puede leer y escribir en discos con formato ext2, ext3, fat, vfat, ntfs, cdroms, floppies... e incluso en sistemas de almacenamiento conectados a través de redes, como servidores Windows, GNU/Linux... Y lo más característico es que, para el usuario, todos los sistemas de

ficheros se manejan igual. Todos responden a los mismos comandos (`cd`, `cp`, `ls`, `touch`, `mv`, `rm`, `mkdir`...) Este nivel de abstracción, destinado a facilitar el trabajo de las aplicaciones y los usuarios, se consigue gracias a una capa de abstracción del núcleo, denominada VFS (Virtual File System).

El Sistema Virtual de Ficheros se encarga de "traducir" los comandos del usuario a instrucciones que el sistema de ficheros destino puede entender. Y a traducir la respuesta de ese sistema a lo que el usuario (o la aplicación) esperan.

Además, el VFS mantiene cachés de páginas de datos y descriptores de ficheros (inodes) para acelerar su acceso.

7.3. Montaje de Sistemas de Ficheros

Para que un sistema de ficheros este disponible es necesario "montarlo". Esto se hace con el comando `mount`.

Al arrancar, el sistema monta automáticamente todos los dispositivos indicados en el fichero `/etc/fstab` que no estén marcados con la opción `noauto`. Y los pone a disposición de los usuarios (con las restricciones indicadas).

Los usuarios normales podrán montar nuevos sistemas de ficheros siempre que estén indicados en la tabla fstab y tengan permisos para ello. Por ejemplo, el floppy y el cdrom. Pero no podrán montar sistemas que no estén es esa tabla, esta operación está reservada para root.

7.4. Los inodes

Cada fichero tiene asociado un descriptor con sus características principales. Este descriptor se llama inode, y puedes consultar el número de inode de cada fichero con la opción -i del comando `ls`.

```
ls -i
```

7.5. Tipos de Ficheros

La ventaja de GNU/Linux es que TODO es un fichero. Esto permite, por ejemplo, escribir en un dispositivo fácilmente, o consultar la información de un proceso con el comando `cat`, etc.

Generalizando, hay cuatro tipos de ficheros, cada uno tiene su propio inode:

- Ficheros: Los normales. En sus páginas pueden albergar datos o código si son programas.
- Directorios: También son ficheros. Pero en este caso, su página de datos contiene números de inodos. Estos corresponden a los ficheros y subdirectorios del directorio.
- Enlaces: Son referencias (nombres) que comparten un mismo fichero. Se construyen con el comando `ln`.

```
ln /home/usuario/fichero /home/otrousuario/elFichero
```

Con este comando se crea un enlace duro. Es decir, los directorios `/home/usuario` y `/home/otrousuario` contienen el mismo inodo (el mismo fichero). Los enlaces duros no pueden hacer referencia a directorios.

```
ln –s /home/usuario/directorio
    /home/otrousuario/elDirectorio
```

En este caso, el enlace es simbólico (o blando), y consiste en un fichero de texto que el sistema operativo interpretará (por ejemplo, para navegar a él). Pero en este caso, ambas referencias (`/home/usuario/directorio` y `/home/otrousuario/elDirectorio`) son dos ficheros (y dos inodes) distintos.

• Ficheros Especiales: Hay dos tipos: los dispositivos, que se sitúan en /dev y los procesos, que se sitúan en /proc. Tanto unos como otros pueden ser manejados como ficheros gracias al VFS. Eso quiere decir que es posible leer, por ejemplo, un cdrom igual que se lee un fichero de texto. O consultar la situación de un proceso del mismo modo.

7.6. Sistemas de Ficheros

Linux (el núcleo) reconoce varios tipos distintos de sistemas de ficheros. Puedes saber qué sistemas están soportados consultando `/proc/filesystems` (recuerda que se trata de un fichero especial del /proc).

Estos son algunos de los sistemas de ficheros más importantes:

- **ext2 y ext3**. Son los sistemas más habituales. Se trata de una evolución del sistema minix que fue el primero utilizado en Linux. La diferencia entre ext2 y ext3 es que este último incluye journaling, por lo que en caso de caída no requiere realizar un escaneo completo del disco y con eso se reduce mucho el tiempo de recuperación.
- **msdos, vfat, ntfs**. Son los sistemas de ficheros utilizados por los sistemas operativos de Microsoft (DOS, Windows 3, 95, 98 Me, NT, 2000, etc.). Linux soporta también umsdos que es una versión extendida y compatible que permite utilizar nombres largos de fichero. Además, Linux soporta sistemas de ficheros típicos de otros sistemas operativos: hpfs de OS/2, ncpfs de Novell, hfs de Apple, etc.

- **iso 9660**. Es el sistema de ficheros utilizado en los cdrom y dvds.
- **Reiserfs**. Es un sistema de ficheros tan rápido como ext pero especializado en el manejo de grandes directorios con pequeños ficheros (www.namesys.com)
- **jfs**. Sistema de ficheros de IBM con journaling.
- **xfs**. Sistema de ficheros de la plataforma SGI IRIX, con altas prestaciones. Incluye journaling, es totalmente multithread, soporta ficheros de gran tamaño y tiene unas magníficas prestaciones en cuanto a rendimiento y escalabilidad.
- Linux también soporta sistemas distribuidos de ficheros, como **NFS, Samba, CIFS, SecureRPC, Coda...**

7.7. Bibliografía y Referencias

El VFS pertenece al núcleo Linux, por lo que la mejor documentación es el propio código del núcleo (www.kernel.org). Claro, que pocos hay dispuestos a echarle un vistazo. Para ellos, existen las siguientes referencias, todas ellas muy obsoletas, aunque útiles:

- The Linux Kernel Guide, podéis encontrarla en www.tldp.org y su capítulo 9 está dedicado al sistema de ficheros EXT2 y al VFS.
- También hay un documento en castellano escrito por Juan Antonio Martínez Castaño, en la misma dirección (bajo el tema de artículos periodísticos).

Capítulo 8. Gestión de Usuarios

8.1. Usuarios, Contraseñas y Grupos

Las máquinas GNU/Linux son multiusuario. Eso significa que varios usuarios pueden utilizarlas de forma simultánea.

Para permitirlo, el sistema operativo gestiona los nombres de usuario, sus contraseñas y sus características. Pero además, estos usuarios se estructuran en grupos de tal forma que todo usuario tiene su grupo principal y, además, puede pertenecer a otros grupos.

Todo esto, debe ser organizado y gestionado por el administrador, quien dispone de un nombre especial (root) y permisos especiales.

8.2. Comandos de Gestión de Usuarios

Existen varios comandos relacionados con la gestión de usuarios. Son:

- `adduser` - Crea un nuevo usuario.
- `deluser` - Elimina un usuario existente.
- `passwd` - Modifica la contraseña de un usuario.
- `addgroup` - Crea un nuevo grupo.
- `delgroup` - Elimina un grupo existente.

Lo único que hacen estos comandos (y no es poco) es editar un conjunto de ficheros de textos. Y, en algunos casos, crear y eliminar ficheros y directorios (por ejemplo, los de trabajo del usuario). Es decir, nada que no pueda realizar "a mano" un buen administrador.

8.3. Ficheros Asociados a la Gestión de Usuarios

Si bien pueden utilizarse los comandos anteriores para gestionar los usuarios del sistema, su comportamiento por defecto puede no ser suficiente para algunas instalaciones. Si el administrador realmente quiere controlar la gestión de sus usuarios puede hacerlo conociendo y editando los ficheros de texto asociados.

/etc/passwd

Guarda la información de los usuarios. Se trata de una base de datos, con una línea por usuario, en la que se indica su nombre, clave encriptada, UID (el número de identificación de usuario, GID (identificador de su grupo principal), datos del usuario, directorio de trabajo y aplicación de atención.

Bastará con editar y modificar la información de este fichero para crear, eliminar y modificar la información de los usuarios.

/etc/group

Este fichero contiene la información de grupos. También se trata de una base de datos con un línea por grupo. En ella se indican: nombre del grupo, contraseña, GID y una lista separada por comas de nombres de usuarios que pertenecen al grupo.

/etc/shadow

El fichero passwd tiene un problema de seguridad. Debe ser legible por todos los usuarios, aunque solo root tiene permiso para modificarlo.

Sin embargo, esto significa que cualquier usuario tiene acceso a la contraseña encriptada de root y puede intentar romperla. Para evitar esto se creó el sistema de shadow.

Mediante shadow, el fichero /etc/passwd no contiene las contraseñas. En su lugar, aparece una x. Las contraseñas, y la información para su gestión se almacenan en el fichero /etc/shadow al que solo root tiene acceso.

Al igual que ocurre con passwd y group, el fichero shadow es editable con cualquier editor de texto plano (por ejemplo, con vim).

Normalmente, los administradores de sistemas GNU/Linux, editan los ficheros de configuración, pero utilizan el comando `passwd` para modificar las contraseñas. Los buenos administradores se generan sus propios comandos de administración, que asignan los grupos y permisos que realmente desean.

8.4. Permisos

Toda esta gestión de usuarios se realiza para controlar el acceso a los recursos (ficheros, hardware, comandos, aplicaciones...) del sistema. Afortunadamente, para GNU/Linux todo son ficheros, así que basta con establecer una política de permisos sobre ficheros para tener controlado el sistema.

8.5. Categorías de Usuarios

Para cada fichero (o lo que sea) en GNU/Linux se establecen tres categorías de usuarios:

- (U)suario: Es el propietario del fichero.
- (G)rupo: Es el grupo al que pertenece el fichero.
- (O)tros: Es el conjunto de usuarios que no son ni el propietario del fichero ni pertenecen al grupo al que pertenece el fichero.
- (A)Todos: No es una categoría, sino el conjunto de todas ellas.

Para cada categoría pueden establecerse permisos distintos.

8.5.1. Tipos de Permisos

Sobre cada fichero existen tres permisos básicos:

- Lectura (r): Es el permiso para leer el contenido del fichero.
- Escritura (w): Es el permiso para modificar el contenido del fichero y para borrarlo.
- Ejecución (x): Es el permiso para ejecutar el fichero.

8.5.2. Políticas de Gestión de Permisos

A la hora de establecer los permisos de un fichero hay que hacerse tres preguntas:

¿A quién debe pertenecer el fichero?
¿A qué grupo debe pertenecer el fichero?
¿Qué permisos deben establecerse?

Lo importante es darse cuenta de que sin las dos primeras preguntas, la tercera no tiene sentido.

8.5.3. Comandos para Gestionar Permisos

Estos son los comandos necesarios para establecer y gestionar los permisos:

- `ls -l` - Muestra el contenido de un directorio, incluyendo los permisos de cada fichero. Los permisos aparecen en la parte izquierda de cada línea. El primer carácter indica si se trata de un directorio, y después, agrupados de tres en tres (rwx) están los permisos del usuario, del grupo y del resto de usuarios. Finalmente, se muestra el dueño y el grupo del fichero.
- `chown` - Permite cambiar el propietario de un fichero.
- `chgrp` - Permite cambiar el grupo de un fichero.
- `chmod` - Permite establecer los permisos de un fichero. Esto puede hacerse de forma "humana" utilizando las abreviaturas u, g, o y a para establecer permisos de la forma `chmod ug+x fichero` (añade permisos de ejecución para el usuario y el grupo) o `chmod a=rw fichero` (establece los permisos de lectura y escritura para todas las categorías). O también puede utilizarse la forma octal, atribuyendo los valores 4 para lectura, 2 para escritura y 1 para ejecución.

8.5.4. Permisos Especiales

Además de los permisos indicados, hay tres considerados especiales:

Permisos de Directorio

Conviene señalar que el permiso de ejecución sobre un directorio permite a los usuarios que lo tengan navegar hasta él.

Permiso de SUID

Permite que un fichero se ejecute con los permisos de su propietario, y no con los del usuario que lo llama.

El permiso SUID solo tiene sentido si, además, el fichero tiene permiso de ejecución. Se muestra como una s en el comando `ls`.

Permiso de SGID

Funciona igual que SUID pero en lugar de los permisos del propietario, el fichero se ejecuta con los del grupo.

Permiso Sticky

Aplicado sobre un directorio, evita que usuarios distintos del propietario puedan borrar o mover ficheros sobre los que tienen permisos de escritura. Se muestra con la letra t.

8.6. Bibliografía y Referencias

La mejor documentación sobre los comandos está, como siempre, en sus páginas de manual. También podemos encontrar toda la información de shadow, y la estructura de su fichero en la página de manual de shadow.

Capítulo 9. Configuración de Redes

Los sistemas GNU/Linux son intrínsecamente máquinas TCP/IP. Aunque pueden trabajar con otros protocolos de red, e incluso hay distribuciones que implementan comunicaciones por SNA, una máquina GNU/Linux sin TCP/IP es, cuanto menos, rara.

9.1. Configuración de TCP/IP

Configurar las comunicaciones requiere configurar cada uno de los elementos que integran la estructura de comunicación. Estos son los siguientes:

9.1.1. Estructura

- Interfaz: Es la "tarjeta de red", el enganche físico entre el cable de red que vayamos a utilizar y el bus de comunicaciones de nuestro ordenador. En este tema abordaremos la configuración de dos tipos de interfaz: tarjetas para redes Ethernet y módems para comunicaciones PPP.
- Módulo: Es el software que permite gestionar el hardware del interfaz. Es propio para cada uno de los interfaces. En otros sistemas se denominan drivers.
- Núcleo: El propio núcleo del sistema operativo forma parte de la estructura de comunicación, ya que debe estar compilado para permitir los servicios de TCP/IP que vayamos a utilizar. La configuración del núcleo la abordaremos en otro tema.
- Demonios del Protocolo: La implementación de las reglas del protocolo es responsabilidad de diversos demonios de comunicación. Su configuración es crítica para el funcionamiento de la estructura de comunicación.
- DNS (Domain Name Service): El Servicio de Nombres de Dominios es solo uno de los servicios que se presta sobre TCP/IP. Pero debido a la importancia que tiene en la correcta configuración de nuestras comunicaciones, lo mencionaremos aquí.

9.1.2. Ficheros de Configuración

Como casi todo en GNU/Linux, la configuración de cada uno de los elementos de la estructura de red, se puede hacer editando un fichero de texto. Estos son:

/etc/network/interfaces

Contiene la información de los interfaces del sistema. La línea auto indica qué interfaces se intentarán iniciar en el arranque del sistema.

Además, este fichero contiene la información de configuración del protocolo TCP/IP para cada interfaz. Es decir, direcciones de red, máscaras de subred, gateways, direcciones de broadcast, etc.

/etc/modules[5]

Este fichero contiene la lista de módulos que se cargarán al inicio del sistema. Si queremos que un determinado interfaz se inicie en el arranque, debemos asegurarnos de que el nombre de su módulo figure en este fichero.

/usr/src/.config

Este es el fichero de configuración de la compilación del núcleo. Sólo lo menciono aquí para tener la lista completa. Pero no conviene modificarlo sin un aaaammmmpppplllliiiiooo conocimiento de lo que estamos haciendo. Ya lo veremos cuando hablemos de compilación del núcleo.

/etc/hosts

Este fichero contiene una lista de direcciones IP y los nombres que les asociamos. De esta forma, el sistema "conoce" algunos nombres de máquinas y nos evita tener que introducir toda la dirección IP. Suele utilizarse para nombres de máquinas locales, que no pueden resolverse mediante DNSs (si no son privados).

/etc/resolv.conf

Para resolver los nombres de dominio que no figuren en nuestro fichero /etc/hosts necesitamos recurrir al servicio de DNS. En este fichero (/etc/resolv.conf) indicamos al sistema qué servidores de DNS queremos utilizar.

Nota: Si tu máquina utiliza interfaces conectados a buses distintos del IDE o PCI (por ejemplo, PCMCIA en un portátil), estos no son los archivos de configuración. En el caso de PCMCIA, hay que utilizar los archivos que encontraremos en /etc/pcmcia.

9.2. Comandos Útiles para la Configuración de la Red

Tras editar y modificar adecuadamente estos ficheros, nuestra red funcionará. Pero como a veces es necesario algo de ayuda, aquí hay algunos comandos útiles:

- `lspci` - Lista los nombres de los dispositivos conectados al bus PCI, lo que nos puede ayudar a identificar nuestros interfaces sin necesidad de utilizar el destornillador.
- `lsmod` - Lista los nombres de los módulos cargados por el núcleo.
- `insmod` - Carga un módulo en el núcleo.

5 Sólo para núcleos anteriores a 2.6. Los núcleos 2.6 cargan los módulos de forma automática.

- `/etc/init.d/networking restart` - Es el comando para reiniciar los demonios de red. Conviene ejecutarlo tras modificar los archivos de configuración.
- `ifconfig` - Es el comando básico de la gestión de interfaces. Permite conocer su estado, arrancarlos, tirarlos...
- `iwconfig` – Nos muestra los interfaces de red inalámbricos disponibles en el sistema.
- `ping` - Es el comando más utilizado para comprobar el funcionamiento de una red. Este comando comprueba que una máquina está accesible en la red y muestra los tiempos de comunicación. `ping www.terra.es` puede ser utilizado para comprobar el funcionamiento de la red, incluyendo el servicio de DNS. Y `ping 195.235.113.3` prueba la red, pero no el DNS.

9.3. Comunicaciones Punto a Punto con PPP

Existe otro protocolo muy utilizado en la comunicación TCP/IP. Es la conexión punto a punto, habitualmente utilizada para conectarnos a un ISP (Proveedor de Acceso a Internet) mediante una línea telefónica y un módem.

Para ello, el protocolo más utilizado es PPP (Point to Point Protocol), que funciona sobre TCP/IP. En este caso, el interfaz de la estructura es el módem, el módulo será el correspondiente para el módem (si es un linmodem) o el soporte para comunicación por puerto serie, etc. Pero los ficheros de configuración del protocolo son otros, residen en /etc/ppp.

9.3.1. Configuración

Pero así como es habitual modificar la configuración de las conexión Ethernet accediendo directamente a sus archivos de configuración, en el caso de PPP se suele utilizar un programa de configuración llamado `pppconfig`.

Este programa nos preguntará, mediante un interfaz gráfico de ncurses[6] los parámetros necesarios para configurar nuestra conexión, tales como número de teléfono del proveedor, nombre de usuario, contraseña, etc. Y, sobre todo, el dispositivo de módem que vamos a utilizar.

Si el módem se conecta mediante puerto serie, lo podremos encontrar en `/dev/ttyS0` o en algún otro de los puertos (`/dev/ttyS1`, `/dev/ttyS2`, etc.). Pero si el módem es interno (de los denominados winmodems) habrá sido necesario buscar, encontrar y configurar su módulo. Lo habitual es que este proceso, nos instale el módem en `/dev/modem`.

9.3.2. Comandos para la Gestión de Comunicaciones PPP

Así como en una conexión Ethernet, el comando `ifconfig` nos permite arrancar y

6 "¿Qué no estás familiarizado con las ncurses? - Pues espera a que empecemos con el núcleo."

cerrar interfaces, en el caso de PPP los comandos son:

- `pon` - Arranca una conexión PPP. Si no le indicamos un nombre, la conexión por defecto se denomina provider.
- `poff` - Termina una conexión PPP.

9.4. Bibliografía y Referencias

- Toda la información sobre el fichero `/etc/network/interfaces` está en `man interfaces` (¿dónde sino?).
- Toda la información sobre el fichero `/etc/hosts` está en... `man hosts`.
- Y toda la información sobre el fichero `/etc/resolv.conf` está en `man resolv.conf` (Fácil ¿verdad?).
- Saber que toda la información sobre `pppconfig` está en `man pppconfig` no tiene misterio.
- Pero saber que en `www.linmodems.org` se puede encontrar toda la información (y enlaces a los módulos) para configurar un winmodem puede ahorrar muchas horas de trabajo y frustración.

Capítulo 10. Arranque del Sistema

Vamos a empezar desde el principio. Nuestro objetivo es saber qué ocurre en nuestra máquina desde que pulsamos el botón de ON/OFF hasta que aparece el prompt para identificarnos.

10.1. El Proceso de Arranque

Al arrancar el ordenador, lo primero que hace es comprobarse a sí mismo. Es el denominado *Power on self test*. Después, un programa llamado cargador de boot que reside en la ROM BIOS toma el control. La misión de este programa es encontrar un sector de arranque.

Un sector de arranque es el primer sector de un disco, que contiene un pequeño programa capaz de cargar un sistema operativo. Los sectores de arranque se reconocen porque tienen el valor 0xAA55 (43603) en los dos últimos bytes del sector (0x1FE y 0x1FF). A este primer sector, en un disco duro, se le denomina MBR (Master Boot Record).

El cargador de arranque tiene una lista (en la BIOS) de sitios donde buscar sectores de arranque. Normalmente, buscará primero en el floppy, luego en el cdrom, sino en el disco duro... Cuando encuentra un sector de arranque, lo carga en memoria y le pasa el control.

En una máquina GNU/Linux este primer sector forma parte de un *gestor de arranque*.

10.2. Gestores de Arranque

El gestor de arranque es el primer programa que se carga al arrancar un ordenador. Su misión es cargar un sistema operativo (donde quiera que se encuentre) y transferirle el control.

Todos los sistemas operativos tienen sus gestores de arranque, pues es imposible instalar un sistema operativo entero en el sector de arranque. El problema es que algunos sistemas operativos disponen de gestores de arranque muy pobres, incapaces de cargar otros sistemas operativos que no sean los suyos propios. Este es el caso de OS/2 y de ... Güindos

Naturalmente, los gestores de arranque utilizados en GNU/Linux no sólo son capaces de arrancar su sistema operativo, sino que pueden ser configurados para que el ordenador arranque en cualquiera de los sistemas operativos que tenga instalados (en distintas particiones o discos), a elección del usuario.

Así que, si tienes un sistema operativo anticuado, instálalo primero y luego, instala y configura alguno de los gestores de arranque de GNU/Linux. De esta forma, en el MBR tendrás un buen gestor de arranque capaz de arrancar cualquiera de tus sistemas operativos.

Para un sistema GNU/Linux hay dos gestores de arranque típicos:

10.2.1. LILO

Es el más utilizado.

LILO instala en el MBR un pequeño programa llamado cargador de primer paso cuya única misión es cargar otro programa más grande (el cargador de segundo paso) que será quien se encargue de cargar el sistema operativo definitivo.

Este segundo cargador, si se configura para ello, mostrará un menú de opciones al usuario con los diferentes sistemas operativos que puede arrancar.

Configuración de LILO

LILO se configura en el archivo de texto /etc/lilo.conf. El archivo de configuración tiene dos partes, una general y otra por cada sistema operativo que queramos arrancar.

La configuración general es del tipo:

```
boot = /dev/hda
compact
install = /boot/boot.b
map = /boot/map
```

La opción `boot` indica el dispositivo en el que se instalará el gestor de arranque. En este caso, en el MBR del primer disco IDE. La opción `compact` realiza una serie de optimizaciones. No la quites si no vas a experimentar. La opción `install` señala al fichero que se instalará en el sector de arranque. Y por último, la opción `map` señala el fichero de mapa que LILO creará al instalarse con una descripción del sistema.

Además de esta sección general, en fichero lilo.conf debe contener una sección por cada sistema operativo a arrancar. Son de este estilo:

*** GNU/Linux:**

```
image = /boot/vmlinuz   #Localización del núcleo
label = GNU/Linux #Etiqueta para el menú de opciones
root = /dev/sda1  #Localización de la partición raíz
vga = ask   #Pregunta el modo de texto VGA a utilizar.
                #También podría ser
                #     normal (80x25)
                #     extended (132x60)
```

*** Otros:**

```
other = /dev/hda1  #Localización de la partición
table = /dev/hda #Localización de la tabla de particiones
label = Güindos   #Etiqueta para el menú de opciones
```

Si necesitamos pasar algún parámetro especial al núcleo en el arranque, podemos añadir una línea `append`. Por ejemplo, `append = single` para arrancar en modo monousuario.

Instalación de LILO

El problema de LILO es que necesita escribir los datos de configuración en el sector de arranque. Para ello, tras modificar el fichero de configuración ES IMPRESCINDIBLE ejecutar

```
# lilo
```

Este programa leerá el fichero de configuración y modificará el sector de arranque. Si no se hace, el sistema puede no arrancar, pues en el sector de arranque no figurará la posición del núcleo a utilizar.

10.2.2. GRUB

GRUB es un gestor de arranque mucho menos extendido que LILO, pero también mucho más potente y flexible.

La diferencia entre ellos reside en que GRUB, al contrario que LILO, es capaz de entender distintos sistemas de ficheros. De esta forma, al arrancar GRUB, es capaz de leer directamente su archivo de configuración y buscar la imagen del núcleo. Esto hace que no sea necesario ejecutar ningún programa de escritura de sectores de arranque tras la configuración, el sector de arranque no varía.

También nos puede resultar útil en otras circunstancias. Por ejemplo, LILO solo es capaz de arrancar aquellos núcleos que están indicados en su fichero de configuración y que han sido escritos en el sector de arranque (ejecutando el comando lilo). GRUB no. Al ser capaz de leer sistemas de ficheros, con GRUB podemos entrar en el menú de inicio y buscar la imagen que queremos arrancar. Aunque no se la indiquemos en el fichero de configuración.

En el resto, es muy parecido a LILO. También consta de dos pasos, denominados stage1 y stage2. También dispode de un stage1_5 que es quien sabe manejar un determinado sistema de ficheros. Con stage1_5 incluso podemos variar la posición del fichero de stage2.

Configuración de GRUB

GRUB se configura en el fichero /boot/grub/menu.lst. Y como no, consta de dos secciones, una general y otra para cada sistema operativo que queramos arrancar.

La parte general un fichero de configuración típico es:

```
timeout 5   #Tiempo de espera del menú de opciones
default 0   #Opción a arrancar por defecto
fallback 1  #Opción a arrancar si falla la seleccionada
foreground 35943d #Color de la letra del menú
```

La parte particular para cada sistema operativo es del tipo:

*** GNU/Linux:**

```
title GNU/Linux   #Nombre de la opción en el menú
root (hd0,0)      #Partición a arrancar
kernel /boot/vmlinuz    #Localización de la imagen del
                        #núcleo
```

*** Otros:**

```
title Güindos           #Nombre de la opción en el menú
rootnoverify (hd0,1)    #Partición a arrancar, pero no la
                        #comprueba
makeactive        #Activa la partición (para primarias)
chainloader +1          #Carga el primer sector
```

Otras Características de GRUB

GRUB tiene un modo interactivo al que podemos acceder desde el menú de opciones pulsando "c". Y pedir ayuda con el comando "help". Esto debería ser suficiente para salir de las situaciones más desesperadas.

También podemos ejecutarlo desde el propio sistema operativo, con el comando:

```
# grub
```

Instalación de GRUB

Podemos instalarlo desde el mismo modo interactivo, con los comandos:

`grub> root (hd0,2)` - Indica la partición con los archivos de GRUB

`grub> setup (hd1)` - Identifica el disco sobre el que instalar GRUB en su MBR

10.3. Rescate del Sistema

Pero no siempre sale bien. A veces, un mensaje del tipo "Kernel Panic:" nos indica que ha sido imposible cargar la imagen del núcleo. Otras, al arrancar, ni siquiera es capaz de encontrar un núcleo.

Así que necesitamos saber como recuperar un sistema que no arranca.

La estrategia es siempre la misma. De alguna forma tenemos que conseguir arrancar una imagen del núcleo, que no esté dañada y que sea compatible con nuestro sistema. No basta con el núcleo, también necesitamos un sistema de ficheros raíz que esté en condiciones.

Hay tres opciones:

- Disco de Rescate: Si disponemos de un disco de rescate, este tendrá un núcleo compatible con nuestro sistema. Si nuestro sistema raíz funciona, ya podemos arrancar. Sino, necesitaremos también un disco con un sistema de ficheros.

  Podemos crear nuestros propios discos de rescate. Es interesante e instructivo, pero medianamente completo. También podemos bajar discos de rescate para configuraciones estándar de cualquier distribución GNU/Linux. Allí encontraremos las instrucciones para crearlos.

- Distribución Live: Vaya, se nos olvidó crear el disco de rescate o el que hicimos no funciona. ¡No problem!, podemos coger una distribución de tipo live (LinEx-live, GuadalinEx, Knoppix, GnomeCD-Live...) y arrancar nuestro sistema desde ella.

  Estas distribuciones no instalan nada en el disco duro, y nos permiten disponer de un sistema operativo completo, con un montón de aplicaciones, simplemente desde el cdrom.

  Si no tenemos ninguna distribución live, siempre podemos tirar del cdrom (o los discos) de instalación. Basta con iniciar el proceso de instalación y en el prompt de boot: poner rescue root=/dev/sda1.

  Naturalmente, root señala a la partición donde podemos encontrar una imagen del núcleo que no esté corrupta. El parámetro rescue arrancará el sistema en modo monousuario, por lo que no necesitaremos la contraseña de root (¡¡Qué agujero de seguridad!!).

- Gestor de Arranque: Pero cualquiera de estas dos opciones resulta muy incómoda. Un administrador precavido nunca las necesitará. Basta con configurar el gestor de arranque para que nos ofrezca la posibilidad de arrancar en modo rescate con una imagen segura. En el caso de GRUB, incluso pueden establecerse contraseñas de acceso a determinadas opciones del menú.

10.3.1. Y después... ¿qué?

No todo es encontrar el núcleo. Una vez cargado, el núcleo empieza a enviar mensajes a la consola sobre las operaciones que va realizando. Estos mensajes, junto a los mensajes de ejecución, se almacenan en /var/log/syslog.

Calor, que algunos mensajes se generan incluso antes de que esté en funcionamiento el sistema de log. Estos mensajes pueden verse utilizando el comando `dmesg`.

Entre los mensajes que aparecen está el tipo de consola, la velocidad de la CPU, los dispositivos PCI, la versión del núcleo, puertos... Si algo va mal en el arranque, estará ahí.

Finalmente, cuando hemos conseguido cargar un núcleo compatible con nuestro sistema, y este ha encontrado un sistema de ficheros raíz, se pone en marcha el primer proceso, el padre de todos los procesos... `init`.

10.4. init

Casi todos los sistemas GNU/Linux usan el denominado estilo SystemV para su proceso init, cuya misión es tener todo lo necesario en funcionamiento.

La configuración del proceso init está en el fichero /etc/inittab. Lo primero que hace es ejecutar un script llamado /etc/init.d/rcS. Básicamente chequea y monta los sistemas de ficheros, pone el reloj, habilita la swap, arranca la red... Realmente, lo que hace este script de inicialización, es ejecutar todos los ficheros que encuentra en /etc/rcS.d.

Posteriormente, init entra en el run level indicado en su fichero de configuración. El estilo SystemV determina distintos niveles de ejecución para una máquina. Algunos están predefinidos:

- Run Level 0: Parada del sistema.
- Run Level 1: Modo monousuario, con un conjunto mínimo de controladores.
- Run Level 2-5: Modos multiusuario.
- Run Level 6: Reinicio del sistema.

Podemos cambiar de nivel de ejecución con el comando `init`. Así que `init 6` es lo mismo que `restart`.

Al entrar en un nivel de ejecución, el sistema va al directorio correspondiente (/etc/rcN.d, donde N es el nivel de ejecución) y...

1. Ejecuta los programas que empieza por K, en el orden especificado y pasándoles el parámetro stop.
2. Ejecuta los programas que empieza por S, en el orden especificado y pasándoles el parámetro start.

Como se puede ver, estos programas no son más que enlaces a demonios y programas que residen en /etc/init.d. Así que los servicios de que dispondremos en cada nivel de ejecución pueden ser perfectamente configurados por el administrador sin más que añadir o eliminar enlaces de esos directorios.

Pero... hacerlo directamente a mano tiene un problemilla. Cuando actualices tu sistema, el proceso de actualización recuperará la configuración original de cada directorio rc. Si quieres que el conjunto de demonios que se arrancan y se detienen en cada runlevel permanezca tras la actualización del sistema, deberás utilizar el comando `update-rc.d`

Fácil ¿verdad?.

10.5. Bibliografía

- En “From Power Up to Bash Prompt HOWTO“ hay toda una explicación técnica del arranque del sistema.
- man lilo y “LILO crash rescue HOWTO“ tienen información sobre la configuración y el uso de LILO.
- El paquete grub-doc incluye toda la documentación sobre el uso de GRUB.
- Y naturalmente man init y man inittab tienen información sobre el proceso init y su archivo de configuración.
- El resto de comandos mencionados en este capítulo, tiene su correspondiente página de manual.

Capítulo 11. Gestión de Procesos

11.1. ¿Qué es un Proceso?

Es una aplicación que se ejecuta en un sistema.

O dicho de otra forma, es un fichero ejecutable que se ha puesto en la tabla de ficheros en ejecución. Por estar en esta tabla, el núcleo le concederá el uso de la CPU durante determinados periodos de tiempo, durante los cuales, se ejecutarán las instrucciones definidas en el fichero.

Los procesos consumen recursos de la máquina. No solo necesitan tiempo de CPU para ejecutar sus instrucciones, además necesitan espacio en memoria para albergar tanto su propio código como sus datos, utilizan los dispositivos, los ficheros...

11.2. Gestión de Procesos

Entendemos por gestionar no solo saber qué hace cada proceso en nuestro sistema y qué está consumiendo, sino también ser capaces de arrancarlos, detenerlos y pasarles parámetros según nuestras necesidades.

Toda la administración de procesos suele hacerse como root, si bien algunos procesos (de usuario) podrán lanzarse sin requerir permisos de administración.

11.2.1. Conocer los Procesos del Sistema

Lo primero para poder gestionar es conocer.

Saber qué procesos hay en funcionamiento en nuestro sistema es fácil basta con ejecutar el comando `ps` (process status), que nos informará de ello. La forma habitual es con el parámetro -A, para que nos muestre todos los procesos.

```
# ps -A
```

Este comando nos mostrará una línea por cada proceso en ejecución. Y de cada proceso nos indicará:

- PID: Un número que identifica unívocamente cada proceso.
- TTY: El nombre del terminal en el que se ejecuta el proceso (si es que se ejecuta en un terminal).
- TIME: El tiempo de CPU consumido por el proceso.
- CMD: El comando que lanzó el proceso.

Si queremos hacer un seguimiento de los procesos en ejecución, podemos utilizar el comando `top`, que nos mostrará como varían los datos de ejecución de los procesos.

Si queremos saber mucho más sobre los procesos, podemos curiosear el directorio `/proc`. Este es un directorio virtual (no contiene verdaderos archivos sino que lo crea el

núcleo). En él encontraremos un directorio para cada proceso en ejecución, identificado con su PID.

En el directorio de cada proceso encontraremos mucha información sobre él. Entre otros:

- cmdline: Contiene una copia del comando que lanzo el proceso.
- cwd: Es un enlace al directorio de trabajo del proceso.
- environ: Las variables de entorno del proceso.
- exe: Un puntero al binario que se ejecutó para lanzar el proceso.
- fd: Se trata de un directorio con una entrada por cada fichero que mantiene abierto el proceso.
- maps: Un mapa de las regiones de memoria asociadas al proceso y sus permisos.
- mem: La memoria a la que accede el proceso.
- mounts: Información sobre sistemas de ficheros montados.
- root: Raiz del sistema de ficheros para este proceso (ver man chroot).
- stat, statm, status: Información sobre el estado del proceso.

No dejes de consultar man proc para saber más.

11.2.2. Arrancar Procesos

Para arrancar un proceso basta con llamar a un fichero ejecutable. Conviene tener en cuenta que habitualmente, en una máquina GNU/Linux el directorio actual NO se encuentra en el PATH, por lo que la llamada directa a un fichero ejecutable del directorio actual no suele arrancar el proceso.

Por ejemplo, si quieres arrancar el proceso asociado al fichero miProceso del directorio en el que te encuentras, el comando Te dará el error:

```
> miProceso
bash: miProceso: command not found
```

Deberías hacer lo siguiente:

```
> ./miProceso
```

Esta curiosa definición del PATH se debe a motivos de seguridad, pues evita que se ejecuten troyanos con nombres habituales (¿que tal ls?) sin que el usuario lo note.

Puedes conocer el valor de la variable PATH con

```
> echo $PATH
```

11.2.3. Detener Procesos

Para detener un proceso se utilizan los comandos `kill` y `killall`.

kill identifica el proceso a detener por su PID (podemos conocer el PID mediante el comando ps). Así, para detener el proceso con PID=534 podemos hacer:

```
# kill 534
```

Este comando admite una señal para enviar al proceso. Podemos conocer todas las señales disponibles con `kill -l` y naturalmente, consultando la página de manual del comando.

La señal más utilizada es la 9 (KILL) que no puede ser bloqueada.

```
# kill -9 534
```

killall identifica el proceso a detener por su nombre. Así, para matar todos los procesos asociados al comando grep, podemos hacer:

```
# kill grep
```

Esto nos ahorrará consultar el PID de cada proceso (habitualmente, con el comando ps).

11.3. Demonios

A los procesos que proporcionan servicios generales de la máquina y que por lo tanto suelen estar siempre en funcionamiento, se les denomina demonios.

En un sistema Debian, todos los demonios suelen residir en el mismo directorio, en /etc/init.d/ (es decir, el infierno).

Los demonios se arrancan, detienen y gestionan como cualquier otro proceso, pero además, suelen permitir que se les pasen parámetros.

Por ejemplo, para conocer los parámetros que permite el demonio fetchmail (encargado de recoger el correo de buzones POP3), podemos hacer:

```
# /etc/init.d/fetchmail
```

Lo que nos indicará que este demonio acepta los parámetros start, stop, restart, awaken y debug-run.

Así que, aunque un demonio puede pararse con un comando kill, lo habitual suele ser pararlo de forma "poco dolorosa" con

```
# /etc/init.d/demonio stop
```

Y arrancarlo de la misma forma.

11.4. Bibliografía

- Páginas de Manual de ps, top, proc, kill y killall.

Capítulo 12. Compilación del Núcleo

12.1. Introducción

12.1.1. ¿Qué es el Núcleo?

El núcleo de un sistema operativo (kernel en inglés) es la aplicación que gestiona el acceso del resto de las aplicaciones a los recursos principales del sistema (CPU, memoria RAM, buses, memoria de intercambio...).

Dicho de otra forma, el corazón de nuestro sistema GNU/Linux.

12.1.2. ¿Para qué sirve?

Para lo mismo que tu corazón, sin él, no funciona nada. El núcleo carga las aplicaciones en memoria para que se ejecuten, les da paso a la CPU, las baja a memoria de intercambio cuando corresponde, les proporciona formas de acceder al hardware, gestiona los sistemas de ficheros, realiza las comunicaciones de red...

12.1.3. ¿Porqué querría yo recompilar mi núcleo?

Hay muchísimas razones. Las más habituales son:

1. Porque necesitas trabajar con un nuevo gadcheto-molón que no está soportado por tu núcleo actual.
2. Porque no sabes compilar el núcleo y no estás dispuesto a seguir siendo un "linuxero" de tercera.
3. Porque acabas de compilar el núcleo y ha fallado, así que tienes que compilarlo de nuevo (opción más habitual).
4. Porque no te crees el rollo del software libre y quieres "modificar" algunas líneas del código a ver qué pasa.
5. Porque no te quieres quedar fuera de las conversaciones sobre batallitas al compilar el núcleo de tus colegas.

12.2. Módulos

El núcleo de Linux es una única aplicación, que se corresponde con un único archivo ejecutable (generalmente denominado vmlinuz).

Sin embargo para que el núcleo pueda dar soporte a todo lo que puede llegar a necesitar un sistema, este archivo puede llegar a ser muy grande, y por lo tanto, consumir mucha memoria. No olvidemos que el núcleo debe estar siempre en memoria, y por lo tanto, ocupa parte de la memoria RAM disponible en el sistema.

Por ejemplo, si queremos que nuestro sistema pueda operar con unidades de disco flexible (floppies) y compilamos nuestro núcleo para ello (enseguida veremos como), nuestro

núcleo incluirá varios kbytes de código para este soporte. Este código estará ocupando memoria del sistema incluso cuando no estemos utilizando el disco flexible.

Para evitar este despilfarro existen los módulos. Estos son partes del código del núcleo que pueden compilarse como librerías dinámicas. Estas serán cargadas (subidas a memoria) sólo cuando sean requeridas por el sistema. El predominio de este tipo de módulos, hace que el núcleo sea más ligero y más flexible, por eso suele considerarse una buena idea.

Pero no todo está disponible como módulo. El proceso de compilación nos indicará que partes de la funcionalidad del núcleo pueden compilarse como módulos y que partes no. Lo que compilemos como módulos, se transformará en un fichero .o que podrá ser gestionado como módulo.

Tampoco es conveniente compilar todo lo posible como módulo, además de que algunas cosas pueden no funcionar (conviene consultar SIEMPRE la documentación de cada módulo), hay cosas que no podremos poner como módulo. Por ejemplo, el soporte para el sistema de ficheros en el que resida nuestra raíz[7].

12.3. Actualización

El primer objetivo es conseguir la última versión del núcleo de Linux (o la que queramos instalar). Hay dos formas de conseguirla.

12.3.1. Instalando el Paquete Debian

Para ello, basta con seleccionar el paquete de la versión del núcleo que queremos.

12.3.2. Bajando el Código de kernel.org

En ocasiones, la última versión del núcleo no está disponible en nuestra distribución. Podemos entonces conectar con www.kernel.org y bajarnos la versión que necesitamos. Elegiremos para ello el formato tar.gz.

El fichero correspondiente debemos guardarlo en /usr/src/ y descomprimirlo con el comando:

```
# tar xvjf kernel-source-2.4.24.tar.gz
```

A continuación, creamos un enlace al directorio

```
# ln -s kernel-2.4.24 linux
```

De esta forma, el código del núcleo activo siempre estará en `/usr/src/linux`.

7 Si lo piensas, esto es una obviedad (y frecuentemente la causa del famoso „Kernel Panic“). Si ponemos el soporte para ext2 (por ejemplo) como módulo, y ponemos el código del módulo en un sistema de ficheros ext2, ¿cómo diablos vamos a cargar el módulo para soporte ext2 si no tenemos soporte ext2?. Lo mismo se aplica al bus IDE, SCSI... o cualquier otra cosa involucrada en el soporte a nuestro sistema de ficheros raíz.

12.4. Si solo queremos parchear

Puede ocurrir que ya tengamos el código, y que solo queramos parchearlo.

Para esto, bastará con que nos bajemos el parche de www.kernel.org y lo copiemos en /usr/src/. Ojo al elegir la versión del parche, porque si nos hemos saltado alguna actualización, tendremos que bajarnos y aplicar sucesivamente cada uno de los parches que faltan entre nuestra versión actual de núcleo y aquella a la que queremos actualizarnos.

Para aplicar el parche tenemos que utilizar el comando desde el directorio /usr/src/linux.

```
# patch -p0 < fichero_parche
```

12.5. Compilación

Compilar el núcleo tiene seis pasos que deben realizarse de forma secuencial desde /usr/src/linux. Son:

make menuconfig

Este comando arranca una aplicación gráfica en la que podemos elegir qué opciones queremos incluir en nuestro módulo, y si estas se incluyen directamente en el núcleo o como módulo.

Para elaborar el menú de opciones que presenta la aplicación, el comando analiza el código fuente instalado en el directorio. Por este motivo, las opciones pueden variar en función de la versión del núcleo y los parches aplicados. Léete la documentación y la ayuda tanto de la aplicación como de las opciones que vayas a elegir o descartar.

Como estrategia, si es la primera vez que compilas el núcleo, te recomiendo que crees un núcleo de mínimos, con las mínimas opciones para que funcione. Compilar por primera vez el núcleo de una máquina es difícil y suele fallar (no arranca). Posteriormente será mucho más fácil ir añadiendo opciones y perfeccionando tu núcleo. Vamos, que la primera vez no escojas ninguna tarjeta de sonido, ni acelerador gráfico, ni opciones de video ... aunque los tengas.

Las decisiones que tomemos quedarán reflejadas en el fichero `.config` que como su propio nombre indica, es oculto. (¿Qué tal hacerle una copia de seguridad?).

Por cierto, para poder hacer esto, necesitarás tener instalado el paquete libncurses5-dev.

make dep

Este comando comprobará que en el directorio del código están todos los ficheros referidos en las cabeceras de compilación de las opciones elegidas (los famosos #include xxxx.h de C).

Verás que salen un montón de líneas. No hace falta que las mires todas. Si hay algún error, el proceso se detendrá.

make clean

Este comando limpiará el directorio de compilación. Es decir, eliminará los ficheros con extensión .o. Estos se corresponden con los módulos y con imágenes del núcleo que no hubieran concluido compilaciones anteriores.

make bzImage

Este comando compila el núcleo (por fin). Cuando concluya habrá generado la imagen del núcleo en `/usr/src/linux/arch/i386/boot/bzImage`.

make modules

Este comando compila los módulos.

Frecuentemente aparecerá la línea "nothing done for ...". No pasa nada, simplemente es que no hemos elegido ningún módulo de ese área (por ejemplo, soporte de comunicación por radio).

make modules_install

Este comando copiará los ficheros .o correspondientes a los módulos compilados al directorio que les corresponde (/lib/modules/2.4.24/...).

make install

Este comando copia la imagen del núcleo al directorio raíz, con el nombre vmlinuz. Previamente habrá renombrado la imagen actual como vmlinuz.old. Además, dejará en el directorio raíz una copia del .config, con el nombre config y el mapa del sistema (con el nombre System.map).

12.5.1. ¿Qué estamos haciendo?

El comando `make opción`, lo único que hace es ir al fichero Makefile del directorio correspondiente y ejecutar las instrucciones que figuran bajo la etiqueta `opción:`. Puedes consultar `man make` y ver el fichero Makefile de /usr/src/linux (y el resto de subdirectorios) para saber más.

12.5.2. Compilación con el Núcleo 2.6

Para el núcleo 2.6 se simplificó el proceso de compilación. Ahora, basta con realizar los comandos: make menuconfig, make all y make install.

12.6. Configuración

Hay que dejar el sistema preparado para el rearranque con el nuevo núcleo. Para ello, basta con actualizar el gestor de arranque (si es LILO, tendrás que ejecutar lilo).

Procura que el gestor de arranque, además de ser capaz de encontrar el nuevo núcleo, sea capaz de encontrar el antiguo (u otro de rescate). Solo por si acaso...

12.7. Diagnóstico

Para comprobar que tu nuevo núcleo funciona, haz lo siguiente:

1. Rearranca el sistema, eligiendo en tu gestor de arranque el nuevo núcleo.
2. Una vez inciado el sistema, ejecuta el comando

```
> uname -a
```

Con él podrás ver si efectivamente estás ejecutando el núcleo que acabas de compilar.

12.8. Vocabulario

- Compilar - El núcleo de Linux está programado en C. Para pasar del código fuente en modo texto, escrito en lenguaje C, a un código ejecutable binario que nuestra máquina pueda entender, es necesario "compilar el código". Con el proceso de Actualización descrito, el código fuente del núcleo se instala en /usr/src/linux (échale un vistazo y modificalo). Al compilarlo, realizamos los procesos habituales de compilación y linkado de programas C.
- Parchear - En ocasiones el equipo que desarrolla el núcleo de Linux publica modificaciones parciales. Estas resuelven pequeños problemas y errores encontrados. Para evitar tener que bajarse todo el código del núcleo por unas simples modificaciones, se publica lo que se denomina "un parche". El fichero de parche solo contiene las diferencias entre la versión anterior y la corregida, y ocupa mucho menos de los más de 27 MBytes del núcleo completo. Parchear no es más que aplicar un parche para actualizar rápidamente nuestro núcleo.

12.9. Bibliografía

- Documentación de configuración del núcleo. En /usr/src/linux/Documentation/
- “Kernel HOWTO“, generalmente en /usr/share/doc/HOWTO/en-txt/Kernel-HOWTO.gz.
- Páginas de Manual de patch (man patch).

- Para conocer los pormenores del núcleo, también puede consultarse “The Linux Kernel” de David A. Rusling. Puede encontrarse en www.tldp.org.

Capítulo 13. Configuración de Hardware

13.1. Introducción

Ya comentábamos en el primer capítulo, que el soporte para hardware era una de las desventajas de los sistemas GNU/Linux. Aunque esto va desapareciendo día a día, es innegable que basta con pinchar cualquier componente en un sistema Güindos para que este lo detecte de forma casi automática, lo configure y lo ponga a nuestra disposición. Esto, en GNU/Linux, no es así.

Cada vez es mayor el conjunto de componentes de hardware soportados. Y día a día, el trabajo de la comunidad de hackers, es mejor y da soporte a mayor cantidad de componentes y dispositivos. Al mismo tiempo, las distribuciones mejoran día a día sus sistemas de autodetección, autoinstalación y autoconfiguración. Pero queda mucho camino por recorrer.

Si bien es cierto que hoy, es difícil encontrar hardware no soportado en GNU/Linux, no lo es menos que la configuración de algunos dispositivos dista mucho de ser "sencilla". Por este motivo resulta imprescindible conocer la estructura que da soporte a nuestros dispositivos en GNU/Linux y establecer un procedimiento para instalar y configurar cualquier nuevo dispositivo.

Pero cuidado. El hecho de que establezcamos un procedimiento no significa que sea sencillo. Ni siquiera significa que vayamos a tener éxito en cualquier configuración que nos propongamos. Simplemente, se trata de una ayuda, una guía.

Por último, casi todos los casos de dispositivos no soportados por los sistemas GNU/Linux, se deben a que el fabricante se ha negado (como es su derecho) a publicar las características técnicas de sus interfaces. También es nuestro derecho señalarles y negarnos a comprar sus dispositivos, potenciando así una mayor apertura del mercado.

13.2. Cómo funciona esto

El primer de soporte de hardware es el propio núcleo. Ya vimos que el soporte para elementos de hardware pueden incluir como código propio del núcleo o como módulos que se cargarán cuando sean requeridos.

Hay que tener en cuenta que el código de estos módulos puede estar incluido en los fuentes de nuestra versión del núcleo, disponible como paquete adicional de nuestra distribución o, incluso, publicado en algún web de "dudosa" reputación.

Por lo tanto, el primer paso para configurar nuestro hardware consiste en darle el soporte necesario en el núcleo.

13.2.1. Soporte en el Núcleo

En primer lugar fíjate a qué bus o interfaz se conecta el nuevo dispositivo. Si es USB, por infrarrojos, FireWire, PCI, SCSI... habilita el soporte correspondiente en tu núcleo y recompila (esto es mucho más fácil decirlo que hacerlo).

Si es posible, ponlo como módulo para que podamos hacer más pruebas.

Algunos dispositivos se configuran directamente en el núcleo. Es decir, el código del núcleo puede llevar soporte para ellos. Este es el caso de las tarjetas de red, aceleradores gráficos, interfaces de vídeo, etc. Si el soporte está ahí, estamos de enhorabuena. Basta con activarlo y configurarlo.

13.2.2. Gestión de Módulos

Si ha sido posible darle soporte a nuestro dispositivo en el núcleo como módulo, debemos gestionarlo adecuadamente. La gestión de módulos es muy fácil.

Nota: Si no hemos podido configurarlo como módulo y lo hemos introducido directamente en el núcleo, el soporte para nuestro dispositivo estará siempre presente y no necesitaremos hacer nada de gestión de módulos al respecto.

La configuración de módulos suele hacerse con la herramienta gráfica modconf.

modconf

Esta herramienta nos permite ver la lista de módulos disponibles en nuestro sistema, agrupados por tipos (sonido, video, usb...). Podemos incluirlos en el núcleo simplemente seleccionándolos (es decir, ponerlos activos) o bajarlos.

modconf es una herramienta ideal para probar módulos. Si tienes duda de cual de los módulos soportados en el núcleo corresponde a tu dispositivo, siempre puedes compilarlos todos y probarlos con modconf. Ten en cuenta que el hecho de que un módulo se cargue perfectamente en el núcleo no significa que corresponda con el dispositivo que tenemos instalado. Prueba siempre el dispositivo.

Hay que tener en cuenta que si subimos un módulo con modconf, y no lo bajamos, este quedará incluido en el archivo /etc/modules y se cargará automáticamente cada vez que iniciemos la máquina.

Comandos Básicos

Como todos los interfaces gráficos, modconf lo único que hace es utilizar algunos comandos básicos para la gestión de módulos. Son:

- `lsmod` - Muestra los módulos cargados en el kernel.
- `insmod` - Sube un módulo al kernel.
- `modprobe` - Prueba a subir un módulo al kernel.
- `rmmode` - Quita un módulo del kernel.

13.3. Procedimiento para la configuración de hardware

- "¿Y si nuestro núcleo no tiene soporte para mi nuevo dispositivo?" -

- "Pues actualizate a la última versión del núcleo (www.kernel.org)." -

- "¿Y si sigo sin soporte?" -

- "Entonces, tienes un problema." -

Vamos a ver un procedimiento estándar que nos permitirá seguir una receta a la hora de instalar y configurar nuevo hardware en nuestro sistema. No garantiza que podamos instalar cualquier cosa, pero nos mantendrá entretenidos un ratito. Estos son los pasos:

1. Preguntar al fabricante. Lo primero es ver si el fabricante del hardware incluye instrucciones (en el paquete o en su web) para la configuración de su componente en Linux.
2. Buscar en Google. Es recomendable hacer una primera búsqueda en Google, con el nombre del componente. Mejor que acudir directamente a www.google.com es ir al buscador especializado en linux (www.google.com/linux). En función de los resultados podremos saber si la configuración será pan comido o trabajo de chinos. Podemos encontrar pistas y opiniones.
3. Buscar en nuestra distribución. Puede que nuestro sistema de paquetes incluya alguna aplicación relacionada con el componente. Para algunos dispositivos no es necesario soporte en el núcleo, basta con una aplicación de gestión (por ejemplo, xane para los escánners).
4. Buscar profundamente en Internet. Siempre es posible encontrar a alguien que ya ha configurado ese dispositivo en algún sistema parecido al nuestro. No conviene olvidar la consulta a Comunidades de Desarrollo, como www.sourceforge.net o www.savannah.org. Por ejemplo, Sourceforge alberga el proyecto pcmcia para GNU/Linux.
5. Desarrollar el módulo. Si no podemos encontrar a nadie que lo haya hecho antes, nuestras opciones son:

    a) Devolver el dispositivo y cambiarlo por otro soportado por Linux.

    b) Esperar a que alguien desarrolle el soporte.

    c) Desarrollar nosotros mismos el módulo.

13.4. Módems PCI (WinModems)

Se trata de módems de bajo coste. Parte de su funcionalidad "se la ceden" al sistema operativo, y hacen que nuestra CPU, además de su propio trabajo, tenga deberes del módem. Nos ahorramos algo de pasta con ellos, y por eso muchos fabricantes los incluyen en sus configuraciones. El problema es que están diseñados para Güindos.

Pero hoy en día, gracias a la comunidad de hackers, casi todos están soportados en mayor o menor medida (puede que no oigamos su altavoz, por ejemplo).

Estos módems (linmodems si existe soporte para ellos) suelen conectarse al bus PCI. La configuración de módems PCI tiene algún truquillo. Si el propio fabricante no nos da instrucciones, lo mejor es:

a) Sacar la tarjeta y mirar el nombre del chipset. O utilizar lspci

b) Conectarnos a www.linmodems.org y buscar las instrucciones para nuestro Winmodem, pero en función de su chipset, no de su fabricante.

13.5. Escánners

El soporte para escáners en GNU/Linux, se realiza a través del sistema xane. Basta con instalar este servicio en nuestra máquina e instalar los módulos de nuestro escáner (si es que están disponibles para xane).

13.6. Impresoras

El soporte para impresoras es mucho más sencillo. En primer lugar, casi todos los fabricantes incluyen soporte para GNU/Linux. La principal fuente de errores es que olvidamos dar soporte en el núcleo a nuestro interfaz de conexión con la impresora, ya sea el puerto paralelo, USB, etc.

Hay muchos paquetes de módulos de impresora en Debian (apt-cache search printer).

13.7. Bibliografía

- Sobre gestión de módulos, además de las páginas de manual para los comandos lsmod, insmod, modprobe y rmmod, puedes consultar Modules-HOWTO.
- Para saber algo sobre cómo programar tus propios módulos, puedes consultar "Programación en Linux", de la editorial Prentice Hall. Capítulo 31 "Controladores de Dispositivo". Incluye un ejemplo de desarrollo de un módulo para un motor paso a paso.
- Sobre la configuración y soporte de xane, puedes consultar www.freshmeat.net/projects/sane/.

Capítulo 14. Ofimática

14.1. Estructura del Servidor Gráfico

Al igual que hicimos con la estructura de red, es interesante conocer como se estructura el servidor gráfico en un sistema GNU/Linux. De esta forma, nos resultará más lógica su configuración.

En la presentación de gráficos (colores, líneas, movimientos...) de un sistema intervienen los siguientes elementos:

- Monitor: Es la pantalla. Puede ser un tubo de rayos catódicos (CRT) un display de cristal líquido (LCD) o ...
- Interfaz: Es la tarjeta de video. Contiene la memoria de idem donde se guarda la información de la imagen que tiene que presentar el monitor. Se encarga de enviar la información a éste.
- Módulo: Es el programa capaz de gestionar el interfaz. Se desarrolla a medida para cada tarjeta y sirve para integrarlo con el resto de aplicaciones.
- Servidor X: En GNU/Linux, el núclo no controla el servidor gráfico, lo hace el famoso servidor de las X. Se trata de un programa que da servicios gráficos a las aplicaciones. Es decir, cuando una aplicación quiere pintar una línea en la pantalla, se lo pide al servidor de las X. Es él quien controla los interfaces. Hay varios servidores de X, pero aquí solo hablaremos del libre, XFree86.
- Librería Gráfica: Son librerías que facilitan la gestión gráfica a las aplicaciones. Por ejemplo, en lugar de pedir varias líneas para configurar una ventana, una aplicación puede utilizar una librería gráfica y pedirle a ella directamente que le abra una ventana. En GNU/Linux hay dos muy conocidas, la QT y las GTK.
- Escritorio: Son aplicaciones que dan servicios al usuario. Le configuran la pantalla para que parezca un escritorio fácil de usar: ponen fondos, menús, incluyen aplicaciones de configuración, etc. En GNU/Linux hay dos que se llevan la palma: GNOME y KDE. Pero además, hay otros como Window Maker.
- Gestor de Ventanas: Son aplicaciones que se encargan de gestionar las ventanas (en las que se ejecuta cada aplicación). El gestor de ventanas se integra con el escritorio, y es quien sabe abrir ventanas, moverlas, redimensionarlas, etc. Como siempre, en GNU/Linux hay muchos gestores de ventanas, cada uno con sus propias características y posibilidades. Entre ellos, Sawfish, kwm...

14.2. Eligiendo Escritorio

Una de las decisiones más difíciles a las que se enfrentan los novatos de GNU/Linux es la elección del escritorio. "¿Debo escoger KDE que es más sencillo de usar? - ¿O debo decantarme por GNOME, que es más potente?".

¡¡¡ DESPIERTA !!! Esto es GNU, instálate los dos y pruébalos, es gratis.

14.3. Configuración[8]

La configuración del servicio gráfico se basa en la configuración del servidor de las X. Esta configuración es, cuanto menos, difícil. Naturalmente, se basa en la edición de un fichero de texto `/etc/X11/XF86Config-4`.

Este fichero está dividido en secciones, entre ellas:

- Server Layout. Permite designar configuraciones para poder cambiar entre ellas rápidamente.
- Files. Indica los ficheros relevantes para el servidor de las X (tipos de letra, principalmente).
- Module. Indica los módulos que puede cargar el sistema.
- Input Devices. Identificación de los dispositivos de entrada, generalmente, teclado, ratón,... Se incluye una de estas secciones por cada dispositivo.
- Monitor. Sección para la identificación de monitores, se indica su frecuencia de barrido horizontal y vertical.
- Device. Identificación de la tarjeta de video. Un sistema puede tener varias tarjetas de video, y tendrá una entrada por cada tarjeta.
- Screen. Se trata de una combinación de monitor y tarjeta. En esta sección se indican, por ejemplo, las posibles resoluciones pantalla.

La configuración de este fichero puede hacerse editándolo directamente. Aunque si bien este puede ser un mecanismo válido para pequeños cambios, resulta engorroso para la configuración incial. Hay varias aplicaciones gráficas que permiten configurar "las X". La más utilizada es

```
# Xf86cfg -textmode.
```

En algunos escritorios, (KDE, GNOME) su propio panel de control permite configurar las X de forma gráfica. (Claro, que esto no vale de nada si el servidor de las X no consigue arrancar).

8 Actualmente hay otro servidor X disponible: Xorg. Se trata de una implementación desde cero y está destinado a sustituir a XFree86. Su configuración es muy similar a la descrita para XFree.

14.4. Aplicaciones Ofimáticas

Afortunadamente, cada vez más los usuarios se van acostumbrando a realizar determinadas tareas con aplicaciones "estándar". Por ejemplo, los presupuestos se realizan con hojas de cálculo, las cartas con editores de textos... Es lo que se denomina monopolio de producto, frente al tradicional monopolio de marca.

Así, cualquier empresa o comunidad que desarrolle una hoja de cálculo (por ejemplo) sabe para qué la podrán utilizar sus usuarios.

En el mundo GNU/Linux hay un montón de aplicaciones para casi todo. Si bien es posible realizar documentos con un editor de texto plano como vim, la mayoría de usuarios preferirán un editor de textos gráfico, como OpenOffice.

La siguiente tabla muestra la correspondencia entre las aplicaciones más utilizadas y los programas más utilizados en GNU/Linux.

- Suite Ofimática: Hay varias, la más utilizada es OpenOffice, pero también está KOffice (de KDE) o Corel. Todas son ampliamente compatibles con Microsoft Office.
- Editor de Textos: Naturalmente todas las suites anteriores incluyen editores de texto. Pero también aplicaciones (no incluidas en suites) que realizan el trabajo. Una de las más utilizada es AbiWord. Naturalmente, también es compatible con MS.
- Hoja de Cálculo: Ocurre lo mismo que con los editores de texto: KOffice tiene su hoja de cálculo, OpenOffice también... Y de nuevo, hay programas no incluidos en suites que realizan el trabajo. El más utilizado como hoja de cálculo es GNUmeric. (Si bien vim dispone de un script con funcionalidad de hoja de cálculo que no está nada mal :-))
- Presentaciones: También incluida habitualmente en las suites. Fuera de ellas, está MagicPoint.
- Navegador: OpenOffice incluye el suyo propio. Pero además está Mozilla (motor libre de Netscape), Galeón, Ópera, Konqueror (incluido en KDE)... Y el tradicional navegador en modo texto: links.
- Correo Electrónico: Fuera de las suites, KMail (de KDE), Netscape Messenger, Ximian Evolution (que además incluye agenda, tareas ... vamos, como MS Outlook), Sylpheed... o el tradicional de modo texto, mutt.
- IRC-Chat: Puedes utilizar X-Chat, Gaim, Jabber o KMerlin.
- Gráficos: Para gráficos en modo raster, el más utilizado es gimp. Para vectoriales, sodipodi.
- Reproductor de Audio: Por ejemplo, xmms.
- Rippeo de Cds: Jack the Ripper, grip.
- Visualización de Video: xine.
- Grabación de CDs: Nero.
- Administración de Sistemas: Tanto el Centro de Control de Gnome (gnomecc) como el Centrol de Control de KDE (KControl) incluyen casi toda la funcionalidad necesaria

para la administración del sistema. Puede completarse con aplicaciones como KUser (para administración de usuarios).

Todos ellos son programas de primera línea, totalmente profesionales e incluyen todas las funciones y características necesarias para su uso. Pueden encontrarse como paquetes Debian para su instalación.

Capítulo 15. Seguridad

15.1. Introducción

La Seguridad es un tema abierto y siempre incompleto. Constantemente aparecen nuevos agujeros de seguridad, nuevos tipos de ataques e incluso nuevos daños. Y con ellos, nuevas formas de prevenir ataques, de limitar daños, de resolver errores... Hay quien afirma que la Seguridad es "un estado de ánimo" subrayando que esto, no es una ciencia cierta.

Así que aquí, nos vamos a limitar a ver una introducción a varios de los aspectos de seguridad en nuestros sistemas GNU/Linux. No es completo, ni puede serlo, pero sí representativo.

15.1.1. ¿Qué es la Seguridad?

Podemos definir un ordenador seguro como aquel que hace todo lo que esperamos que haga. Sí, esta definición incluye mucho de configuración, pero es que en seguridad, casi todo es configuración.

Conviene no olvidar algo obvio: Un ordenador NUNCA será 100% seguro. Así que la inversión en Seguridad (elementos, coste, tiempo...) debe ser proporcional al interés de lo que estamos protegiendo y a nuestros objetivos de seguridad.

15.1.2. Tipos de Daños

Con la definición anterior, un ordenador que sufre una corrupción de disco por un fallo de corriente ES un ordenador inseguro. Y ciertamente lo es. Son mucho más frecuentes los daños por este tipo de "accidentes" que los ataques de peligrosos crakers desde ex-repúblicas soviéticas en busca de números de tarjetas de crédito. Aunque hay que reconocer, que en las películas quedan mejor estos últimos.

Por falta de seguridad, un ordenador puede sufrir dos tipos de daños: de información o de servicio.

A su vez, los daños pueden ser de pérdida, de alteración o de acceso.

Y podemos seguir clasificándolos. Por ejemplo, por su intencionalidad: accidentales, inintencionados o maliciosos. O por su objetivo: ficheros, paquetes de red, servicios...

Pongamos algunos ejemplos:

- Un corte de luz, causa la pérdida de datos del disco.
- Un usuario borra accidentalmente un fichero.
- Un ex-empleado altera la página web corporativa.
- Un craker consigue leer el tráfico que circula por la red.
- Un administrador descuidado detiene el servicio de copias de seguridad.

- Un troyano modifica el funcionamiento de nuestro gestor de correo.
- Un grupo pacifista lanza un ataque masivo que colapsa nuestro servicio de acceso remoto.
- Un ladrón consigue la clave para realizar transacciones bancarias.

Algunos ataques causarán daños irreparables, otros serán fácilmente recuperables y otros, los peores, no causarán ningún daño aparente. No hay peor ataque que el que pasa desapercibido.

15.1.3. Tipos de Ataques

Cuando hablamos de ataque, hablamos de intención.

Al igual que en los daños, existen muchos criterios para clasificar los ataques. Pero eso no es lo importante. Aunque nos dejemos muchos, vamos a enunciar algunos de los más conocidos.

- Escaneo de Puertos: Consiste en acceder a los puertos de un servidor. En función de su respuesta es posible saber qué servicios tiene activos y preparar un ataque específico.
- Troyanos: Son programas aparentemente inofensivos (salvapantallas, chistes...) que instalamos en nuestro ordenador voluntariamente (lo suele mandar el amigo cachondo por correo electrónico). Pero en realidad, al tomar el control, además de la funcionalidad esperada realizan alguna otra actividad "imprevista".
- Por Fuerza Bruta: Son los ataques contra servicios o ficheros de contraseña. Un ataque "por fuerza bruta" significa que se envían combinaciones secuenciales o palabras de diccionario para encontrar contraseñas. Si no hay un retardo entre reintentos o el administrador no se da cuenta, se pueden probar varios millones de combinaciones en muy poco tiempo.
- Denegación de Servicio: Consiste en saturar un servicio con peticiones falsas para que le resulte imposible atender a sus verdaderos usuarios (Está muy de moda).
- Exploits de Servidores: Consiste en aprovechar errores de programa para generar el caos o acceder a sistemas.
- Buffer Overflow: Consiste en forzar un desbordamiento de un buffer de programa (generalmente aprovechando errores de programación). Esto genera un error en el programa y puede permitir ejecutar código malicioso. Es un caso particular de exploit.
- Sniffers: Consiste en leer los paquetes que circulan por la red. A veces, ni siquiera es necesario descifrarlos. Para algunos sniffers puede ser suficiente con saber que el tráfico entre dos puntos es más intenso de lo habitual.
- Virus: Esto es algo casi inexistente en GNU/Linux. En contra de lo que ocurre con otros sistemas operativos, al poder establecer permisos de acceso a los recursos, los virus tienen muy limitado su ámbito de actuación. Salvo si el propio administrador los ejecuta (inconscientemente). Por eso hay una máxima muy extendida en GNU/Linux No usarás el nombre de root en vano

- Spoofing: Consiste en suplantar la identidad de una máquina, por ejemplo, enviando paquetes TCP con una dirección IP falsa.
- RST: Es un tipo de spoofing. Si se monitoriza adecuadamente una conexión, resulta posible enviar un paquete TCP con el flag de RESET, interrumpiendo así la conexión.
- Undelete: Es... buscar en la basura. Consiste en acceder a ficheros que están teóricamente borrados para recuperarlos y leer su contenido.

Hay muchísimos tipos de ataque. Y seguro que mientras te lees esto, acaban de inventar otro. Así que conviene mantenerse al día.

15.1.4. Fuentes de Información

Hay muchos sitios con información sobre Seguridad. Si quieres tener sistemas realmente seguros, conviene que te apuntes a alguno de ellos.

Por ejemplo, Debian tiene una lista de seguridad debian-security@lists.debian.org a la que conviene estar subscrito.

También hay webs especializados, como www.linuxsecurity.com con cantidad de información y documentación.

15.2. Seguridad Física

No estaría mal que, tras invertir millones en asegurar puertos, poner firewalls, monitorizar logs, actualizar contraseñas, etc. para evitar un ataque externo, llegara alguien y, simplemente, cogiera el ordenador (o su disco) y se lo llevara a su casa (o lo apagara, quitara el cable, lo incendiara...)

La primera seguridad es la física, y sin ella, el resto, no tienen sentido.

Empecemos por la seguridad en el propio proceso de instalación.

15.2.1. Seguridad Física en la Instalación

En la instalación, conviene tener en cuenta algunos aspectos:

- Diseño de Particiones. No queremos que alguien se dedique, por ejemplo, a enviar grandes correos electrónicos y sature nuestra capacidad de disco. Conviene que algunos directorios correspondan con particiones o discos distintos de los del sistema base. Incluso aquellas partes del sistema que no van a varias, podrían estar en un medio de solo lectura.
- Elige buenos sistemas de ficheros. Pon ext3 y evitarás pérdidas de datos.
- No te conectes a la Red. Puede que alguien sepa que vas a instalar y te esté esperando. Durante la instalación tu ordenador no está preparado para repeler ni detectar un ataque. Eres vulnerable.
- Pon una BUENA contraseña a root. La contraseña de root es la base de la seguridad de tu sistema. Pon un poco de trabajo en ella.
- Activa shadow y md5. Es básico y permite contraseñas más largas.

- Ejecuta solo los servicios necesarios. Cada servicio es un agujero en potencia.
- Instala solo los paquetes necesarios. Cada paquete es un agujero en potencia.
- Lee la lista de correo de seguridad de Debian. Hay que estar informado.

15.2.2. Seguridad Física después de la Instalación

Como aspectos de la Seguridad Física, podemos indicar los siguentes:

- Restringir el acceso físico al ordenador y limitarlo exclusivamente a personas con autorización (No, el personal de limpieza tampoco).
- Activar la contraseña de la BIOS. Sin ella, cualquiera podría llegar, poner un floppy, cambiar la secuencia de inicio y tener el control total de la máquina.
- Configurar la BIOS. Establece el orden de arranque y NO permitas el arranque por floppy. Si lo necesitas, podrás activarlo con la contraseña de BIOS.
- Activar la contraseña del gestor de arranque. Tampoco queremos que nadie arranque opciones que no debe arrancar.
- Realiza copias de seguridad. Utiliza para ello los comandos tar y cron.

15.3. Seguridad Interna

La segunda fuente de problemas de seguridad son los propios usuarios del sistema. Estos pueden cometer errores y tener despistes que o bien causen problemas de seguridad o bien permitan ataques externos. Pero también tenemos que tener en cuenta que nuestros usuarios (con quienes tomamos café) son atacantes en pontencia (es decir, malintencionados). Por si no te habías dado cuenta, todo responsable de seguridad está un poco paranoico.

¿Qué tenemos que hacer para mejorar la seguiridad interna de nuestro sistema?

- Establece una política de permisos. Define los usuarios y sus grupos de forma segura y establece un umask (permisos por defecto).
- Evita el login de root. Se puede limitar en qué consolas puede hacerse login con root. Basta con editar los ficheros /etc/securetty y /etc/login.defs. También pueden configurarse otras opciones de login en /etc/pam.d/login y en /etc/security/access.conf.
- Evita el reboot desde consola. ¿No quieres que cualquier usuario tire el sistema? pues quítalo de /etc/inittab.
- Define adecuadamente el proceso de login de los usuarios. Para ello, tienes que configurar PAM (Pluggable Authentication Modules) (/etc/pam.d). En él, se pueden establecer los controles de contraseñas, métodos de autenticación, etc. También conviene revisar /etc/login.defs
- Monta las particiones correctamente. Por ejemplo, no permitas que se ejecuten programas en el directorio /tmp, y evitarás que un usuario ejecute un troyano sin darse cuenta.

- Comprueba las contraseñas. Además es divertido. Intenta romper la contraseña de alguno de tus usuarios. Puedes utilizar programas como crak o jack.
- Desconecta a los usuarios "ociosos". Un terminal abandonado (para ir a comer, al café, etc.) es un agujero de seguridad. Puedes configurar el bash o el salvapantallas del escritorio para que, tras un tiempo, requieran una contraseña o incluso desconecten al usuario.
- Establece cuotas de disco. Puedes establecerlas por usuarios o por grupos. Para ello, tendrás que habilitar el soporte en el núcleo e instalar el paquete "quota". Luego bastará con añadir, en el fichero /etc/fstab opciones de usrquota y/o grpquota. Además, hay que crear archivos quota.group y quota.user en las raices de los sistemas de ficheros.
- Usa los permisos especiales. Si, todavía hay más permisos especiales. Se denominan inmutable y append. Con ellos, se puede impedir completamente la modificación de ficheros o directorios, incluso para el superusuario (ver lsattr y chattr).
- Cambia locate por slocate. Así evitarás que nadie localice ficheros para los que no tiene permiso de lectura.
- Comprueba periódicamente las firmas md5. Esto garantiza que los archivos no han sido alterados. Puedes utilizar sXid, AIDE...

15.4. Seguridad Externa

Al conectar un ordenador a la red, estamos abriendo la posibilidad a nuevos ataques. Si esta conexión es a Internet, aunque utilicemos firewalls, routers, proxies y ristras de ajos, las posibilidades de un ataque crecen a lo bestia.

15.4.1. ¿Qué podemos hacer?

- Quita telnet. ¡¡Pero YA!!. Usa solo protocolos encriptados para acceder al ordenador, como ssh. Y mejora la seguridad de ssh (/etc/ssh/sshd_config). No permitas conexiones como root, obliga a utilizar su o sudo.
- Configura host.allow y host.deny
- Parchea el kernel. Esta es la mejor forma de evitar ataques conocidos contra errores de software (como los de buffer overflow).
- Usa chroot. Esto permite que los servicios de red se ejecuten en un "entorno controlado" sin que puedan causar daños fuera de él. También conviene que no se ejecuten con permisos de root.
- Mejor LDAP que NIS. Puedes poner un servidor LDAP y configurar PAM para que lo utilice en la autentificación.
- Desconecta el servicio RPC. Si puedes, algunos servicios, como NIS y NFS lo necesitan.
- Pon un firewall. Puede ser externo o incluso local a tu propio sistema.

15.5. Otros Aspectos de la Seguridad

Hay más cosas, estas son algunas recomendaciones:

- Revisa los logs. Aunque consigas un sistema seguro al 99%, si no tienes una buena política de avisos, no te enterarás de ese 1%. Debian tiene aplicaciones para la revisión de logs (swatch, logcheck...). También es conveniente que dirijas la salida del fichero de log a una consola virtual y así, poder monitorizarla (/etc/syslog.conf).
- Instala las actualizaciones de seguridad. Tan pronto como las recibas.
- Subscríbete a la lista de Seguridad.

15.6. Bibliografía

- Hay un montón de información sobre seguridad en www.linuxsecurity.com. En concreto, es muy recomendable el documento Securing Debian Manual.
- Sobre análisis de logs, puedes consultar www.counterpane.com.

Capítulo 16. Firewall

16.1. Introducción

16.1.1. ¿Qué es un firewall?

Los firewall son sistemas de seguridad, y constituyen la primera línea defensiva de nuestro sistema. La ventaja es que los sistemas GNU/Linux están especialmente capacitados para implementar sistemas de firewall, tanto indivuduales como para redes (bridge/firewalls), al disponer de soporte TCP/IP directamente en el núcleo.

El firewall nos permitirá defender nuestro sistema de ataques externos gracias a su capacidad de filtrado de paquetes.

16.1.2. ¿Qué es el filtrado de paquetes?

Es una técnica que permite decidir, en base a un conjunto de reglas denominadas cadenas que paquetes TCP/IP serán procesados por nuestro sistemas y cuales serán rechazados. De esta forma, podremos rechazar paquetes que consideremos peligrosos y evitar que desde el exterior se acceda a servicios que son privados.

En GNU/Linux, el filtrado de paquetes lo realiza directamente el núcleo. Por lo tanto, necesitamos saber qué camino recorren los paquetes al llegar a nuestro sistema.

16.1.3. ¿Cómo gestiona el núcleo los paquetes?

Cuando un paquete TCP llega a nuestro sistema, puede provenir de dos orígenes diferentes. Puede llegar a través de uno de los interfaces de red existentes en el sistema o puede haber sido generado en el propio sistema. Vamos con los primeros.

Lo primero que hace el núcleo con un paquete entrante desde un interfaz de red es decidir sobre su enrutado. Esta decisión se toma en base a las reglas de routing que estén definidas, y en ellas se especifica si el paquete debe ser procesado por nuestro sistema, o enrutado por alguno de nuestros interfaces de red hacia su destino.

Si la decisión de enrutado determina que el paquete debe ser procesado por el sistema, pasa a una cola denominada INPUT. Si se encamina hacia otro destino, se pasa a una cola denominada FORWARD. Y, por último, los paquetes generados en nuestro sistema, se pasan para su envío a una cola denominada OUTPUT.

Las reglas de firewall se aplican sobre cada una de estas colas. Al conjunto de reglas que se aplica a cada cola se le denomina cadena. Por eso se habla de la cadena de INPUT, por ejemplo.

16.2. Las tablas

El sistema de firewall que vamos a ver permite especificar cadenas (conjuntos de reglas) para tres tablas. Aunque solo veremos una de ellas. Las tablas son:

- filter. Es la tabla por defecto, y la que veremos. Contiene las cadenas para INPUT, OUTPUT y FORWARD, tal y como hemos definido.
- nat. Es la tabla que se consulta para paquetes que crean nuevas conexiones.
- mangle. Es una tabla especial.

16.3. Instalación

Veamos en primer lugar cómo se instala un firewall local. Es decir, un firewall que da protección a nuestra máquina.

16.3.1. Compilación del Núcleo

En primer lugar, debes asegurarte de que el núcleo del sistema se ha compilado con las siguientes opciones en Networking Options:

* Packet Socket.

* Network Packet Filtering.

* Socket Filtering.

* Unix Domain Sockets.

* TCP/IP Networking.

* IP Netfilter Configuration, con las siguientes opciones

    o IP tables support.

    o Packet Filtering.

16.3.2. Bridge Firewall

Un bridge firewall es una combinación de bridge y firewall. Por un lado, al funcionar como bridge interconecta dos redes Ethernet como si fueran una sola. Pero al trabajar también como firewall, nos permite definir las reglas de filtrado de paquetes, para evitar determinados tráficos entre ambas.

El uso más habitual de un bridge firewall es para incrementar la seguridad exterior de una red de área local. En esta configuración, la conexión a Internet está en uno de los interfaces del firewall, y el otro, se conecta a la red de área local.

Con una correcta política de reglas, podemos evitar que tráfico no deseado alcance a los ordenadores de la red de área local desde Internet. Por ejemplo, evitando que entre en la LAN paquetes destinados al protocolo ssh o telnet. Al mismo tiempo, también podemos restringir los paquetes salientes de nuestro sistema, evitando por ejemplo, que desde la LAN se envíen paquetes a puertos asociados a aplicaciones como emule o edonkey.

Para configurar un bridge firewall, además de la configuración de firewall, debemos

configurar nuestro sistema como un bridge. Para ello, bastará con:

1. Compilar el núcleo con las siguientes opciones:
   * Network Packet Filtering
   * 802.1d Ethernet Bridging
   * IP Netfilter Configuration (con las opciones necesarias)
2. Instalar el paquete bridge-utils (con apt-get install bridge-utils)
3. Configurar el bridge, por ejemplo, con los siguientes comandos:
   1. brctl addbr br0 # Crea un interfaz de bridge
   2. brctl addif br0 eth0 # Añade el interfaz eth0 al bridge 0
   3. brctl addif br0 eth1 # Añade el interfaz eth1 al bridge 0
   4. ifconfig eth0 0.0.0.0 # Arranca el interfaz eth0
   5. ifconfig eth1 0.0.0.0 # Arranca el interfaz eth1
   6. ifconfig br0 [IP] netmask broadcast # Arranca el interfaz br0 con los parámetros de IP, netmask y broadcast indicados.

16.4. Configuración de Firewalls

Configurar un firewall consisten en especificar las reglas que se aplicarán para cada tabla y para cada cadena de cada tabla.

Hasta la versión 2.2 del núcleo, la gestión del sistema de firewall se hacía mediante ipchains. Desde la versión 2.4 se realiza con iptables, que es el único sistema que veremos aquí.

Las reglas de firewall de nuestro sistema se gestionan con el comando iptables que nos va a permitir consultarlas, definirlas, eliminarlas y agruparlas.

16.4.1. Consulta

Podemos consultar las reglas activas en nuestro sistema con:

```
iptables -t filter -L
```

Con este comando, visualizaremos las reglas que se aplican a cada una de las cadenas de la tabla filter.

16.4.2. Definición

Para definir una nueva regla, basta con indicar, mediante el parámetro -A a qué cadena la vamos a añadir. Por ejemplo:

```
iptables -t filter -A INPUT -s 10.100.5.220 -j DROP
```

Esta regla indica que, a todos los paquetes provinientes de la dirección 10.100.5.220 se les aplicará la acción DROP. Vamos, que no admitiremos ninguno.

Existen parámetros para definir reglas en base a muchos conceptos. Entre ellos:

* -s (source) - Indica la dirección IP origen del paquete.

* -d (destiny) - Indica la dirección IP destino del paquete.

* -p (protocol) - Indica el protocolo asociado al paquete.

* -dport - Indica el puerto destino del paquete.

* -i (interface) - Indica el interfaz asociado al paquete.

* -f (fragment) - Indica si es un fragmento de paquete.

Utilizando dos puntos (:) se pueden definir rangos.

```
iptables –t filter –A OUTPUT –d 0.0.0.0 –p tcp –dport 6600:6999 –j DROP # Para evitar cualquier conexión a emule, edonkey...
```

Mediante el parámetro -j se especifica la acción que se realizará cuando el paquete coincida, que puede ser DROP (desechar), REJECT (rechazar, con un mensaje de error), ACCEPT (aceptar) o especificar otra cadena de reglas que se debe aplicar al paquete.

16.4.3. Eliminación

Las reglas se eliminan con -D.

16.4.4. Administración

Agrupación de Reglas

Administrar la reglas de una en una puede resultar demasiado complejo en sistemas grandes. Por eso, con iptables es posible agrupar las reglas y definir así nuevas cadenas. Estas cadenas se aplicarán cuando lo determine otra reglas, generalmente, de las cadenas por defecto.

```
iptables –t filter –N SPAM # Crea una nueva cadena SPAM
```

```
iptables –t filter –A SPAM –s IP_del_Spammer1 –j DROP # Añadimos una regla
```

```
iptables –t filter –A SPAM –s IP_del_Spammer2 –j DROP # Añadimos otra regla
```

```
iptables –t filter –A INPUT –s 0.0.0.0 –p tcp –dport 25 –j SPAM # Aplicamos la cadena SPAM a todos los paquetes que vengan por correo electrónico SMTP.
```

Registro de Reglas

El firewall es un elemento de seguridad, y en seguridad nada hay más importante que tener los ojos abiertos. Para eso, necesitamos registrar qué acciones está tomando nuestro firewall. Un registro bien

configurado nos permitirá detectar no solo ataques, sino también errores en nuestra configuración.

Para ello, basta con añadir la acción LOG, que enviará un mensaje al registro del sistema. Esta regla no detiene la ejecución del resto de reglas.

```
iptables -t filter -A INPUT -p tcp -dport 21 -j LOG --log-level 2 --log-prefix "Intento de Acceso a telnet"
```

Almacenamiento

Por último, es necesario que podamos guardar las reglas y no tener que definirlas de una en una. Para eso, existe un demonio, iptables que sabe hacerlo.

```
/etc/init.d/iptables save_active
```

Las reglas se almacenan en /var/lib/iptables y se ejecutan en el arranque del sistema.

Aplicaciones de Administración

Naturalmente no todo hay que hacerlo a mano (en consola). Existen aplicaciones gráficas que te ayudarán a gestionar tu firewall. Pero... si tu firewall es un sistema de seguridad y le pones más elementos (soporte gráfico, gestores de reglas...) ¿no crees que estás creando más puntos potenciales de ataque?

En fin, estos son:

- Para firewalls individuales: knetfilter, gfcc
- Para bridge firewalls y cosas más serias: bastille, mason

Capítulo 17. LDAP

17.1. Introducción

En este tema, nos limitaremos a ver cómo instalar y configurar un servidor LDAP para que ofrezca servicios de autentificación y sus clientes. LDAP puede utilizarse para muchísimo más.

LDAP fue inicialmente desarrollado por Sun Microsystems (no hay que restarles méritos). Para este caso, utilizaremos su versión GNU, llamada OpenLDAP.

17.1.1. ¿Qué es LDAP?

LDAP son las siglas de "Light Directory Access Protocol" (Protocolo de Acceso a Directorios Ligeros). Se trata de un protocolo cliente/servidor para acceder a un servicio de directorio.

- "Vale, pero ¿qué es un servicio de directorio?" -

> Pues como una base de datos, pero optimizada para consultas. Como sabes, el punto fuerte de las bases de datos es que permiten el acceso concurrente a la información manteniendo su integridad. Sin embargo, en muchos casos, la integridad no está comprometida, sobre todo, si predominan las operaciones de consulta. Para estos casos, en lugar de una base de datos, se puede utilizar un servicio de directorio, que resulta mucho más sencillo.

Los servicios de directorio almacenan información descriptiva, generalmente basada en atributos (no en relaciones) y están optimizados para responder rápidamente a un elevado volumen de consultas.

En muchos casos (y LDAP) es uno de ellos, pueden configurarse servidores replicados (esclavos) y resulta muy conveniente. Los "replicantes" no solo mejoran los tiempos de acceso, sino que constituyen un nivel más de redundancia ante fallos, incrementando así la seguridad del sistema.

Mejor lo repito. Si vas a autentificar a tus usuarios utilizando LDAP (como es el caso de esta documentación) ¡ pon un servidor replicado !. ¿Porqué?, porque así, si tienes un error de hardware (o de lo que sea) en el principal, no dejarás a todos tus usuarios sin acceso a sus sistemas. Estás avisado.

17.1.2. ¿Cómo funciona LDAP?

Fácil, esto es cliente/servidor. Un cliente se conecta con un servidor LDAP y le hace una pregunta. Este responde con la respuesta de su "base de datos" o un enlace donde buscar más información (normalmente, otro servidor LDAP). Cualquiera que sea el servidor al que un cliente se conecte, siempre verá la misma respuesta a la misma pregunta. Para eso es un servicio global de directorio.

Esta arquitectura tan sencilla, puede emplearse para muchas cosas: listas de empleados,

callejeros, ... Pero si vas a enviar contraseñas por la red, conviene que lo hagas de forma encriptada. (¿Lo repito?). Por ejemplo, con ssh.

17.2. Arquitectura

En un sistema como el descrito, entran en juego varios elementos:

17.2.1. Servidor

El servidor tiene en ejecución dos demonios: slapd, que responde las preguntas de los clientes y slurpd, que se encarga de sincronizar cada replicador con el servidor central. Adeás, requiere de:

* Base de Datos - Puede utilizar o la base de datos de Berkeley (libdb2) (recomendado) o la base de datos GNU (GDBM).

* Hilos - Es decir, soporte para ejecución multithread.

* TCP Wrappers - Para incrementar la seguridad en las conexiones.

17.2.2. Cliente

Solo necesita un programa LDAP-compatible, por ejemplo, para autentificarse.

17.2.3. Transporte

Entre ambos, es necesario lo siguiente:

- SSL (Secure Sockets Layers) - Para garantizar la confidencialidad de la comunicación. Sobre todo, que las contraseñas no viajen en claro por la red.
- Servicio de Autentificación de Kerberos (recomendado) - Sistema de autentificación de usuarios y servicios en red. Requiere un tercer sistema de confianza, el servidor de Kerberos.
- SASL (Simple Authentication and Security Layer) - Es un método para añadir soporte de autenticación a protocolos orientados a conexión (RFC 2222).

17.3. Servidor de LDAP

17.3.1. Instalación

Para instalar un servidor LDAP con Debian, basta con instalar el paquete slapd.

```
# apt-get install slapd
```

17.3.2. Configuración

Un servidor LDAP puede almacenar tres tipos de bases de datos (backends):

* LDBM - La habitual.

* SHELL - Para comandos de Unix.

* PASSWD - Para contraseñas.

En este documento, solo analizaremos la configuración de bases de datos del tipo LDBM.

Toda la configuración de un servidor LDAP se realiza en el fichero `/etc/ldap/slapd.conf`. Este fichero se divide en tres secciones:

- Global - Establece valores que afectan a todas las bases de datos, pero que pueden ser sobreescritos por valores particulares especificados para un backend o para una base de datos concreta. Entre los posibles parámetros de esta sección están:
    - include - Inclusión de esquemas, que llevan la definición de atributos y de clases de objetos.
    - schemacheck - Indica si los esquemas se comprobarán en la introducción de datos.
    - pidfile - Nombre del fichero con el PID del proceso del servidor.
    - argsfile - Fichero con los argumentos que se pasarán al arrancar el servidor.
    - replog - El fichero de registro de las replicaciones.
    - loglevel - Nivel de registro de incidencias.
- Backend - Establece valores que se aplicarán a cada tipo de base de datos. Tiene un parámetro característico: backend, que indica el tipo de base de datos (ldbm, shell, passwd o sql).
- Base de Datos - Establece los valores para cada base de datos. Admite los siguentes parámetros:
    - database - Inicia la sección e indica el tipo de base de datos.
    - suffix - Indica la base del directorio, a partir de la cual, se aplica la base de datos de la sección.
    - lastmod - Indica si el servidor guardará la información de la última modificación de cada entrada de la base de datos.
    - readonly - Pues eso. Indica si la base de datos es sólo de lectura.
    - replica - Indica la máquina que actuará como replicador de esta base de datos.
    - replogfile - Nombre del fichero de registro de replicación.
    - rootdn - Indica el nombre distinguido que no estará sujeto al control de acceso en esta base de datos..
    - rootpw - Contraseña del anterior.
    - updatedn - En un replicador, indica qué nombre distinguido tendrá acceso para modificar la réplica (habitualmente, slurpd).

  Para las bases de datos LDBM, además, están los siguientes parámetros:
    - cachesize - Número de entradas que el servidor mantendrá en la caché.
    - dbcachesize - Tamaño en bytes de la caché asociada a cada índice.

- directory - Directorio que contiene los archivos de datos.
- index - Índices que mantendrá el servidor para determinados atributos.
- mode - Indica la protección que tendrá cada fichero de índice al crearse.

17.3.3. Control de Acceso

El control de acceso puede especificarse de forma global (en la sección Global), para un backend o para cada base de datos. También puede ponerse en un fichero externo (y cargarse con include) o directamente en el slapd.conf.

Tiene las siguiente sintaxis:

```
access to [algo] by [quien] [nivel de acceso] defaultaccess read
```

Por ejemplo,

```
access to dn=cajabadajoz by * search
```

O particularizando para algún atributo

```
access to attribute=userPassword
      by dn="cn=admin,dc=cajabadajoz" write
      by self write
      by * none
```

17.4. Cliente de LDAP

17.4.1. Instalación

Para instalar un cliente LDAP, hay que instalar el paquete, ldap-utils.

17.4.2. Configuración

La configuración del cliente se realiza en el fichero /etc/ldap/ldap.conf.

En este fichero, entre otras cosas, se indica la base del árbolo de directorios con la que actuará el cliente (si no se indica nada en contra en el comando o aplicación) y la localización del servidor (o replicador) de LDAP.

Capítulo 18. Uso

El intercambio de información entre el cliente y el servidor se hace en un formato denominado LDIF.

Por ejemplo, un fichero con el contenido siguiente correspondería con una entrada de la base de datos en formato LDIF:

```
dn: cn=admin, dc=ilkebenson, dc=com
objectClass: person
cn: admin
sn: Administrador
userPassword: {SSHA}xCR3mqw/XupMWMrf59hPRS61fBnEuDxy
description: El administrador omnipotente del sistema
```

Un fichero LDIF puede contener varios registros, separados por una línea en blanco. El formato de cada registro depende del diseño del árbol que se realice.

Comandos en línea

Para gestionar la base de datos LDAP desde el cliente disponemos de los siguentes comandos:

- ldapsearch - Especifica condiciones para realizar búsquedas en el servidor y presenta los resultados.
- ldapadd - Añade el contenido de un fichero LDIF a la base de datos.
- ldapdelete - Borra la entrada indicada del servidor LDAP.
- ldapmodify - Pues eso, modifica, según un fichero LDIF, registros de la base de datos.

18.1. Diseño de Árboles de Directorios

El diseño de los árboles es crítico para conseguir bases de datos de fácil administración.

Conviene tener en cuenta las ventajas de las base de datos de LDAP, principalmente:

- Suponen una forma sencilla de facilitar el acceso masivo a la información centralizada, evitando los problemas de integridad que suponen las bases de datos.
- Pueden tener atributos múltiples sin que eso suponga una condición previa de diseño ni un consumo adicional de espacio de almacenamiento. Esta característica potencia enormemente las opciones de búsqueda.

Por ejemplo, el la ficha de una persona se pueden incluir varios registros de nombre. Podríamos rellenar alguno con el valor de Francisco otro con Paco y otro con Curro, según como se conozca a la persona en la organización. Con LDAP, búsquedas con cualquiera de estos nombres ofrecerían resultados.

18.1.1. Atributos

A la hora de definir nuevos registros (que deberán quedar recogidos en un esquema que se incluirá en el slapd.conf del servidor), se deben definir los siguientes valores:

- Nombre del atributo. Deberá buscarse un nombre que tenga ni pueda tener conflictos con los oficiales. Para ello, puede ser conveniente precederlo del nombre o dominio de la organización. Por ejemplo, cajabadajoz.com-Oficina.
- Descripción - Pues la descripción de lo que significa el atributo.
- Identificador - Un identificador numérico único del atributo. Para conseguir números de atributos de forma gratuita se puede acudir a www.iana.org.
- Sintaxis - Que puede ser uno de los siguientes valores:

  - dn - Nombre distinguido.
  - cis - Cadena de texto no sensible al tipo (mayúsculas/minúsculas).
  - ces - Cadena de texto sensible al tipo (mayúsculas/minúsculas).
  - int - Entero.
  - tel - Número de teléfono. Para las búsquedas se ignorarán los caracteres que no sean dígitos.
  - bin - Contenido binario.
  - personalizada - Sintaxis definida por el usuario.
- Unicidad - Indica si el atributo debe ser único.

18.1.2. Objetos de Clase

Cada registro contenido en una base de datos LDAP se corresponde con uno o varios objetos de clase (objectClass). Esto significa que debe seguir las reglas sintácticas para ellos. Estas reglas indican qué atributos puede tener el registro, de qué tipo son y cuales son obligatorios.

18.1.3. Criterios de Diseño

A la hora de diseñar un árbol de directorios, conviene tener en cuenta el criterio de invariabilidad. Es decir, nos aterra tener que modificar la estructura del árbol.

Por este motivo, son preferibles los árboles anchos y poco profundos, que en su estructura no sigan referencias arbitrarias propensas al cambio (como el organigrama de la organización). Suele ser más recomendable, en lugar de crear una Unidad Organizacional (ou) para cada departamento de la organización, crear una por cada tipo de entidad que vayamos a gestionar (oficinas, personas, equipos, etc.).

18.2. Identificación Centralizada con LDAP

Para conseguir que una máquina GNU/Linux identifique a sus usuarios a través de un servidor LDAP, es necesario instalar los siguientes paquetes: libnss_ldap y libpam_ldap.

libpam_ldap hará que el proceso de conexión (login) valide el usuario y la contraseña con el servidor LDAP que le indiquemos. Pero no basta con eso.

Muchas aplicaciones buscan información del usuario, como su UID para diversos motivos. Por ejemplo, el comando ls muestra el nombre del usuario propietario de un fichero y del grupo. Esta información también debe proporcionarla el servidor LDAP. Para ello, el paquete libnss_ldap permitirá que el sistema NSS (Name Service Switch) interactúe con el servidor LDAP.

Para que este sistema pueda acceder al servidor LDAP, hay que modificar el fichero de configuración /etc/nsswitch.conf del siguiente modo:

```
passwd: files ldap
shadow: files ldap
group:  files ldap
```

Esto indica al sistema que busque las contraseñas primero en el fichero local (/etc/passwd) y después en el servidor LDAP. Y lo mismo para la información de shadow y grupos.

18.3. Bibliografía

- En www.openldap.org podemos encontrar toda l información y el software necesario de este sistema.
- Guía del Administrador de OpenLDAP
- Naturalmente, las Páginas de Manual de cada uno de los comandos y los ficheros de configuración.

Anexo I - Protocolo de Red TCP/IP

El Modelo OSI

A principios de los años 80, la Organización Internacional para la Estandarización (ISO) consideró que merecía la pena definir un modelo que estableciera un estándar para la construcción de protocolos de redes telemáticas. Hasta la fecha, cada fabricante de ordenadores había desarrollado su propio protocolo de comunicación. Y estos resultaban siempre propietarios y de difícil interconectividad.

Fruto de esta iniciativa surgió el modelo OSI (Open Systems Interconnectivity) que establecía un modelo de siete niveles que todo protocolo de red debería cumplir (para ser estándar).

Ningún fabricante ha cumplido jamás este modelo, sobre todo por su amplia extensión. Pero sí que se utiliza de referencia para la catalogación de protocolos. De esta forma, los protocolos existentes se clasifican ahora según el nivel OSI al que corresponde su funcionalidad.

Para poder tener una visión completa de la funcionalidad que aporta cualquier protocolo de red, echaremos un vistazo rápido a la "Torre OSI".

Básicamente, el modelo OSI divide la funcionalidad de la comunicación entre ordenadores, en siete niveles.

Nivel de Aplicación

Es el nivel superior, y estable la comunicación extremo a extremo entre las aplicaciones finales. Por ejemplo, en un chat IRC, la aplicación IRC de un cliente (Gaim) habla "extremo a extremo" con la aplicación IRC del otro cliente (Netmeeting).

La funcionalidad de este nivel depende en gran medida de qué aplicaciones finales se trate (navegador web - servidor web, cliente de correo - buzón, videoconferencia - videoconferencia, etc.). Por eso el modelo OSI no define específicamente la funcionalidad de este nivel.

Nivel de Presentación

Es el nivel responsable de la adaptación de formatos entre los niveles de aplicación de los dos extremos. Es decir, su trabajo consiste en acordar los formatos de intercambio (codificación UTF-8 o ISO-8859-1, a 16 o 32 bits, etc.). También se encargar de la compresión y descompresión de datos así como de la encriptación y desencriptación.

Nivel de Sesión

Es el nivel responsable de establecer y mantener la información de sesión ente emisor y receptor. Por ejemplo, mantener el nombre de usuario y su contraseña, el directorio de trabajo, identificador de base de datos, etc.

Nivel de Transporte

Es el nivel responsable de la comunicación extremo a extremo. Se encarga de que los datagramas que gestionará el nivel de red (y que puede llegar desordenados) se pasen íntegros, completos y ordenados al nivel de sesión. Para ello, realiza funciones de solicitud de reenvíos, comprobación de integridad, control de flujo, etc.

Nivel de Red

Es el primer nivel que ya no se encarga de la comunicación extremo a extremo, sino de la comunicación entre extremos de cada segmento de red (punto a punto). Se encarga de funciones tales como la búsqueda de caminos (routing), tarificación de tráficos (accounting), etc.

Nivel de Enlace de Datos

Su responsabilidad es la transmisión de datos punto a punto sin errores. Para ello, divide el datagrama proveniente del nivel de red en tramas, controlando la retransmisión de las mismas en caso de error, el control de flujo, etc.

Nivel Físico

Es el último nivel. Su responsabilidad es la transmisión de bits punto a punto, determinando para ello parámetros físicos de red, como voltaje correspondiente al 1, microsegundos de duración, número de pines de los conectores.

El Protocolo TCP/IP

El protocolo TCP/IP se ha impuesto como un estándar de facto en las redes telemáticas. Es el protocolo nativo de Internet y hoy se aplica en un gran número de redes de área local y de todo tipo. Básicamente, es la opción por defecto.

Como todos los protocolos, y especialmente como todos los anteriores a la definición del modelo OSI, no sigue claramente el estándar, pero básicamente se compone de dos protocolos: IP, que se corresponde con un protocolo de nivel de red, y TCP que se corresponden con un protocolo de nivel de transporte.

Ambos protocolos (TCP e IP) fueron diseñados por DoD (Departamento de Defensa de Estados Unidos) para su uso en la red ARPANET (embrión de la que es hoy Internet).

Internet Protocol (IP)

Como ya hemos dicho, se trata de un protocolo de nivel de red. Es decir, su misión es garantizar la comunicación punto a punto. Fundamentalmente, tiene que resolver la determinación del camino que deben seguir los paquetes (ya que IP no realiza tareas de contabilidad (accounting)).

Características

IP es un protocolo orientado a datagrama. Esto quiere decir que la información que recibe del nivel de transporte (TPC) y que se

denomina paquete es descompuesta en datagramas de un tamaño máximo de 64Kb. Cada datagrama es enviado por la red hacia su destino, pero no todos los datagramas siguen el mismo camino. Ni siquiera los que integran un mismo mensaje. Esto hace que los datagramas puedan llegar desordenados al destino e incluso que algunos no lleguen. Será responsabilidad del protocolo TCP reordenarlos y garantizar que el mensaje está completo.

Un datagrama IP se compone de una cabecera y un texto (datos). La cabecera tiene los siguientes campos:

1. Versión: Indica la versión del protocolo. De esta forma, un nodo puede atender datagramas de versiones distintas.
2. IHL: Longitud total de la cabecera. Como las cabeceras de los datagramas tienen longitud variable, en algún sitio hay que indicarle al nodo donde terminan y empiezan los datos.
3. Tipo de Servicio: Indica el tipo de servicio que debe darse al datagrama, pudiendo optar entre velocidad o fiabilidad.
4. Longitud Total: Indica la longitud total del datagrama, cabecera más datos.
5. Identificación: Indica a qué datagrama pertenece el fragmento, en el caso que haya sido dividido.
6. DF: Se trata de una bandera que indica si el datagrama No Debe ser Fragmentado (Do not Fragment).
7. MF: Se trata de una bandera que indica si el datagrama tiene más fragmentos (More Fragments).
8. Offset del Fragmento: Si el datagrama está fragmentado, indica el número de orden del fragmento.
9. Tiempo de Vida: Indica el tiempo en segundos, hasta un máximo de 255 segundos, que puede vivir el paquete en la red. Se trata de un contador que se va descontando.
10. Protocolo: Indica el protocolo de nivel de transporte que se hará cargo del datagrama cuando esté completo.
11. Checksum: Suma de control utilizada para verificar la integridad (ausencia de errores) del datagrama.
12. Dirección de Origen: Dirección IP del origen del datagrama.
13. Dirección de Destino: Dirección IP del destino del datagrama.
14. Opciones: Un campo de ampliación que se utiliza para seguridad, enrutamiento, control de errores, depuración, sellos temporales, ampliaciones...

Direcciones IP

Cada uno de los extremos de una comunicación IP se identifica mediante una dirección, llamada dirección IP (¡qué original!).

Las direcciones IP están formadas por cuatro dígitos hexadecimales de 8 bits (IPv4). Es decir, que varían entre 0.0.0.0 y 255.255.255.255. Cada dirección IP tiene dos partes, la dirección de red a la que pertenece la máquina en cuestión, y la dirección de la máquina en la red. Pero la parte de la dirección IP que corresponde a la dirección de red y la que corresponde a la máquina varía según el tipo de red.

Hay tres tipos de redes:

- Tipo A: Tienen más de 65.025 máquinas en la red y menos de 16.581.375 (si la red tiene un número mayor de máquinas, debe utilizar otra versión de IP, IPv6). En las redes tipo A, la dirección de red es el primer número de la dirección IP, y los tres siguientes constituyen la dirección de máquina.
- Tipo B: Tienen más de 255 y menos de 65.025 máquinas en la red. En ellas, los dos primeros números son la dirección de red, y los dos últimos la de máquina.
- Tipo C: Tienen menos de 255 máquinas, y en ellas, la dirección de red está formada por los tres primeros números y la de máquina por el último.

Por ejemplo, la dirección IP 163.23.109.68 se corresponde con:

- En una red tipo A: La dirección de red es 163.0.0.0 (se completa con ceros), y la de máquina 23.109.68.
- En una red tipo B: La dirección de red es 163.23.0.0 y la de máquina 109.68.
- En una red tipo C: La dirección de red es 163.23.109.0, y la de máquina 68.

Y ... ¿cómo podemos saber el tipo de red?. Pues por otro parámetro que se denomina máscara de subred. La máscara de subred permite identificar qué direcciones IP pertenecen a la misma red y cuales son externas.

Una máscara de subred, para una red tipo C, podría ser 255.255.255.0. Basta con realizar un AND entre la máscara de subred y la dirección IP de un destino para saber si éste pertenece a la misma red. Por ejemplo: 163.23.109.68 AND 255.255.255.0 = 163.23.109.0 que coincide con la dirección de red de la máquina.

Se pueden buscar máscaras de red más complicadas para crear subredes lógicas.

Enrutamiento

La principal función del protocolo IP de un nodo que recibe un datagrama es determinar por donde debe enviarlo para que llegue a su destino. Y esta decisión debe ser tomada en función de la dirección IP. A este problema se le denomina enrutamiento.

Ya hemos visto como una máquina puede saber si una IP de destino está o no en su subred. Si está, el enrutamiento es fácil, pongo el datagrama en el cable de mi red y ya se encargará el destinatario (que estará conectado a mi red) de cogerlo. Pero... ¿y si no está?

Supongamos que utilizamos una máquina para interconectar varias redes. Naturalmente a esa máquina le instalamos varios interfaces de red (tarjetas) uno para cada red. El problema del protocolo IP de esa máquina es como enrutar cada uno de los datagramas que recibe (o genera) para que lleguen a su destino.

La forma de hacerlo está muy condicionada por el origen del protocolo, en el que prevalecía la seguridad (frente a un ataque nuclear) que la rapidez. Por ello, el enrutamiento se realiza en base a tablas de routing en las que cada nodo registra las reglas de enrutamiento. Cada regla permite decidir porqué interfaz se enrutará cada datagrama en función de su destino. Los nodos de enrutamiento, recuerdan los datagramas que procesan y por qué interfaz se enrutó. Al llegar un datagrama, el nodo verifica si existe alguna regla para su enrutamiento, y de no existir, utiliza un enrutamiento por defecto.

IPv6 vs IPv4

Cuatro cifras de 8 bits dan para muchas direcciones IP. Sin embargo, la penosa gestión que se ha hecho de su asignación ha llevado a que algunas universidades americanas tengán reservadas más direcciones IP globales que algunos paises asiáticos.

Además, algunas direcciones están reservadas para usos especiales (redes privadas, broadcasting...). En consecuencia, la versión 4 del protocolo IP (IPv4) se ha quedado corta. Ya ha aparecido para reemplazarla (de forma progresiva) la versión de 6 cifras IPv6.

Pero además de disponer de más direcciones IP, IPv6 tiene otras ventajas:

- Autoconfiguración: Con IPv4, cada ordenador que se conecta a la red necesita ser configurado con una dirección IP, una máscara de subred, un gateway... Aunque existen parches para realizar esta configuración de forma automática (DHCP, BOOTP...), no son intrínsecos al protocolo y, por lo tanto, no son eficientes. En IPv6, la autoconfiguración es obligatoria para todas las redes, facilitando el mantenimiento de grandes redes de ordenadores.
- Seguridad: En IPv6 es obligatorio que las comunicaciones viajen encriptadas, garantizando la integridad y autenticidad.
- Multicast: Con IPv4, si un emisor quiere mandar un datagrama a varios receptores, el datagrama debe ser enviado múltiples veces. Si esta comunicación es, por ejemplo, un vídeo, el consumo de tráfico de red es muy alto. Y todo, para enviar varias copias de un mismo mensaje. Con IPv6 el método de multicast permite enviar una sola copia, siendo la propia red la que se encarga de realizar las copias cuando sea necesario.

- Movilidad: IPv6 permite que nos conectemos a distintos nodos de la red manteniendo nuestra configuración. De esta forma, la informática puede ser mucho más móvil.
- Velocidad: Las cabeceras de los datagramas en IPv4 tienen un tamaño variable. Al no estar alineados ni a 16 ni a 32 bits, los procesadores de los routers deben hacer varias operaciones de lectura para su proceso. En IPv6, el tamaño de las cabeceras se alinea a 32 bits, permitiendo a los procesadores de enrutamiento un trabajo más eficiente, y aumentando así la velocidad de la red.

IPv6 es ya una realidad. No solo la utilizan varios proveedores de acceso asiáticos para resolver su problema, sino que está soportada por el núcleo de Linux.

Transmission Control Protocol (TCP)

Al igual que IP, el protocolo TCP fue creado por el DoD para su red ARPANET. Se trata, básicamente, de un protocolo de nivel de transporte.

Características

TCP está orientado a conexión extremo a extremo, lo que significa que la comunicación entre ambos niveles del protocolo se realiza directamente entre el emisor y el receptor. Siendo necesarias determinadas operaciones para el establecimiento y el cierre. La unidad de trabajo es el paquete que posteriormente se dividirá en uno o varios datagramas en el nivel de red. El tamaño máximo de paquete para TCP/IP es de 64Kb.

Un paquete TCP se compone de una cabecera y unos datos. La cabecera tiene los siguientes campos:

- Puerto de Origen: Identifica el origen del paquete.
- Puerto de Destino: Identifica el destino del paquete.
- Número de Secuencia: Indica el número de secuencia del paquete para poder ordenarlos.
- Piggyback: Permite reconocer la recepción correcta en un paquete de respuesta.
- Longitud de Cabecera: Indica la longitud de la cabecera del paquete.
- Flag URG: Indica si el paquete continen datos urgentes.
- Flag ACK: Indica si se utiliza el campo de Piggybacking.
- Flag EOM: Señala el final del mensaje.
- Flag RST: Resetea la conexión.
- Flag SYN: Indica el inicio de una conexión.

- Flag FIN: Concluye la conexión.
- Window: Permite el control de flujo indicando el estado de las ventanas.
- Checksum: Suma de control de integridad.
- Puntero de Urgente: Indica en qué parte del paquete están los datos urgentes.
- Opciones: Campo utilizado para comunicar tamaños de buffers y otras lindezas durante la conexión.

Servicios

El protocolo TCP ofrece a su nivel de superior un conjunto de servicios. Entre ellos están el establecimiento de conexión, el envío de datos, etc.

Estos servicios de TCP se utilizan mediante sockets. Hay dos tipos de sockets: los de servidor permiten recibir comunicaciones por un determinado puerto de una máquina, y los de cliente, que solicitan comunicaciones a otras máquinas en determinados puertos.

Conceptos de Interconexión de Redes

El protocolo TCP es, probablemente, el más extendido. Por este motivo, se ha aplicado a muy diversas situaciones para resolver problemas muy diferentes. Esto ha generado un lenguaje propio, con nombres comunes que se utilizan tanto para describir funciones como para identificar las máquinas que las realizan. A continuación se describen algunos de estos conceptos:

- Bridge: Consiste en interconectar dos redes, mediante un ordenador con dos tarjetas, de modo que se comporten como una única red de mayor extensión. Permite unir redes con tecnologías distintas.
- IPMasquerade: Permite que varios equipos conectados en una misma red utilicen una misma dirección IP para salir a Internet. La máquina que hace el servicio se encarga de redirigir los paquetes a su verdadero destinatario.
- IPAccounting: Son servicios para el control de comunicación. Permiten conocer el tráfico y tarificarlo.
- IPAliasing: Mediante este servicio, un mismo interfaz puede responder a varias direcciones IP. Se utiliza, por ejemplo, en ISPs que albergan varias webs.
- Traffic Shapping: Permite controlar el tráfico de un canal, estableciendo máximos por usuarios y/o aplicaciones. Se utiliza en ISPs que establecen cuotas de tráfico a sus usuarios.

- Firewall: Consiste en un filtrado de paquetes en base a sus direcciones IP y puertos de destino. Los firewalls se utilizan para mejorar la seguridad de una red, impidiendo que se accedan a puertos y máquinas sensibles.
- Port Forwarding: Este servicio permite redirigir los paquetes que una máquina recibe en un puerto hasta otra, ocultando así la verdadera dirección del destinatario.
- Load Balancing: Este servicio permite distribuir la carga entre varios servidores.
- EQL: Consiste en una equalización de tal forma que si una máquina dispone de varias líneas de comunicación lenta (por ejemplo, RTB) se comportan como una única línea de capacidad la suma de todas.
- Proxy: El servicio de proxy permite que una máquina actúe en nombre de otras frente a terceros, impidiendo así que se conozca la dirección original.
- Dial on Demand: Consiste en un servicio que establece conexiones según sean demandadas por los usuarios.
- Tunnelling: Permite encapsular otros protocolos (IPX, Netbeui...) en paquetes TCP.

Demonios y Servicios TCP/IP en Linux

Las máquinas Linux son máquinas intrínsecamente TCP/IP. Incluso aunque no estén conectadas a una red, el protocolo TCP/IP está funcionando en ellas mediante un interfaz de loopback.

Internamente, las comunicaciones TCP que recibe el ordenador tienen dos formas de ser atendidas:

- Stand Alone: Mediante este sistema, cada puerto de comunicación tiene asignado un programa de atención que se encarga de la comunicación en ese puerto. El problema de este sistema es que cada programa de atención debe gestionar la comunicación y el protocolo TCP/IP. Por este motivo, la configuración más habitual es la de superdemonio.
- Superdemonio: Esta es la configuración más habitual. En ella, un superdemonio llamado

```
inetd
```

recibe las comunicaciones en todos los puertos y las transmite a los programas de atención. De esta forma, cada programa de atención se especializa en su función (servicio de páginas web, direccionamiento de correo, comunicación por telnet...) dejando que inetd se encarge de las comunicaciones.

Bibliografía y Referencias

- En athos.rutgers.edu/runet/tcp-intro.doc puede obtenerse un documento de introducción a TCP/IP.
- También los siguientes HOWTO ofrecen mucha información sobre la estructura y la configuración de la red en sistemas Linux: Net-HOWTO y Networking-Overview-HOWTO.