Administración de sistemas GNU/Linux

Book · September 2010

CITATIONS
0

READS
1,580

2 authors:

Josep Jorba Esteve
Universitat Oberta de Catalunya
68 PUBLICATIONS **304** CITATIONS
SEE PROFILE

Remo Suppi
Autonomous University of Barcelona
85 PUBLICATIONS **217** CITATIONS
SEE PROFILE

Administración de sistemas GNU/Linux

Josep Jorba Esteve
Remo Suppi Boldrito

PID_00157330

Material docente de la UOC

Universitat Oberta de Catalunya
www.uoc.edu

Josep Jorba Esteve

Ingeniero Superior en Informática. Doctor ingeniero en Informática por la UAB. Profesor de los Estudios de Informática, Multimedia y Telecomunicaciones de la UOC, Barcelona.

Remo Suppi Boldrito

Ingeniero en Telecomunicaciones. Doctor en Informática por la UAB. Profesor del Departamento de Arquitectura de Computadores y Sistemas Operativos en la Universidad Autónoma de Barcelona.

Primera edición: septiembre 2010

Av. Tibidabo, 39-43, 08035 Barcelona
Diseño: Manel Andreu
Realización editorial: Eureca Media, SL
ISBN: 978-84-693-6360-7
Depósito legal: B-33.177-2010

Los autores agradecen a la Fundación para la Universitat Oberta de Catalunya la financiación de la primera edición de esta obra, enmarcada en el máster internacional de Software libre ofrecido por la citada institución.

Introducción

Los sistemas GNU/Linux han alcanzado un grado de madurez significativa, que los hacen perfectamente válidos para su integración en cualquier ambiente de trabajo, ya sea desde el escritorio del PC personal, hasta el servidor de una gran organización.

El objetivo principal que nos proponemos en este curso es nuestra introducción al mundo de la administración de los sistemas GNU/Linux.

Aprenderemos cómo proporcionar desde GNU/Linux los servicios necesarios a diferentes ambientes de usuarios y máquinas. El campo de estudio de la administración de sistemas es de dimensiones muy amplias y variado en conocimientos necesarios. Existen muchas tareas diferentes a realizar, amplios problemas por tratar, y debemos disponer de grandes conocimientos de hardware y software, asimismo, tampoco estaría de más un poco de psicología para tratar con los usuarios finales de los sistemas.

El curso no pretende abordar una distribución GNU/Linux particular, pero se han escogido un par de ellas significativas, por su amplio uso, para tratar los ejemplos de tareas: Debian y Fedora.

Respecto al campo de la administración, ésta se intentará gestionar desde el nivel más bajo posible, normalmente desde la línea de comandos y los ficheros de configuración. Se comentarán, en su caso, herramientas de más alto nivel, pero hay que tener cuidado con estas últimas, ya que suelen ser fuertemente dependientes de la distribución utilizada e incluso de la versión de ésta; además, estas herramientas suelen variar funcionalidades entre versiones. La administración de bajo nivel suele ser mucho más dura, pero conocemos con qué estamos operando y dónde podemos obtener los resultados, además de que nos aporta muchos más conocimientos extra, y control preciso sobre las diferentes tecnologías utilizadas.

Las distribuciones escogidas han sido las ultimas disponibles de: Debian (o compatibles como las variantes de Ubuntu), y Fedora (o compatibles como diferentes versiones comerciales de Red Hat, o de comunidad como CentOS), siendo estas (Debian y Fedora) las más utilizadas en el momento de confeccionar estos materiales. La distribución Debian es un paradigma dentro del movimiento Open Source, por no pertenecer a ninguna empresa y estar confeccionada básicamente por las aportaciones de los voluntarios distribuidos por todo el mundo. Debian, además, integra casi exclusivamente software libre

(pueden añadirse otros aparte). Además, gran número de distribuciones con éxito importante (como las variantes de Ubuntu, que de hecho han superado a Debian como uso en escritorio) tienen a Debian como distribución base.

Nota

Principales distribuciones y empresas comentadas en el texto:

http://www.debian.org/

http://www.ubuntu.com/

http://www.canonical.com/

http://fedoraproject.org/

http://www.redhat.com/

http://www.centos.org/

http://www.novell.com/linux/

http://www.opensuse.org/

Red Hat, por otra parte, es una de las empresas más solventes en el panorama comercial, y por eso sea quizás la que otorgue más soporte a nivel empresarial (mediante servicios de pago), mediante sus distribuciones comerciales de Red Hat Enterprise Linux (tambien existen variantes libres alternativas como CentOS). Por otra parte, su entrada como patrocinador ha permitido ampliar los resultados del proyecto Fedora (y la consecuente distribución GNU/Linux), como conjunto de test para sus distribuciones posteriores, y un desarrollo de una amplia comunidad que ha crecido a su alrededor. Este caso forma una interesante experiencia combinada de comunidad y empresa en el desarrollo de la distribución (como después se ha visto con la ampliación a otros casos como Novell, con su versión comercial SUSE Linux y su proyecto de comunidad OpenSUSE). Por el contrario, en Debian el soporte depende de los voluntarios y del conocimiento compartido de los usuarios, centrándose en su comunidad, aunque en los últimos tiempos también ha recibido el soporte (no sin ciertos problemas en el proceso) de Canonical, desarrollador de Ubuntu.

Siendo la administración de sistemas un campo tan amplio, este manual sólo pretende introducirnos en este apasionante (y cómo no, también a veces frustrante) mundo. Veremos algunas de las tareas típicas, y cómo tratar las problemáticas que nos aparecen. Pero la administración es un campo que se aprende día a día, con el trabajo diario. Y desde aquí advertimos que este manual es un trabajo abierto, que con sus aciertos y los más que probables errores se puede ver complementado con los comentarios de sus (sufridores) usuarios. De modo que son bienvenidos, por parte de los autores, cualquier tipo de comentarios y sugerencias de mejora de los presentes materiales.

Comentamos, por último, que el contenido refleja el estado de las distribuciones y de las herramientas de administración en el momento de su confección, o de las correspondientes revisiones o reediciones.

Contenidos

6. Usuarios y grupos
7. Servidores de impresión
8. Discos y gestión *filesystems*
9. Software: actualización
10. Trabajos no interactivos
11. Taller: prácticas combinadas de los apartados

Módulo didáctico 6

Administración de red

Remo Suppi Boldrito

1. Introducción a TCP/IP (TCP/IP suite)
2. Conceptos en TCP/IP
3. ¿Cómo se asigna una dirección Internet?
4. ¿Cómo se debe configurar la red?
5. Configuración del DHCP
6. IP Aliasing
7. IP Masquerade
8. NAT con el kernel 2.2 o superiores
9. ¿Cómo configurar una conexión DialUP y PPP?
10. Configuración de la red mediante hotplug
11. *Virtual private network* (VPN)
12. Configuraciones avanzadas y herramientas

GNU Free Documentation License

Version 1.3, 3 November 2008

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible.

You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

- D. Preserve all the copyright notices of the Document.

- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

- H. Include an unaltered copy of this License.

- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English

version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the

Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this
document under the terms of the GNU Free Documentation Li-
cense, Version 1.3 or any later version published by the Free
Software Foundation; with no Invariant Sections, no Front-
Cover Texts, and no Back-Cover Texts.
```

A copy of the license is included in the section entitled "GNUFree Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with
theFront-Cover Texts being LIST, and with the Back-Cover Texts
being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Bibliografía

[Bai03] Bailey, E. C. (2003). *RedHat Maximum RPM*. http://www.redhat.com/docs/books/max-rpm/index.html

Ofrece una amplia visión de los sistemas de paquetes de software de las distribuciones Debian y Fedora/Red Hat.

[Ban] Banerjee, T. "Linux Installation Strategies HOWTO". *The Linux Documentation Project.*

The Dot Gnu Project. http://www.gnu.org/software/dotgnu/

[Bar] Barrapunto. *barrapunto site*. Noticias Open Source. http://barrapunto.com

[Bas] Mike, G. "BASH Programming - Introduction HOWTO". *The Linux Documentation Project.*

Ofrece una amplia introducción (y conceptos avanzados) a la programación de *shell scripts* en *bash*, así como numerosos ejemplos.

Barnett, B. "AWK". © General Electric Company. http://www.grymoire.com/Unix/Awk.html

"Big Admin: Script Examples". http://www.sun.com/bigadmin/scripts/filter.jsp

[Bro01] Scott Bronson (2001). "VPN PPP-SSH". *The Linux Documentation Project,*.

[Bul] Bulma. "Bulma Linux User Group". Documentación general y comunidades de usuarios. http://bulmalug.net

[Cen] The Comunity ENTerprise Operatyng System. http://www.centos.org

[Cis00] Cisco (2000). "TCP/IP White Paper". http://www.cisco.com

[Com01] Douglas Comer (2001). *TCP/IP Principios básicos, protocolos y arquitectura*. Prentice Hall.

[Coo] Cooper, M. (2006). "Advanced bash Scripting Guide". *The Linux Documentation Project* (guías). http://tldp.org/LDP/abs/html/index.html

Ofrece una amplia introducción (y conceptos avanzados) a la programación de *shell scripts* en *bash*, así como numerosos ejemplos.

[Debb] Comunidad Debian. "Distribución Debian". http://www.debian.org

[Deb] Debian. "Sitio Seguridad de Debian". http://www.debian.org/security/

Ofrece una amplia visión de los sistemas de paquetes de software de las distribuciones Debian y Fedora/Red Hat.

[Debba] Debian. "Software Libre vs Software Abierto". http://www.debian.org/intro/free.es.html

[Dis] Distrowatch. "Distribuciones Linux disponibles". Seguimiento de las distribuciones GNU/Linux y novedades de los paquetes software. Y enlaces a los sitios de descarga de las imágenes ISO de los CD/DVD de las distribuciones GNU/Linux. http://www.distrowatch.com

[Dra99] Joshua Drake. "Linux Networking". *The Linux Documentation Project*, 1999.

[Fed] The Fedora Project. http://fedoraproject.org

[FHS] *FHS Standard* (2003). http://www.pathname.com/fhs

[Fre] Freshmeat. "Freshmeat site". Listado de proyectos Open Source. http://freshmeat.org

[Fri02] Frisch, A. (2002). *Essential System Administration*. O'Reilly.

Administración de GNU/Linux y UNIX. Comenta de forma amplia aspectos de administración local y gestión de sistemas de impresión.

[FSF] FSF. "Free Software Foundation y Proyecto GNU". http://www.gnu.org

[Gar98] Bdale Garbee (1998). *TCP/IP Tutorial*. N3EUA Inc.

Garrels, Machtelt. "Bash Guide for Beginners". http://tldp.org/LDP/Bash-Beginners-Guide/html/index.html

"Man bash". http://www.gnu.org/software/bash/manual/bashref.html

[Gnu] Gnupg.org. *GnuPG Web Site*.http://www.gnupg.org/

[Gon] Gonzato, G. "From DOS/Windows to Linux HOWTO". *The Linux Documentation Project*.

[Gt] Taylor, G.; Allaert, D. "The Linux Printing HOWTO". *The Linux Documentation Project*.

Ofrece información actualizada de los sistemas de impresión y su configuración, así como detalles de algunas impresoras. Para detalles concretos de modelos de impresora y controladores, podéis dirigiros a: http://www.linuxprinting.org/

[Hin00] Hinner, M. "Filesystems HOWTO". *The Linux Documentation Project*.

Información sobre los diferentes sistemas de ficheros disponibles y los esquemas de creación de particiones para la instalación del sistema.

[His] HispaLinux. "Comunidad Hispana de Linux". Documentación general y comunidades de usuarios. http://www.hispalinux.es

[IET] IETF. "Repositorio de Request For Comment desarrollado por Internet Engineering Task Force (IETF) en el Network Information Center (NIC)".

[Joh08] Johnson, Michael K. (1998). "Linux Information Sheet". *The Linux Documentation Project*.

[KD00] Olaf Kirch; Terry Dawson. *Linux Network Administrator's Guide*.

[Knp] Distribución Knoppix. http://knoppix.org

[Koe] Koehntopp, K. "Linux Partition HOWTO". *The Linux Documentation Project.*

Información sobre los diferentes sistemas de ficheros disponibles y los esquemas de creación de particiones para la instalación del sistema.

[Law07] David Lawyer (2007). "Linux Módem". *The Linux Documentation Project*.

[Lev] Levenez, Eric. "UNIX History". http://www.levenez.com/unix

Linux Journal. *Linux Journal [Revista Linux]*. Revista GNU/Linux. http://www.linuxjournal.com

Linux Magazine. *Linux Magazine*. Revista GNU/Linux. http://www.linux-mag.com/

[lin03b] *FHS Standard* (2003). http://www.pathname.com/fhs

[Linc] *Linux Standards Base project*. http://www.linux-foundation.org/en/LSB

[Log] LogCheck. http://logcheck.org/

[LPD] LPD. *The Linux Documentation Project*. http://www.tldp.org

Proporciona los Howto de los diferentes aspectos de un sistema GNU/Linux y un conjunto de manuales más elaborados.

[Mal96] Fred Mallett (1996). *TCP/IP Tutorial*. FAME Computer Education.

Mike G. "Programación en BASH - COMO de introducción". http://es.tldp.org/COMO-INSFLUG/COMOs/Bash-Prog-Intro-COMO/

[Mon] Monit. http://www.tildeslash.com/monit/

[Mor 03] Morill, D. (2003). *Configuración de sistemas Linux*. Anaya Multimedia.

Buena referencia de configuración de sistemas Linux, con algunos casos de estudio en diferentes entornos; comenta diferentes distribuciones Debian y Red Hat.

[Mou01] Gerhard Mourani (2001). *Securing and Optimizing Linux: The Ultimate Solution*. Open Network Architecture, Inc.

[Mor03] Morill, D. (2003). *Configuración de sistemas Linux*. Anaya Multimedia.

[Nem06] Nemeth, E.; Snyder, G.; Hein, T. R. (2006). "Linux Administration Handbook" (2.ª ed.). Prentice Hall.

Trata de forma amplia la mayoría de aspectos de administración y es una buena guía genérica para cualquier distribución.

[New] Newsforge. "Newsforge site". Noticias Open Source. http://newsforge.org

[Nt3] NTFS-3g Project: NTFS-3G Read/Write Driver. http://www.ntfs-3g.org/

O'Reilly Associates (2000). Y como *e-book* (free) en Free Software Foundation, Inc. http://www.tldp.org/guides.html

[OSDb] OSDN. "Open Source Development Network". Comunidad de varios sitios web, noticias, desarrollos, proyectos, etc. http://osdn.com

[OSIa] OSI. "Listado de licencias Open Source". http://www.opensource.org/licenses/index.html

[OSIb] OSI (2003). "Open Source Definition". http://www.opensource.org/docs/definition.php

[OSIc] OSI (2003). "Open Source Iniciative". http://www.opensource.org

[PPP00] Linux PPP (2000). "Corwin Williams, Joshua Drake and Robert Hart". *The Linux Documentation Project*.

[Pri] Pritchard, S. "Linux Hardware HOWTO". *The Linux Documentation Project*.

[PS02] Enríquez, R.; Sierra, P. (2002). *Open Source*. Anaya Multimedia.

[Qui01] Quigley, E. (2001). *Linux shells by Example*. Prentice Hall.

Comenta los diferentes *shells* de programación en GNU/Linux, así como sus semejanzas y diferencias.

[Ran05] David Ranch. (2005). "Linux IP Masquerade" y John Tapsell. Masquerading Made Simple . *The Linux Documentation Project*.

[Ray98] Raymond, Eric (1998). *La catedral y el bazar*. http://es.tldp.org/Otros/catedral-bazar/cathedral-es-paper-00.html

[Ray02a] Raymond, Eric (2002). "UNIX and Internet Fundamentals". *The Linux Documentation Project*.

[Rayb] Raymond, E. S. "The Linux Installation HOWTO". *The Linux Documentation Project*.

Red Hat Inc. "Distribución Red Hat". http://www.redhat.com

[Rid00] Daniel López Ridruejo (2000). "The Linux Networking Overview". *The Linux Documentation Project*.

Robbins, Arnold. "UNIX in a Nutshell" (3.ª ed.). http://oreilly.com/catalog/unixnut3/chapter/ch11.html

"The GNU awk programming language". http://tldp.org/LDP/Bash-Beginners-Guide/html/chap_06.html

[Sal94] Salus, Peter H. (1994, noviembre). "25 aniversario de UNIX". *Byte España* (núm. 1).

Scientific Linux. http://www.scientificlinux.org

[Sam] Samba Project. http://samba.org

[Sama] Samba HOWTO and Reference Guide (Chapter Domain Control). http://samba.org/samba/docs/man/Samba-HOWTO-Collection/samba-pdc.html

[Samb] Samba Guide (Chapter Adding Domain member Servers and Clients). http://samba.org/samba/docs/man/Samba-Guide/unixclients.html

[Sec00] Andrés Seco (2000). "Diald". *The Linux Documentation Project*.

[Skob] Skoric, M. "Linux+WindowsNT mini-HOWTO". *The Linux Documentation Project*.

[Sla] Slashdot. "Slashdot site". Sitio de noticias comunidad Open Source y generales informática e Internet. http://slashdot.org

[SM02] Schwartz, M. y otros (2002). *Multitool Linux - Practical Uses for Open Source Software*. Addison Wesley.

[Smb] Entrada "Server Message Block" en la Wikipedia. http://en.wikipedia.org/wiki/Server_Message_Block

[Smi02] Smith, R. (2002). *Advanced Linux Networking*. Addison Wesley.

Administración de GNU/Linux y UNIX. Comenta de forma amplia aspectos de administración local y gestión de sistemas de impresión.

[Sob10] Sobell, M. G. (2010). *A Practical Guide to Fedora and Red Hat Enterprise Linux*. Prentice Hall.

Es una buena guía de administración local para distribuciones Red Hat y Fedora.

[Sou] Sourceforge. "Sourceforge site". Listado de proyectos Open Source. http://sourceforge.org

[Sta02] Stallman, Richard (2002). "Discusión por Richard Stallman sobre la relación de GNU y Linux". http://www.gnu.org/gnu/linux-and-gnu.html

[Ste07] French, S. "Linux CIFS Client Guide". http://us1.samba.org/samba/ftp/cifs-cvs/linux-cifs-client-guide.pdf

[Stu] Stutz, M. "The Linux Cookbook: Tips and Techniques for Everyday Use". *The Linux Documentation Project* (guías).

Es una amplia introducción a las herramientas disponibles en GNU/Linux.

[Sun 02] Sundaram, R. (2002). "The dosemu HOWTO". *The Linux Documentation Project*.

[Tan87] Tanenbaum, Andrew (1987). *Sistemas operativos: diseño e implementación*. Prentice Hall.

[Tan06] Tanenbaum, Andrew; Woodhull, Albert S. (2006). *The Minix Book: Operating Systems Design and Implementation* (3.ª ed.). Prentice Hall.

[Tri] Tripwire.com. *Tripwire Web Site*. http://www.tripwire.com/

[Ubn] Distribución Ubuntu. http://www.ubuntu.com

[Vas00] Alavoor Vasudevan (2000). "Modem-Dialup-NT". *The Linux DocumentationProject*.

[War] Ward, I. "Debian and Windows Shared Printing mini-HOWTO". *The Linux Documentation Project*. Wine Project. http://www.winehq.com/

[Wil02] Matthew D. Wilson (2002). "VPN". *The Linux Documentation Project*.

[Wm02] Welsh, M. y otros (2002). *Running Linux 4th edition*. O'Reilly.

Administración de GNU/Linux y UNIX. Comenta de forma amplia aspectos de administración local y gestión de sistemas de impresión.

[Woo] Wood, D. "SMB HOWTO". *The Linux Documentation Project*.

[Xin] Xinetd Web Site. http://www.xinetd.org/

Introducción al sistema operativo GNU/Linux

Josep Jorba Esteve

PID_00167539

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

Los sistemas GNU/Linux [Joh98] ya no son una novedad; cuentan con una amplia variedad de usuarios y de ámbitos de trabajo donde son utilizados.

Su origen se remonta al mes de agosto de 1991, cuando un estudiante finlandés llamado Linus Torvalds anunció, en el *newsgroup* comp.os.minix que había creado su propio núcleo de sistema operativo y lo ofreció a la comunidad de desarrolladores para que lo probara y sugiriera mejoras para hacerlo más utilizable. Este sería el origen del núcleo (o *kernel*) del operativo que, más tarde, se llamaría Linux.

Nota

Podéis ver una copia del mensaje de Linus y las reacciones iniciales en http://groups.google.com/group/comp.os.minix/browse_thread/thread/76536d1fb451ac60/b813d52cbc5a044b

Por otra parte, la FSF (Free Software Foundation), mediante su proyecto GNU, producía software desde 1984 que podía ser utilizado libremente, debido a lo que Richard Stallman (miembro de la FSF) consideraba software libre: aquel del que podíamos conseguir sus fuentes (código), estudiarlas y modificarlas, y redistribuirlo sin que nos obliguen a pagar por ello. En este modelo, el negocio no está en la ocultación del código, sino en el software complementario añadido, en la adecuación del software a los clientes y en los servicios añadidos, como el mantenimiento y la formación de usuarios (el soporte que les ofrezcamos), ya sea en forma de material, libros y manuales, o en cursos de formación.

La combinación (o suma) del software GNU y del *kernel* Linux es la que nos ha traído a los actuales sistemas GNU/Linux. Actualmente, tanto los movimientos Open Source, desde diferentes organizaciones (como FSF) y empresas como las que generan las diferentes distribuciones Linux (Red Hat, Canonical Ubuntu, Mandrake, Novell SuSe...), pasando por grandes empresas (como HP, IBM o Sun, que proporcionan apoyos y/o patrocinios), han dado un empujón muy grande a los sistemas GNU/Linux hasta situarlos al nivel de poder competir, y superar, muchas de las soluciones propietarias cerradas existentes.

Los sistemas GNU/Linux no son ya una novedad. El software GNU se inició a mediados de los ochenta, y el *kernel* Linux a principios de los noventa. Linux se apoya en tecnología probada de UNIX, con más de cuarenta años de historia.

En este módulo introductorio veremos algunas ideas generales de los movimientos Open Source y Free Software, así como un poco de historia de Linux y de sus orígenes compartidos con UNIX, de donde ha heredado más de cuarenta años de investigación en sistemas operativos.

1. Software Libre y Open Source

Bajo la idea de los movimientos (o filosofías) de Software Libre y Open Source [OSIc] [OSIb] (también llamado de código abierto o software abierto), se encuentran varias formas de software que, aunque no son todas del mismo tipo, sí comparten muchas ideas comunes.

Nota

[OSIc] **OSI** (2003). "Open Source Iniciative".
http://www.opensource.org
[OSIb] **OSI** (2003). "Open Source Definition".
http://www.opensource.org/docs/definition.php

La denominación de un producto de software como "de código abierto" conlleva, como idea más importante, la posibilidad de acceder a su código fuente, y la posibilidad de modificarlo y redistribuirlo de la manera que se considere conveniente, estando sujeto a una determinada licencia de código abierto, que nos da el marco legal.

Frente a un código de tipo propietario, en el cual un fabricante (empresa de software) encierra su código, ocultándolo y restringiéndose los derechos a sí misma, sin dar posibilidad de realizar ninguna adaptación ni cambios que no haya realizado previamente la empresa fabricante, el código abierto ofrece, entre otras consideraciones:

1) Acceso al código fuente: ya sea para estudiarlo (ideal para educación) o modificarlo, para corregir errores, adaptarlo o añadir más prestaciones.

2) Gratuidad (de uso y posiblemente de precio): normalmente, el software, ya sea en forma binaria o en la forma de código fuente, puede obtenerse libremente o por una módica cantidad en concepto de gastos de empaquetamiento, distribución y valores añadidos. Lo cual no quita que el software pueda ser distribuido comercialmente a un determinado precio fijado.

3) Evitar monopolios de software propietario: no depender de una única opción o único fabricante de nuestro software. Esto es más importante cuando se trata de una gran organización, ya sea una empresa o estado, que no puede (o no debería) ponerse en manos de una determinada única solución y pasar a depender exclusivamente de ella.

4) Un modelo de avance: no basado en la ocultación de información, sino en la compartición del conocimiento (semejante al de la comunidad científica), para lograr progresos de forma más rápida, con mejor calidad, ya que las elecciones tomadas están basadas en el consenso de la comunidad, y no en los caprichos de empresas desarrolladoras de software propietario.

Crear programas y distribuirlos junto al código fuente no es nuevo. Ya desde los inicios de la informática y en los inicios de la red Internet se había hecho así. Sin embargo, el concepto de *código abierto* como tal, la definición y la redacción de las condiciones que tenía que cumplir, datan de mediados de 1997.

Eric Raymond y Bruce Perens fueron los que divulgaron la idea. Raymond [Ray98] fue el autor del ensayo titulado "La catedral y el bazar", que hablaba sobre las técnicas de desarrollo de software utilizadas por la comunidad Linux, encabezada por Linus Torvalds, y la comunidad GNU de la Free Software Foundation (FSF), encabezada por Richard Stallman. Por su parte, Bruce Perens era en aquel momento el jefe del proyecto Debian, que trabajaba en la creación de una distribución de GNU/Linux integrada únicamente con software libre.

Nota

Podéis ver la versión española de [Ray98] en http://es.tldp.org/Otros/catedral-bazar/cathedral-es-paper-00.html

Dos de las comunidades más importantes son la FSF, con su proyecto de software GNU, y la comunidad Open Source, cuyo máximo exponente de proyecto es Linux. GNU/Linux es el resultado de la unión de sus trabajos.

Una distinción importante entre las comunidades FSF y Open Source son las definiciones de *código abierto* y *software libre*. [Deba] [PS02]

El **Software Libre** (*free software*) [FSF] es un movimiento que parte de las ideas de Richard Stallman, que considera que hay que garantizar que los programas estén al alcance de todo el mundo de forma gratuita, se tenga acceso libre a éstos y puedan utilizarse al antojo de cada uno. Una distinción importante, que causó ciertas reticencias a las empresas, es el término *free*. En inglés, esta palabra tiene el doble significado de 'gratuito' y 'libre'. La gente de la FSF buscaba las dos cosas, pero era difícil vender ambos conceptos a las empresas. La pregunta típica era: ¿cómo se podía ganar dinero con esto? La respuesta vino de la comunidad Linux (con Linus Torvalds a la cabeza), cuando consiguió tener un producto que todavía no había logrado la comunidad GNU y la FSF en esos momentos: un sistema operativo completo libre con código fuente disponible. En este momento fue cuando a una parte de la comunidad se le ocurrió juntar las diferentes actividades que había en la filosofía del Software Libre bajo la nueva denominación de *código abierto* (*open source*).

Open Source se registró como una marca de certificación, a la que podían adherirse los productos software que respetasen sus especificaciones. Esto no gustó a todo el mundo; de hecho, suele haber cierta separación y controversias entre los dos grupos del Open Source y la FSF (con GNU), aunque son más las cosas que los unen que las que los separan.

En cierta manera, para los partidarios del software libre (como la FSF), el código abierto (u *open source*) constituye un paso en falso, ya que representa una cierta "venta" al mercado de sus ideales, y deja la puerta abierta a que se vaya haciendo propietario el software que era libre. Los partidarios de *open source* ven la oportunidad de promocionar el software que, de otro modo, estaría en una utilización minoritaria, mientras que con la divulgación y la puesta en común para todo el mundo, incluidas empresas que quieran participar en código abierto, entramos con suficiente fuerza para plantar cara al software propietario.

La idea que persiguen tanto FSF como Open Source es la de aumentar la utilidad del software libre, ofreciendo así una alternativa a las soluciones únicas que las grandes empresas quieren imponer. Las diferencias entre ambas filosofías son más ideológicas que prácticas.

Una vez establecidas las ideas básicas de la comunidad del código abierto, llegamos al punto en que había que concretar de manera clara qué criterios tenía que cumplir un producto de software para considerarse de código abierto. Se tenía que contar con una definición de código abierto [OSIb], que inicialmente escribió Bruce Perens en junio de 1997 como resultado de comentarios de los desarrolladores de la distribución Debian Linux, y que posteriormente fue reeditada (con modificaciones menores) por la organización OSI (Open Source Initiative). Esta organización está encargada de regular la definición y controlar las licencias de código abierto.

El código abierto está regulado por una definición pública que se utiliza como base de la redacción de sus licencias de software.

Un pequeño resumen (interpretación) de la definición: un Open Source Software [OSIb], o software de código fuente abierto, debe cumplir los requisitos siguientes:

1) Se puede copiar, regalar o vender a terceros el software, sin tener que pagar a nadie por ello. Se permite copiar el programa.

2) El programa debe incluir el código fuente y tiene que permitir la distribución tanto en forma compilada como en fuente. O, en todo caso, hay que facilitar algún modo de obtener los códigos fuente (por ejemplo, descarga desde Internet). No está permitido ocultar el código, o darlo en representaciones intermedias. Garantiza que se pueden hacer modificaciones.

3) La licencia del software tiene que permitir que se puedan realizar modificaciones y trabajos que se deriven, y que entonces se puedan distribuir bajo la misma licencia que la original. Permite reutilizar el código original.

4) Puede requerirse la integridad del código del autor, o sea, las modificaciones se pueden presentar en forma de parches (*patchs*) al código original, o se puede pedir que tengan nombres o números distintos a los originales. Esto protege al autor de qué modificaciones puedan considerarse como suyas. Este punto depende de lo que diga la licencia del software.

5) La licencia no debe discriminar a ninguna persona o grupo. No se debe restringir el acceso al software. Un caso aparte son las restricciones por ley, como las de las exportaciones tecnológicas fuera de Estados Unidos a terceros países. Si existen restricciones de este tipo, hay que mencionarlas.

6) No debe discriminar campos laborales. El software puede utilizarse en cualquier ambiente de trabajo, aunque no haya estado pensado para él. Otra lectura es permitir fines comerciales; nadie puede impedir que el software se utilice con fines comerciales.

7) La licencia es aplicable a todo el mundo que reciba el programa.

8) Si el software forma parte de producto mayor, debe permanecer con la misma licencia. Esto controla que no se separen partes para formar software propietario (de forma no controlada). En el caso de software propietario, hay que informar que hay partes (y cuáles) de software de código abierto.

9) La licencia no debe restringir ningún software incorporado o distribuido conjuntamente, o sea, incorporarlo no debe suponer ninguna barrera para otro producto de software distribuido conjuntamente. Este es un punto "polémico", ya que parece contradecirse con el anterior. Básicamente, dice que cualquiera puede coger software de código abierto y añadirlo al suyo sin que afecte a las condiciones de su licencia (por ejemplo, propietaria), aunque sí que, según el punto anterior, tendría que informar que existen partes de código abierto.

10) La licencia tiene que ser tecnológicamente neutra. No deben mencionarse medios de distribución únicos, o excluirse posibilidades. Por ejemplo, no puede limitarse (por licencia) que se haga la distribución en forma de CD, ftp o mediante web.

Nota

Esta definición de código abierto no es por sí misma una licencia de software, sino más bien una especificación de qué requisitos debería cumplir una licencia de software de código abierto.

La licencia que traiga el programa tiene que cumplir las especificaciones anteriores para que el programa se considere de código abierto. La organización OSI se encarga de comprobar que las licencias cumplen las especificaciones.

En la página web de Open Source Licenses se puede encontrar la lista de las licencias [OSIa], siendo una de las más famosas y utilizadas las GPL (GNU Public Licenses).

Nota

[OSIa] OSI. "Listado de licencias Open Source".
http://www.opensource.org/licenses/index.html

Bajo GPL, el software puede ser copiado y modificado, pero las modificaciones deben hacerse públicas bajo la misma licencia. Y se impide que el código se mezcle con código propietario, para evitar así que el código propietario se haga con partes abiertas. Existe una licencia LGPL que es prácticamente igual, pero permite que software con esta licencia sea integrado en software propietario. Un ejemplo clásico es la biblioteca (*library*) C de Linux (con licencia LGPL). Si ésta fuera GPL, sólo podría desarrollarse software libre, con la LGPL se permite usarlo para desarrollar software propietario.

Muchos proyectos de software libre, o con parte de código abierto y parte propietario, tienen su propia licencia: Apache (basada en la BSD), Mozilla (MPL y NPL de Netscape), etc. Básicamente, a la hora de poner el software como *open source* podemos poner nuestra propia licencia que cumpla la definición anterior (de código abierto), o podemos escoger licenciar bajo una licencia ya establecida o, como en el caso de la GPL, nos obliga a que nuestra licencia también sea GPL.

Una vez vistos los conceptos de código abierto y sus licencias, nos queda por tratar **hasta qué punto es rentable para una empresa trabajar o producir código abierto**. Si no fuera atrayente para las empresas, perderíamos a la vez tanto un potencial cliente como uno de los principales productores de software.

El código abierto es también atrayente para las empresas, con un modelo de negocio donde se prima el valor añadido al producto.

En el código abierto existen diferentes rentabilidades atrayentes de cara a las empresas:

a) Para las empresas desarrolladoras de software, se crea un problema: ¿cómo es posible ganar dinero sin vender un producto? Se gasta mucho dinero en desarrollar un programa y después es necesario obtener beneficios. Bien, la respuesta no es simple, no se puede conseguir con cualquier software, la rentabilidad se encuentra en el tipo de software que puede generar beneficios más allá de la simple venta. Normalmente, hay que hacer un estudio de si la aplicación se tornará rentable al desarrollarla como software abierto (la mayoría sí que lo hará), basándose en las premisas de que tendremos un descenso de gasto en desarrollo (la comunidad nos ayudará), reducción de mantenimiento o corrección de errores (la comunidad puede ofrecer esto muy rápido), y tener en cuenta el aumento de número de usuarios que nos proporcionará el código

abierto, así como las necesidades que tendrán de nuestros servicios de apoyo o documentación. Si la balanza es positiva, entonces será viable prescindir de los ingresos generados por las ventas.

b) Aumentar la cuota de usuarios.

c) Obtener mayor flexibilidad de desarrollo; cuantas más personas intervienen, más gente habrá para detectar errores.

d) Los ingresos en su mayor parte vendrán por el lado del apoyo, formación de usuarios y mantenimiento.

e) En empresas que utilizan software hay que considerar muchos parámetros a la hora de escoger el software para el desarrollo de las tareas; cabe tener en cuenta cosas como rendimiento, fiabilidad, seguridad, escalabilidad y coste monetario. Y aunque parece que el código abierto ya supone de por sí una elección por el coste económico, hay que decir que existe software abierto que puede competir con (o incluso superar) el propietario en cualquiera de los otros parámetros. Además, hay que vigilar mucho con las opciones o sistemas propietarios de un único fabricante; no podemos depender únicamente de ellos (podemos recordar casos, en otros ámbitos, como los vídeos beta de Sony frente a VHS, o en los PC la arquitectura MicroChannel de IBM). Tenemos que evitar el uso de monopolios, con lo que éstos suponen: falta de competencia en los precios, servicios caros, mantenimiento caro, poca (o nula) variedad de opciones, etc.

f) Para los usuarios particulares ofrece gran variedad de software adaptado a tareas comunes, ya que buena parte del software ha sido pensado e implementado por personas que querían hacer esas mismas tareas pero no encontraban el software adecuado. En el caso del usuario particular, un parámetro muy importante es el coste del software, pero la paradoja es que en el usuario doméstico es donde se hace más uso de software propietario. Normalmente, los usuarios domésticos hacen uso de productos de software con copias ilegales. Algunas estadísticas recientes indican índices del 60-70% de copias ilegales domésticas en algunos países. El usuario siente que sólo por tener el ordenador doméstico PC ya tiene "derecho" a disponer de software para usarlo. En estos casos estamos bajo situaciones "ilegales" que, aunque no han sido ampliamente perseguidas, pueden serlo en su día, o bien se intentan controlar por sistemas de licencias (o activaciones de productos). Además, esto tiene unos efectos perjudiciales indirectos sobre el software libre, debido a que si los usuarios hacen un uso amplio de software propietario, esto obliga a quien se quiera comunicar con ellos, ya sean bancos, empresas o administraciones públicas, a hacer uso del mismo software propietario, y ellos sí que abonan las licencias a los productos. Una de las "batallas" más importantes para el software libre es la posibilidad de captar a los usuarios domésticos, lo que se denomina mercado *desktop* (o escritorio), referido al uso doméstico o de oficina en las empresas.

Nota

Las copias ilegales domésticas también son denominadas a veces *copias piratas*.

g) Por último, **los Estados**, como caso particular, pueden obtener beneficios importantes del software de código abierto, ya que pueden disponer de software de calidad a precios "ridículos" comparados con el enorme gasto de licencias de software propietario (miles o decenas de miles). Además de que el software de código abierto permite integrar fácilmente a las aplicaciones, hay que tener en cuenta cuestiones culturales (de cada país) como, por ejemplo, la lengua. Este último caso es bastante problemático, ya que en determinadas regiones, estados pequeños con lengua propia, los fabricantes de software propietario se niegan a adaptar sus aplicaciones, o instan a que se les pague por hacerlo.

2. UNIX. Un poco de historia

Como antecesor de nuestros sistemas GNU/Linux [Sta], vamos a recordar un poco la historia de UNIX [Sal94] [Lev]. En origen, Linux se pensó como un clon de Minix (una implementación académica de UNIX para PC) y de algunas ideas desarrolladas en los UNIX propietarios. Pero, a su vez, se desarrolló en código abierto, y con orientación a los PC domésticos. Veremos, en este apartado dedicado a UNIX y en el siguiente, dedicado a GNU/Linux, cómo esta evolución nos ha llevado hasta los sistemas GNU/Linux actuales que pueden competir con cualquier UNIX propietario y que están disponibles para un amplio número de arquitecturas hardware, desde el simple PC hasta los supercomputadores.

Linux puede ser utilizado en un amplio rango de máquinas. En la lista TOP500 de los supercomputadores más rápidos pueden encontrarse varios supercomputadores con GNU/Linux: por ejemplo, el MareNostrum, en el Barcelona Supercomputing Center, un *cluster* diseñado por IBM, con 10240 CPUs PowerPC/Cell con sistema operativo GNU/Linux (adaptado para los requerimientos de tales máquinas). En las estadísticas de la lista podemos observar que los supercomputadores con GNU/Linux ocupan, en general, más de un 75% de la lista.

Nota

Podemos ver la lista TOP500 de los supercomputadores más rápidos en:
http://www.top500.org

UNIX se inició hacia el año 1969 en los laboratorios BTL (Bell Telephone Labs) de AT&T. Éstos se acababan de retirar de la participación de un proyecto llamado MULTICS, cuyo objetivo era crear un sistema operativo con el cual un gran ordenador pudiera dar cabida a un millar de usuarios simultáneos. En este proyecto participaban los BTL, General Electric y el MIT. Pero falló, en parte, por ser demasiado ambicioso para su época.

Mientras se desarrollaba este proyecto, dos ingenieros de los BTL que participaban en MULTICS, **Ken Thompson y Dennis Ritchie**, encontraron un ordenador que no estaba utilizando nadie, un DEC PDP7, que sólo tenía un ensamblador y un programa cargador. Thompson y Ritchie desarrollaron como pruebas (y a menudo en su tiempo libre) partes de UNIX, un programa ensamblador (del código máquina) y el núcleo rudimentario del sistema operativo.

Ese mismo año, 1969, Thompson tuvo la idea de escribir un sistema de ficheros para el núcleo creado, de manera que se pudiesen almacenar ficheros de forma ordenada en un sistema de directorios jerárquicos. Después de unas cuantas discusiones teóricas (que se alargaron unos dos meses) se implementó el sistema en un par de días. A medida que se avanzaba en el diseño del sistema, en el cual se incorporaron algunos ingenieros más de los BTL, la máquina original

se les quedó pequeña, y pensaron en pedir una nueva (en aquellos días costaban cerca de 100.000 dólares, era una buena inversión). Tuvieron que inventarse una excusa (ya que el sistema UNIX era un desarrollo en tiempo libre) y dijeron que la querían para crear un nuevo procesador de texto (aplicación que daba dinero en aquellos tiempos). Se les aprobó la compra de una PDP11.

UNIX se remonta al año 1969, así que cuenta con más de cuarenta años de tecnologías desarrolladas y utilizadas en todo tipo de sistemas.

Cuando les llegó la máquina, sólo estaba la CPU y la memoria, pero no el disco ni el sistema operativo. Thompson, sin poder esperar, diseñó un disco RAM en memoria y utilizó la mitad de la memoria como disco y la otra para el sistema operativo que estaba diseñando. Una vez llegó el disco, se siguió trabajando tanto en UNIX como en el procesador de textos prometido (la excusa). El procesador de textos fue un éxito (se trataba de Troff, un lenguaje de edición que posteriormente fue utilizado para crear las páginas *man* de UNIX), y los BTL comenzaron a utilizar el rudimentario UNIX con el nuevo procesador de texto, de manera que se convirtieron en el primer usuario de UNIX.

En aquellos momentos comenzaron a presentarse varios principios filosóficos de UNIX [Ray02a]:

- Escribir programas para hacer una cosa y hacerla bien.
- Escribir programas para que trabajaran juntos.
- Escribir programas para que manejaran flujos de texto.

Otra idea muy importante radicó en que UNIX fue uno de los primeros **sistemas pensados para ser independiente de la arquitectura hardware**, y que ha permitido portarlo con éxito a un gran número de arquitecturas hardware diferentes.

La necesidad de documentar lo que se estaba haciendo, ya que había usuarios externos, dio lugar en noviembre de 1971 al *UNIX Programmer's Manual*, que firmaron Thompson y Richie. En la segunda edición (junio de 1972), denominada V2 (se hacía corresponder la edición de los manuales con el número de versión UNIX), se decía que el número de instalaciones de UNIX ya llegaba a las diez. Y el número siguió creciendo hasta cincuenta en la V5.

Entonces se decidió (finales de 1973) presentar los resultados en un congreso de sistemas operativos. Y como resultado, varios centros informáticos y universidades pidieron copias de UNIX. AT&T no daba apoyo ni mantenimiento de UNIX, lo que hizo que los usuarios necesitaran unirse y compartir sus conocimientos para formar comunidades de usuarios de UNIX. AT&T decidió

Nota

Ved: http://www.usenix.org

ceder UNIX a las universidades, pero tampoco les daba apoyo ni corrección de errores. Los usuarios comenzaron a compartir sus ideas, información programas, *bugs*, etc. Se creó una asociación denominada USENIX como agrupación de usuarios de UNIX. Su primera reunión (mayo de 1974) tuvo una docena de asistentes.

Una de las universidades que había obtenido una licencia de UNIX fue la Universidad de California en Berkeley, donde había estudiado Ken Thompson. En 1975, Thompson volvió como profesor allí, y trajo consigo la última versión de UNIX. Dos estudiantes graduados recién incorporados, Chuck Haley y Bill Joy (que posteriormente cofundó SUN Microsystems) comenzaron a trabajar en una implementación de UNIX.

Una de las primeras cosas que les decepcionó eran los editores. Joy perfeccionó un editor llamado *ex*, hasta transformarlo en el *vi*, un editor visual a pantalla completa. Y los dos escribieron un compilador de lenguaje Pascal, que añadieron a UNIX. Hubo cierta demanda de esta implementación de UNIX, y Joy lo comenzó a producir como el BSD, Berkeley Software Distribution (o UNIX BSD).

BSD (en 1978) tenía una licencia particular sobre su precio: decía que estaba acorde con el coste de los medios y la distribución que se tenía en ese momento. Así, los nuevos usuarios acababan haciendo algunos cambios o incorporando cosas, vendiendo sus copias "rehechas" y, al cabo de un tiempo, los cambios se incorporaban en la siguiente versión de BSD.

Joy también realizó en su trabajo del editor *vi* algunas aportaciones más, como el tratamiento de los terminales de texto, de manera que el editor fuera independiente del terminal en que se utilizase. Creó el sistema *termcap* como interfaz genérica de terminales con controladores para cada terminal concreto, de manera que en la realización de los programas ya nos podíamos olvidar de los terminales concretos y utilizar la interfaz genérica.

Un siguiente paso fue adaptarlo a diferentes arquitecturas. Hasta el año 1977 sólo se podía ejecutar en máquinas PDP; en ese año se comenzaron a hacer adaptaciones para máquinas del momento, como las Interdata e IBM. La versión 7 (V7 en junio de 1979) de UNIX fue la primera portable. Esta versión trajo muchos avances, ya que contenía *awk*, *lint*, *make*, *uucp*. El manual ya tenía 400 páginas (más dos apéndices de 400 cada uno). Se incluía también el compilador de C diseñado en los BTL por Kernighan y Ritchie, que se había creado para reescribir la mayor parte de UNIX, inicialmente en ensamblador y luego pasado a C con las partes de ensamblador que fuesen sólo dependientes de la arquitectura. Se incluyeron también una *shell* mejorada (*shell* de Bourne) y comandos como *find*, *cpio* y *expr*.

La industria UNIX comenzó también a crecer; empezaron a aparecer versiones (implementaciones) de UNIX por parte de compañías, como Xenix; hubo una colaboración entre Microsoft (en los orígenes también trabajó con versiones de UNIX) y SCO para máquinas Intel 8086 (el primer PC de IBM); aparecieron nuevas versiones BSD de Berkeley...

Pero surgió un nuevo problema cuando AT&T se dio cuenta de que UNIX era un producto comercial valioso. En la licencia de la V7 se prohibió el estudio en centros académicos, para proteger el secreto comercial. Muchas universidades utilizaban hasta el momento el código fuente de UNIX para docencia de sistemas operativos, y dejaron de usarlo para dar sólo teoría.

En cualquier caso, cada uno solucionó el problema a su modo. En Ámsterdam, **Andrew Tanenbaum** (autor de prestigio de libros de teoría de sistema operativos) decidió escribir desde el principio un nuevo sistema operativo compatible con UNIX sin utilizar una sola línea de código de AT&T; llamó a este nuevo operativo **Minix**. Éste sería el que, posteriormente, le serviría en 1991 a un estudiante finlandés para crear su propia versión de UNIX, que llamó Linux.

Bill Joy, que continuaba en Berkeley desarrollando BSD (ya estaba en la versión 4.1), decidió marcharse a una nueva empresa llamada SUN Microsystems, en la cual acabó los trabajos del 4.2BSD, que después acabaría modificando para crear el UNIX de SUN, el SunOS (hacia 1983). Cada empresa comenzó a desarrollar sus versiones: IBM con AIX, DEC con Ultrix, HP con HPUX, Microsoft/SCO con Xenix, etc. UNIX comenzó (desde 1980) su andadura comercial, AT&T sacó una ultima versión llamada UNIX SystemV (SV), de la cual derivan, junto con los 4.xBSD, los UNIX actuales, ya sea de la rama BSD o de la SystemV. La SV tuvo varias revisiones, por ejemplo, la SV Release 4 fue una de las más importantes. La consecuencia de estas últimas versiones es que más o menos todos los UNIX existentes se adaptaron uno al otro; en la práctica son versiones del SystemV R4 de AT&T o del BSD de Berkeley, adaptadas por cada fabricante. Algunos fabricantes lo especifican y dicen que su UNIX es de tipo BSD o SystemV, pero la realidad es que todos tienen un poco de las dos, ya que posteriormente se hicieron varios estándares de UNIX para intentar uniformizarlos. Entre ellos encontramos los IEEE POSIX, UNIX97, FHS, etc.

Con el tiempo, UNIX se dividió en varias ramas de sistema, siendo las dos principales la que derivaba del AT&T UNIX o SystemV, y la de la Universidad de California, el BSD. La mayoría de UNIX actuales deriva de uno u otro, o son una mezcla de los dos.

Pero AT&T en aquellos momentos (SVR4) pasó por un proceso judicial por monopolio telefónico (era la principal, si no la única, compañía telefónica en Estados Unidos), que hizo que se dividiera en múltiples empresas más pequeñas, y los derechos de UNIX originales comenzaron un baile de propietarios

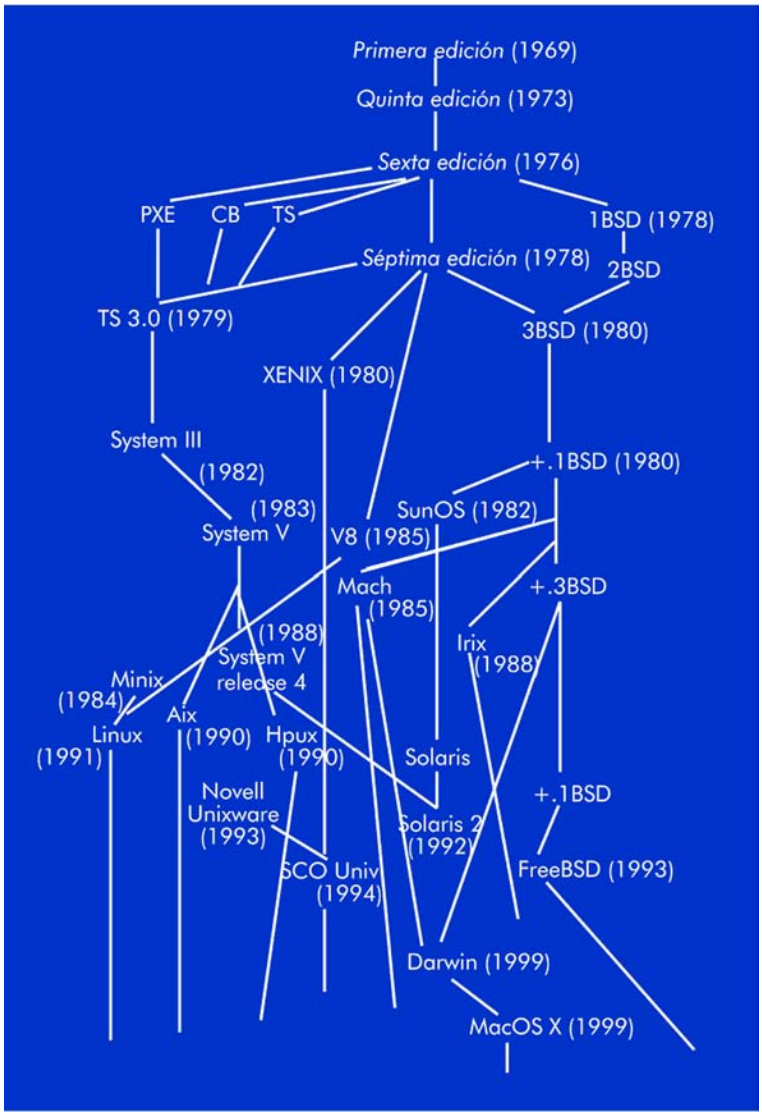
importante: en 1990 los tenían a medias el Open Software Foundation (OSF) y UNIX International (UI), después, UNIX Systems Laboratories (USL), que denunció a la Universidad de Berkeley por sus copias del BSD, pero perdió, ya que la licencia original no imponía derechos de propiedad al código de UNIX. Más tarde, los derechos UNIX se vendieron a la empresa Novell; ésta cedió parte a SCO (que ya disponía de algunos cedidos por Microsoft desde sus productos Xenix), y hoy en día aún no está muy claro quién los tiene finalmente: por diferentes frentes los reclaman Novell, la OSF y SCO.

Un ejemplo de esta problemática fue el caso (2003-10) de la compañía SCO, que puso una demanda legal a IBM porque ésta, según SCO, había cedido parte del código UNIX a versiones del *kernel* Linux, que supuestamente incluyen algún código UNIX original. El resultado a día de hoy es que el asunto aún continúa con cierta vigencia en los tribunales, con SCO convertida en un "paria" de la industria informática que amenaza a los usuarios Linux, IBM y otros UNIX propietarios, con la afirmación de que tienen los derechos UNIX originales, y de que los demás tienen que pagar por ellos. Aunque en los últimos movimientos judiciales, parece ser que finalmente Novell es la que posee los derechos de propiedad intelectual de UNIX. Habrá que ver cómo evoluciona este caso, y el tema de los derechos UNIX con él.

Nota

Podéis ver la opinión de la FSF sobre el caso SCO en http://www.gnu.org/philosophy/sco/sco.html

Figura 1. Resumen histórico de varias versiones UNIX



El panorama actual de UNIX ha cambiado mucho desde la **aparición de Linux** (1991), que a partir de los años 1995-99 comenzó a convertirse en una alternativa seria a los UNIX propietarios, por la gran cantidad de plataformas hardware que soporta y el amplio apoyo de la comunidad internacional y empresas en el avance. Hay diferentes versiones UNIX propietarias que siguen sobreviviendo en el mercado, tanto por su adaptación a entornos industriales o por ser el mejor operativo existente en el mercado, como porque hay necesidades que sólo pueden cubrirse con UNIX y el hardware adecuado. Además, algunos de los UNIX propietarios todavía son mejores que GNU/Linux en cuanto a fiabilidad y rendimiento, aunque cada vez acortando distancias, ya que las mismas empresas que tienen sus UNIX propietarios se interesan cada vez más en GNU/Linux, y aportan parte de sus desarrollos para incorporarlos a Linux. Es de esperar una muerte más o menos lenta de las versiones propietarias de UNIX, hacia distribuciones basadas en Linux de los fabricantes, adaptadas a sus equipos.

Un panorama general de estas empresas:

Nota

Muchas de las empresas que disponen de UNIX propietarios participan en GNU/Linux y ofrecen algunos de sus desarrollos a la comunidad.

- **SUN**: dispone de su implementación de UNIX llamada Solaris (evolución del SunOS). Comenzó como un sistema BSD, pero ahora es mayoritariamente SystemV y partes de BSD. Es muy utilizado en las máquinas Sun con arquitectura Sparc, y en máquinas multiprocesador (hasta unos 64 procesadores). Promocionan GNU/Linux como entorno de desarrollo para Java, y dispusieron de una distribución de GNU/Linux denominada Java Desktop System, que tuvo una amplia aceptación en algunos países. Además, comenzó a usar Gnome como escritorio, y ofrece apoyo financiero a varios proyectos como Mozilla Firefox, Gnome y OpenOffice. También cabe destacar la iniciativa tomada con su última versión de su UNIX Solaris, para liberar su código casi totalmente, en la versión Solaris 10, creando una comunidad para las arquitecturas intel y Sparc, denominada OpenSolaris, que ha permitido la creación de distribuciones libres de Solaris. Además tenemos que señalar iniciativas (2006) para liberar la plataforma Java bajo licencias GPL, como el proyecto OpenJDK. Asimismo, la adquisición de Sun Microsystems por parte de Oracle (2009) está causando cierta incertidumbre en algunos de estos productos y/o tecnologías, por no haber definido la compañía una estrategia clara para ellos.

- **IBM**: tiene su versión de UNIX propietaria denominada AIX, que sobrevive en algunos segmentos de gama alta de estaciones de trabajo y servidores de la firma. Por otra parte, presta apoyo firme a la comunidad Open Source, promoviendo entornos de desarrollo libres (eclipse.org) y tecnologías Java para Linux, incorpora Linux a sus grandes máquinas y diseña campañas publicitarias (marketing) para promocionar Linux. Aparte está teniendo una repercusión importante en la comunidad, por el ambiente judicial de su caso defendiéndose de la firma SCO, que la acusa de violación de propiedad intelectual UNIX, por haber, supuestamente, integrado componentes en GNU/Linux.

- **HP**: tiene su UNIX HPUX, pero da amplio soporte a Linux, tanto en forma de código en Open Source como instalando Linux en sus máquinas. Se dice que es la compañía que ha ganado más dinero con Linux.

- **SGI**: Silicon Graphics tiene un UNIX llamado IRIX para sus máquinas gráficas basadas en arquitecturas MIPS. En el 2006 cambió su estrategia hacia plataformas Intel Xeon/AMD Opteron vendiendo máquinas con Windows, y últimamente la mayoría de los sistemas con versiones comerciales de GNU/Linux, como las producidas por Red Hat y Novell SUSE. A la comunidad Linux le ofrece soporte de OpenGL (tecnología de gráficos 3D) y de diferentes sistemas de ficheros (XFS) y control de dispositivos periféricos.

- **Apple**: se incorporó recientemente (a partir de mediados de los noventa) al mundo UNIX, cuando decidió sustituir su operativo por una variante

UNIX. El núcleo de sus nuevos operativos, llamado XNU (combinado en el sistema operativo Darwin) proviene de una versión 4.4BSD, combinada con *kernel* Mach. Este núcleo Open Source será el que, sumado a unas interfaces gráficas muy potentes, dé a Apple su sistema operativo MacOS X. Está considerado hoy en día como uno de los mejores UNIX y, como mínimo, uno de los más "bellos" en aspecto gráfico. También emplea gran cantidad de software provenientes del proyecto GNU como utilidades de sistema y librerías y compiladores de desarrollo (GNU gcc).

- **Distribuidores Linux**: tanto comerciales como organizaciones, mencionaremos a empresas como Red Hat, SuSe, Mandriva (previamente conocida como Mandrake), y organizaciones/comunidades no comerciales como Debian, Fedora, OpenSUSE, etc... Entre éstas (las distribuciones con mayor despliegue) y las más pequeñas, se llevan el mayor desarrollo de GNU/Linux, y tienen el apoyo de la comunidad Linux y de la FSF con el software GNU, además de recibir contribuciones de las citadas empresas.

- **BSD**: aunque no sea una empresa como tal, mencionaremos cómo desde Berkeley y otros intermediarios se continúa el desarrollo de las versiones BSD, así como otros proyectos libres, clones de BSD, como los operativos FreeBSD, netBSD, OpenBSD (el UNIX considerado más seguro), TrustedBSD, etc., que también, más tarde o más temprano, suponen mejoras o incorporaciones de software a Linux. Además, una aportación importante en la línea BSD es el *kernel* XNU (usado en Darwin) proveniente de 4.4BSD, y que desarrolló Apple como núcleo de código abierto de su sistema operativo MacOS X y su sistema IphoneOS para móviles.

- **Google**: ha tenido una relación bastante importante con la comunidad desde sus inicios como buscador en Internet, ya que toda su infraestructura de *clusters* en diferentes centros de datos, a veces conocida como Google Cluster, está basada en múltiples servidores corriendo GNU/Linux con sistemas de ficheros especialmente diseñados para grandes volúmenes de datos. Asimismo, entró con especial fuerza en el mundo del desarrollo para plataformas móviles, con la plataforma Android, una plataforma con *kernel* Linux, y capas software basadas en GNU y Java.

- **Microsoft**: Siempre ha tenido una relación difícil con el código abierto abierto/libre, al que ve como un competidor y un peligro potencial. Normalmente, ha supuesto más para la comunidad un entorpecimiento en el desarrollo de UNIX y GNU/Linux, ya que ha puesto trabas con incompatibilidades en diferentes tecnologías, y no dispone de una participación directa en el mundo UNIX/Linux. Aunque en sus orígenes desarrolló Xenix (1980) para PC, a partir de una licencia AT&T de UNIX, que no vendió directamente, pero sí lo hizo por medio de intermediarios, como SCO, que se hizo con su control en 1987 y la renombró como SCO UNIX (1989). Como nota curiosa, posteriormente compró parte de derechos de la licencia UNIX a SCO (ésta los había obtenido a su vez por medio de Novell).

Nota

Carta abierta de Novell a la comunidad GNU/Linux en http://www.novell.com/linux/microsoft/community_open_letter.html

No están claros los motivos de Microsoft a la hora de realizar esta adquisición, aunque algunos sugieren que existe alguna relación con el hecho de proporcionar apoyo a SCO en su juicio contra IBM. Además, recientemente (2006), Microsoft llegó a acuerdos con Novell (actual proveedora de la distribución SuSe y la comunidad OpenSuse), en una serie de acuerdos bilaterales para promocionar empresarialmente ambas plataformas. Pero parte de la comunidad GNU/Linux se mantiene escéptica, por las posibles implicaciones sobre la propiedad intelectual de Linux, y los temas que podrían incluir problemas judiciales por uso de patentes.

Otra anécdota histórica curiosa (2002) es que, junto con una empresa llamada UniSys, Microsoft se dedicó a realizar marketing de cómo convertir sistemas UNIX a sistemas Windows. Y aunque el objetivo podía ser más o menos loable, lo curioso era que el servidor original de la web empresarial estaba en una máquina FreeBSD con Apache. En ocasiones, también paga a algunas empresas "independientes" (algunos opinan que bastante poco) para que lleven a cabo estudios de rendimiento comparativos entre UNIX/Linux y Windows (muchas de estas campañas, son vistas como F.U.D, ya que la mayoría de estas campañas no resisten un mínimo análisis técnico, mas allá del marketing empleado). En los últimos años parece que en Microsoft haya habido cierto acercamiento a la comunidad Open Source, estableciendo diferentes comunidades OpenSource, para la integración y interrelación con productos GNU/Linux.

Nota

Algunos portales de Microsoft relacionados con Open Source son http://port25.technet.com/ y http://www.microsoft.com/opensource

Ved también la definición de FUD en Wikipedia, http://en.wikipedia.org/wiki/Fear,_uncertainty_and_doubt, y un caso concreto de FUD en http://news.cnet.com/2100-1001-872266.html

Como resumen general, algunos comentarios que suelen aparecer en la bibliografía UNIX apuntan a que UNIX es, técnicamente, un sistema sencillo y coherente diseñado con buenas ideas que se supieron llevar a la práctica, pero no hay que olvidar que algunas de estas ideas se consiguieron gracias al apoyo entusiasta que brindó una gran comunidad de usuarios y desarrolladores que colaboraron entre sí, compartiendo una tecnología y gobernando su evolución.

Y como la historia se suele repetir, en este momento la evolución y el entusiasmo continúan con los sistemas GNU/Linux.

3. Sistemas GNU/Linux

En los primeros años, los usuarios de los primeros ordenadores personales no disponían de muchos sistemas operativos donde elegir.

El mercado de los ordenadores personales lo dominaba un DOS de Microsoft. Otra posibilidad eran los Mac de Apple, pero a unos precios desorbitados (en comparación) con el resto. La otra opción importante, aunque reservada a grandes (y caras) máquinas, era UNIX.

Una primera opción que apareció fue **MINIX** (1984), creado desde cero por Andrew Tanenbaum, **con el objetivo de usarlo para la educación**, para enseñar diseño e implementación de sistemas operativos.

MINIX fue pensado para ejecutarse sobre una plataforma Intel 8086, muy popular en la época porque era la base de los primeros IBM PC. La principal ventaja de este operativo radicaba en su código fuente, accesible a cualquiera (12.000 líneas de código entre ensamblador y C), ya que estaba incluido en el libro docente de sistemas operativos de Tanenbaum [Tan87]. Pero MINIX era más una herramienta de enseñanza que un sistema eficaz pensado para el rendimiento o para actividades profesionales.

En los noventa, la FSF (Free Software Foundation) y su proyecto GNU motivó a muchos programadores para promover el software de calidad y de distribución libre. Y aparte de software de utilidades, se trabajaba en un núcleo (*kernel*) de operativo denominado HURD, que llevaría varios años de desarrollo.

Mientras, en octubre de 1991, un estudiante finlandés llamado Linus Torvalds presentaría la versión 0.0.1 de su *kernel* de sistema operativo, que denominó **Linux**, orientado a máquinas Intel con 386, y lo ofreció bajo licencia GPL a foros de programadores y a la comunidad de Internet para que lo probaran y, si les gustaba, le ayudaran a su desarrollo. El entusiasmo fue tal que, en poco tiempo, había un gran número de programadores trabajando en el núcleo o en aplicaciones para él.

Algunas de las características que diferenciaron a Linux de los sistemas de su tiempo y que siguen siendo aplicables, y otras heredadas de UNIX, podrían ser:

a) Sistema operativo de código abierto: cualquiera puede disponer de sus fuentes, modificarlas y crear nuevas versiones que poder compartir bajo la licencia GPL (que, de hecho, lo convierte en un software libre).

b) Portabilidad: tal como el UNIX original, Linux está pensado para depender muy poco de una arquitectura concreta de máquina. Consecuentemente Linux es, en su mayor parte, independiente de la máquina de destino y puede portarse a casi cualquier arquitectura que disponga de un compilador C como el GNU *gcc*. Sólo restan algunas pequeñas partes de código ensamblador y de algunos dispositivos dependientes de la máquina, que tienen que ser rescritas en cada nueva arquitectura. Gracias a esto, GNU/Linux es uno de los sistemas operativos que corre en mayor número de arquitecturas: Intel x86 y IA64, AMD x86 y x86_64, Sparc de Sun, MIPS de Silicon, PowerPC (Apple), IBM S390, Alpha de Compaq, m68k Motorola, Vax, ARM, HPPArisc...

c) *Kernel* de tipo monolítico: el diseño del *kernel* está unido en una sola pieza, pero es conceptualmente modular en las diferentes tareas. Otra escuela de diseño de operativos propone los *microkernel* (un ejemplo es el proyecto Mach), donde los servicios se implementan como procesos aparte, comunicados por un (micro) *kernel* más básico. Linux se decidió como monolítico, porque es difícil extraer buen rendimiento de los *microkernels* (resulta un trabajo bastante duro y complejo). Por otra parte, el problema de los monolíticos es el crecimiento; cuando se vuelven muy grandes se vuelven intratables en el desarrollo; esto se intentó solucionar con los módulos de carga dinámica.

Nota

Proyecto original Mach:
http://www.cs.cmu.edu/afs/cs/project/mach/public/www/mach.html

d) Módulos dinámicamente cargables: permiten poner partes del sistema operativo, como *filesystems*, o controladores de dispositivos, como porciones externas que se cargan (o enlazan) con el *kernel* en tiempo de ejecución bajo demanda. Esto permite simplificar el *kernel* y ofrecer estas funcionalidades como elementos que se pueden programar por separado. Con este uso de módulos, se podría considerar a Linux como un *kernel* mixto, ya que es monolítico, pero ofrece una serie de módulos que complementan el *kernel* (aproximación parecida a algunos conceptos de *microkernel*).

e) Desarrollo del sistema por una comunidad vinculada por Internet: los sistemas operativos nunca habían tenido un desarrollo tan amplio y disperso; no suelen salir de la compañía que los elabora (en el caso propietario) o de un pequeño conjunto de instituciones académicas y laboratorios que colaboran para crear uno. El fenómeno de la comunidad Linux permite que cada uno colabore en la medida que el tiempo y sus propios conocimientos se lo permitan. El resultado son de cientos a miles de desarrolladores para Linux. Además, por su naturaleza de sistema de código fuente abierto, Linux es un laboratorio ideal para probar ideas de sistemas operativos al mínimo coste; se puede implementar, probar, tomar medidas y, si funciona, añadir la idea al *kernel*.

Los proyectos se sucedieron y –en el inicio de Linus con el *kernel*– a la gente de la FSF, con el software de utilidad GNU y, sobre todo, con su compilador de C (*GCC*), se les unieron otros proyectos importantes como las XFree/Xorg (una versión PC de las X Window), y proyectos de escritorio como KDE y Gnome. Y el desarrollo de Internet con proyectos como el servidor web Apache, el navegador Mozilla Firefox, o las bases de datos MySQL y PostgreSQL, acabaron

por dar al *kernel* inicial Linux el recubrimiento de aplicaciones suficiente para construir los sistemas GNU/Linux y competir en igualdad de condiciones con los sistemas propietarios. Y convertir a los sistemas GNU/Linux en el paradigma del software de fuente abierta (Open Source).

Los sistemas GNU/Linux se han convertido en la punta de lanza de la comunidad Open Source, por la cantidad de proyectos que se han podido aglutinar y llevar a buen término.

El nacimiento de nuevas empresas, que crearon distribuciones GNU/Linux (empaquetamientos de *kernel* + aplicaciones) y le dieron apoyo, como Red Hat, Mandrake, SuSe, contribuyó a introducir GNU/Linux en las empresas reacias, y a comenzar el imparable crecimiento que vivimos actualmente.

Comentaremos también la **discusión sobre la denominación de los sistemas como GNU/Linux**. El término Linux para identificar el sistema operativo con que se trabaja es de común uso (para simplificar el nombre), aunque en opinión de algunos desmerece el trabajo de la FSF con el proyecto GNU, el cual ha proporcionado las principales herramientas del sistema. Aun así, el término Linux, para referirse al sistema operativo completo, es ampliamente usado comercialmente.

Nota

Podéis leer un artículo de Richard Stallman sobre GNU y Linux en http://www.gnu.org/gnu/linux-and-gnu.html

En general, para seguir una denominación más acorde a la participación de la comunidad, se utiliza el término *Linux*, cuando nos estamos refiriendo sólo al núcleo (*kernel*) del sistema operativo. Esto crea cierta confusión, ya que hay gente que habla de "sistemas Linux" o del "sistema operativo Linux" para abreviar. Cuando se trabaja con un sistema operativo GNU/Linux, se está trabajando sobre una serie de software de utilidades, en gran parte fruto del proyecto GNU, sobre el núcleo Linux. Por lo tanto, el sistema es básicamente GNU con un núcleo Linux.

El proyecto GNU de la FSF tenía por objetivo crear un sistema operativo de software libre al estilo UNIX denominado GNU [Sta02].

Linus Torvalds consiguió, en 1991, juntar su *kernel* Linux con las utilidades GNU, cuando la FSF todavía no disponía de *kernel*. El *kernel* de GNU se denomina Hurd, y hoy en día se trabaja bastante en él, y ya existen algunas versiones beta de distribuciones de GNU/Hurd.

Se calcula que, en una distribución GNU/Linux, hay un 28% de código GNU y un 3% que corresponde al código del *kernel* Linux; el porcentaje restante corresponde a código de terceros, ya sea de aplicaciones o de utilidades.

Para destacar la contribución de GNU [FSF], podemos ver algunas de sus aportaciones incluidas en los sistemas GNU/Linux:

- El compilador de C y C++ (GCC)
- El *shell bash*
- El editor Emacs (GNU Emacs)
- El intérprete *postscript* (GNU *ghostscript*)
- La biblioteca C estándar (GNU C *library*, o también *glibc*)
- El depurador (GNU *gdb*)
- *Makefile* (GNU *make*)
- El ensamblador (GNU *assembler* o *gas*)
- El *linker* (GNU *linker* o *gld*)

Los sistemas GNU/Linux no son los únicos que utilizan software GNU. Los sistemas BSD, por ejemplo, incorporan también utilidades GNU. Y algunos operativos propietarios, como MacOS X (de Apple), también usan software GNU. El proyecto GNU ha producido software de alta calidad, que se ha ido incorporando a la mayor parte de las distribuciones de sistemas basadas en UNIX, tanto libres como propietarias.

Es justo para todo el mundo reconocer el trabajo de cada uno denominando GNU/Linux a los sistemas que trataremos.

4. El perfil del administrador de sistemas

Las grandes empresas y organizaciones dependen cada vez más de sus recursos de computación y de cómo éstos son administrados para adecuarlos a las tareas. El gran incremento de las redes distribuidas, con sus equipos servidores y clientes, ha creado una gran demanda de un nuevo perfil laboral: el llamado administrador de sistemas.

El administrador de sistemas tiene una amplia variedad de tareas importantes. Los mejores administradores de sistema suelen ser bastante generalistas, tanto teórica como prácticamente. Pueden enfrentarse a tareas como realizar cableados de instalaciones o reparar cables; instalar sistemas operativos o software de aplicaciones; corregir problemas y errores en los sistemas, tanto hardware como software; formar a los usuarios, ofrecer trucos o técnicas para mejorar la productividad en áreas que pueden ir desde aplicaciones de procesamiento de textos hasta áreas complejas de sistemas CAD o simuladores; evaluar económicamente compras de equipamiento de hardware y software; automatizar un gran número de tareas comunes, e incrementar el rendimiento general del trabajo en su organización.

Puede considerarse al administrador como un perfil de empleado que ayuda a los demás empleados de la organización a aprovechar mejor y más óptimamente los recursos disponibles, de forma que mejore toda la organización.

La relación con los usuarios finales de la organización puede establecerse de diferentes maneras: mediante la formación de usuarios o con ayuda directa en el caso de presentarse problemas (incidencias). El administrador es la persona encargada de que las tecnologías utilizadas por los usuarios funcionen adecuadamente, o en otras palabras, que los sistemas cumplan las expectativas de los usuarios, así como las tareas que éstos quieran realizar.

Hace años, y aún actualmente, en muchas empresas u organizaciones no hay una perspectiva clara del papel del administrador de sistemas. En los inicios de la informática en la empresa (años ochenta y noventa), el administrador era visto en un principio como la persona "entendida" en ordenadores (el "gurú") que se encargaba de las instalaciones de las máquinas y que vigilaba o las reparaba en caso de problemas. Era una especie de informático polivalente que tenía que solucionar los problemas que fueran apareciendo. Su perfil de currículum no era claro, ya que no necesitaba tener amplios conocimientos sino sólo conocimientos básicos de una decena (como mucho) de aplicaciones (el procesador de texto, la hoja de cálculo, la base de datos, etc.), y algunos

conocimientos básicos de hardware eran suficientes para las tareas diarias. Así, cualquier simple "entendido" en el tema podía dedicarse a este trabajo, de manera que no solían ser informáticos tradicionales, y muchas veces incluso se llegaba a una transmisión oral de los conocimientos entre algún "administrador" más antiguo en la empresa y el nuevo aprendiz.

En aquella situación, nos encontrábamos de alguna manera en la prehistoria de la administración de sistemas (aunque hay personas que siguen pensando que básicamente se trata del mismo trabajo).

Hoy en día, en la época de Internet y de los servicios distribuidos, un administrador de sistemas es un profesional (con dedicación propia y exclusiva) que proporciona servicios en la "arena" del software y hardware de sistemas. El administrador tiene que llevar a cabo varias tareas que tendrán como destino múltiples sistemas informáticos, la mayoría heterogéneos, con objeto de hacerlos operativos para una serie de tareas.

Así las cosas, los administradores necesitan tener unos conocimientos generales (teóricos y prácticos) de áreas muy diferentes, desde tecnologías de redes, sistemas operativos, aplicaciones de ámbitos distintos, programación básica en una amplia variedad de lenguajes de programación, conocimientos amplios de hardware –tanto del ordenador como de los periféricos usados– y tecnologías Internet, diseño de páginas web, bases de datos, etc. Y normalmente también es buscado con el perfil de conocimientos básicos sobre el área de trabajo de la empresa, ya sea química, física, matemáticas, etc. No es de extrañar, entonces, que en una empresa de tamaño medio a grande se haya pasado del "chapuzas" de turno a un pequeño grupo de profesionales con amplios conocimientos, la mayoría con nivel académico universitario, con diferentes tareas asignadas dentro de la organización.

> El administrador debe dominar un rango amplio de tecnologías para poder adaptarse a una multitud de tareas variadas, que pueden surgir dentro de la organización.

Debido a la gran cantidad de conocimientos que debe tener, no es extraño que aparezcan a su vez diferentes subperfiles de la tarea del administrador. En una gran organización puede ser habitual encontrar a los administradores de sistemas operativos (UNIX, MacOS o Windows), que suelen ser diferentes a los orientados a administrador de bases de datos, administrador de copias de seguridad, administradores de seguridad informática, administradores encargados de atención a los usuarios, etc.

En una organización más pequeña, varias o todas las tareas pueden estar asignadas a uno o pocos administradores. Los administradores de sistemas UNIX (o de GNU/Linux) serían una parte de estos administradores (cuando no el

administrador que tendrá que hacer todas las tareas). Su plataforma de trabajo es UNIX (o GNU/Linux en nuestro caso), y requiere de bastantes elementos específicos que hacen este trabajo único. UNIX (y variantes) es un sistema operativo abierto y muy potente, y, como cualquier sistema software, exige cierto nivel de adecuación, configuración y mantenimiento en las tareas para las que vaya a ser usado. Configurar y mantener un sistema operativo es una tarea amplia y seria, y en el caso de UNIX y nuestros GNU/Linux puede llegar a ser bastante frustrante.

Algunas áreas importantes por tratar son:

a) Que el sistema sea muy potente también indica que habrá bastantes posibilidades de adaptarlo (configurarlo) a las tareas que queremos hacer. Habrá que evaluar las posibilidades que se nos ofrecen y cuán adecuadas son para nuestro objetivo final.

b) Un sistema abierto y ejemplo claro de ello es nuestro GNU/Linux, que nos ofrecerá actualizaciones permanentes, tanto en la corrección de errores del sistema como en la incorporación de nuevas prestaciones. Y, evidentemente, todo esto tiene un impacto directo en costes de mantenimiento de las tareas de administración.

c) Los sistemas se pueden utilizar para tareas de coste crítico, o en puntos críticos de la organización, donde no se pueden permitir fallos importantes, o que ralenticen o paren la marcha de la organización.

d) Las redes son actualmente un punto muy importante (si no el que más), pero también es un área de problemas potenciales muy crítica, tanto por su propia naturaleza distribuida como por la complejidad del sistema para encontrar, depurar y solucionar los problemas que se puedan presentar.

e) En el caso particular de los sistemas UNIX, y en nuestros GNU/Linux, la abundancia, tanto de versiones como de distribuciones diferentes del sistema, incorpora problemas adicionales a la administración, ya que es necesario conocer las problemáticas y diferencias de cada versión y distribución.

En particular, las tareas de administración del sistema y de la red suelen presentar particularidades diferentes, y a veces se tratan por separado (o por administradores diferentes). Aunque también pueden verse como dos caras del mismo trabajo, con el sistema propiamente dicho (máquina y software) por un lado, y el ambiente donde el sistema (el entorno de red) convive, por el otro.

Por administración de la red se entiende la gestión del sistema como parte de la red, y hace referencia a los servicios o dispositivos cercanos necesarios para que la máquina funcione en un entorno de red; no cubre dispositivos de red como los *switchs*, *bridges* o *hubs* u otros dispositivos de red, pero unos conocimientos básicos son imprescindibles para facilitar las tareas de administración.

En estos materiales y los siguientes, correspondientes a la asignatura avanzada de administración, cubriremos primero aquellos aspectos locales del propio sistema y básicos de red, y en una segunda asignatura ("Administración avanzada") veremos las tareas de administración de red y despliegue de servicios.

Ya hemos apuntado el problema de determinar qué es exactamente un administrador de sistemas, pues en el mercado laboral informático no está demasiado claro. Era común pedir administradores de sistemas según categorías (establecidas en las empresas) de programador o ingenieros de software, las cuales no se adecuan correctamente.

Un programador es, básicamente, un productor de código. En este caso, un administrador obtendría poca producción, dado que en algunas tareas puede ser necesario, pero en otras no. Normalmente, será deseable que el administrador posea más o menos conocimientos dependiendo de la categoría laboral:

a) Alguna carrera o diplomatura universitaria, preferible en informática, o en algún campo directamente relacionado con la empresa u organización.

Nota

Para ser administrador se requieren generalmente estudios informáticos o afines a la organización, junto con experiencia demostrada en el campo y conocimientos amplios de sistemas heterogéneos y tecnologías de red.

b) Suele pedirse de uno a tres años de experiencia como administrador (a no ser que el puesto sea para ayudante de uno ya existente). La experiencia también puede ampliarse de tres a cinco años.

c) Familiaridad o conocimientos amplios de entornos de red y servicios. Protocolos TCP/IP, servicios de ftp, telnet, ssh, http, nfs, nis, ldap, etc.

d) Conocimientos de lenguajes de *script* para prototipado de herramientas o automatización rápida de tareas (por ejemplo, *shell scripts*, Perl, tcl, Python, etc.) y experiencia en programación de un amplio rango de lenguajes (C, C++, Java, Assembler, etc.).

e) Puede pedirse experiencia en desarrollo de aplicaciones grandes en cualquiera de estos lenguajes.

f) Conocimientos amplios de mercado informático, tanto de hardware como de software, en el caso de que haya que evaluar compras de material o montar nuevos sistemas o instalaciones completas.

g) Experiencia en más de una versión de UNIX (o sistemas GNU/Linux), como Solaris, AIX, AT&T, SystemV, BSD, etc.

h) Experiencia en sistemas operativos no UNIX, sistemas complementarios que pueden encontrarse en la organización: Windows 9x/NT/2000/XP/Vista/7, MacOs, VMS, sistemas IBM, etc.

i) Sólidos conocimientos del diseño e implementación de UNIX, mecanismos de páginas, intercambio, comunicación interproceso, controladores, etc.; por ejemplo, si las tareas de administración incluyen optimización de los sistemas (*tuning*) y/o plataformas arquitecturales hardware.

j) Conocimientos y experiencia en seguridad informática: construcción de cortafuegos (*firewalls*), sistemas de autentificación, aplicaciones de cifrado (criptografía), seguridad del sistema de ficheros, herramientas de seguimiento de seguridad, etc.

k) Experiencia en bases de datos, conocimientos de SQL, etc.

l) Instalación y reparación de hardware y/o cableados de red y dispositivos.

5. Tareas del administrador

Según hemos descrito, podríamos separar las tareas de un administrador GNU/Linux (o UNIX en general) en dos partes principales: administración del sistema y administración de red. En los siguientes subapartados, mostramos de forma resumida en qué consisten en general estas tareas en los sistemas GNU/Linux (o UNIX). Trataremos la mayor parte del contenido con cierto detalle en estos módulos y en los asociados a administración avanzada. Otra parte de las tareas, por cuestiones de espacio o complejidad, la explicaremos superficialmente o no la trataremos.

Las tareas de administración engloban una serie de conocimientos y técnicas de los cuales aquí sólo podemos ver la "punta del iceberg". En la bibliografía adjunta a cada módulo aportamos referencias para ampliar todos los temas a tratar. Como veréis, hay una amplia bibliografía para casi cualquier punto en el que queráis profundizar.

5.1. Tareas de administración local del sistema

- **Arranque y apagado del sistema**: cualquier sistema basado en UNIX tiene unos sistemas de arranque y apagado ajustables, de manera que podemos configurar qué servicios ofrecemos en el arranque de la máquina y cuándo hay que pararlos, o programar el apagado del sistema para su mantenimiento.

- **Gestión de usuarios y grupos**: dar cabida a los usuarios es una de las principales tareas de cualquier administrador. Habrá que decidir qué usuarios podrán acceder al sistema, de qué forma y bajo qué permisos, y establecer comunidades mediante los grupos. Un caso particular será el de los usuarios de sistema, pseudousuarios dedicados a tareas del sistema.

- **Gestión de recursos del sistema**: qué ofrecemos, cómo lo ofrecemos y a quién damos acceso.

- **Gestión de los sistemas de ficheros**: el ordenador puede disponer de diferentes recursos de almacenamiento de datos y dispositivos (disquetes, discos duros, ópticos, etc.) con diferentes sistemas de acceso a los ficheros. Pueden ser permanentes, extraíbles o temporales, con lo cual habrá que modelar y gestionar los procesos de montaje y desmontaje de los sistemas de ficheros que ofrezcan los discos o dispositivos afines.

- **Cuotas del sistema**: cualquier recurso que vaya a ser compartido tiene que ser administrado, y según la cantidad de usuarios, habrá que establecer un sistema de cuotas para evitar el abuso de los recursos por parte de los usua-

rios o establecer clases (o grupos) de usuarios diferenciados por mayor o menor uso de recursos. Suelen ser habituales sistemas de cuotas de espacio de disco, o de impresión, o de uso de CPU (tiempo de computación usado).

- **Seguridad del sistema**: seguridad local, sobre protecciones a los recursos frente a usos indebidos, accesos no permitidos a datos del sistema, o a datos de otros usuarios o grupos.

- ***Backup* y restauración del sistema**: es necesario establecer políticas periódicas (según importancia de los datos), de copias de seguridad de los sistemas. Hay que establecer periodos de copia que permitan salvaguardar nuestros datos de fallos del sistema (o factores externos) que puedan provocar pérdidas o corrupción de datos.

- **Automatización de tareas rutinarias**: muchas de las tareas frecuentes de la administración o del uso habitual de la máquina pueden ser fácilmente automatizadas, ya debido a su simplicidad (y por lo tanto, a la facilidad de repetirlas), como a su temporalización, que hace que tengan que ser repetidas en periodos concretos. Estas automatizaciones suelen hacerse bien mediante programación por lenguajes interpretados de tipo *script* (*shells*, Perl, etc.), o por la inclusión en sistemas de temporalización (*cron*, *at*...).

- **Gestión de impresión y colas**: los sistemas UNIX pueden utilizarse como sistemas de impresión para controlar una o más impresoras conectadas al sistema, así como para gestionar las colas de trabajo que los usuarios o aplicaciones puedan enviar a las mismas.

- **Gestión de módems y terminales**: estos dispositivos suelen ser habituales en entornos no conectados a red local ni a banda ancha:
  - Los módems permiten una conexión a la Red por medio de un intermediario (el ISP o proveedor de acceso), o bien la posibilidad de conectar a nuestro sistema desde el exterior por acceso telefónico desde cualquier punto de la red telefónica.

  - En el caso de los terminales, antes de la introducción de las redes solía ser habitual que la máquina UNIX fuese el elemento central de cómputo, con una serie de terminales "tontos", que únicamente se dedicaban a visualizar la información o a permitir la entrada de información por medio de teclados externos. Solía tratarse de terminales de tipo serie o paralelo. Hoy en día, todavía suelen ser habituales en entornos industriales, y en nuestro sistema GNU/Linux de escritorio tenemos un tipo particular, que son los terminales de texto "virtuales", a los que se accede mediante las teclas Alt+Fxx.

- ***Accounting* (o *log*) de sistema**: para poder verificar el funcionamiento correcto de nuestro sistema, es necesario llevar políticas de *log* que nos puedan informar de los posibles fallos del sistema o del rendimiento que se

obtiene de una aplicación, servicio o recurso hardware. O bien permitir resumir los recursos gastados, los usos realizados o la productividad del sistema en forma de informe.

- ***System performance tunning***: técnicas de optimización del sistema para un fin dado. Suele ser habitual que un sistema esté pensado para una tarea concreta y que podamos verificar su funcionamiento adecuado (por ejemplo, mediante *logs*), para examinar sus parámetros y adecuarlos a las prestaciones que se esperan.

- **Personalización del sistema**: reconfiguración del *kernel*. Los *kernels*, por ejemplo en GNU/Linux, son altamente personalizables, según las características que queramos incluir y el tipo de dispositivos que tengamos o esperemos tener en nuestra máquina, así como los parámetros que afecten al rendimiento del sistema o que consigan las aplicaciones.

5.2. Tareas de administración de red

- **Interfaz de red y conectividad**: el tipo de interfaz de red que utilizamos, ya sea el acceso a una red local, la conexión a una red mayor, o conexiones del tipo banda ancha con tecnologías ADSL, RDSI, u ópticas por cable. Además, el tipo de conectividades que vamos a tener en forma de servicios o peticiones.

- ***Routing* de datos**: los datos que circularán, de dónde o hacia dónde se dirigirán, dependiendo de los dispositivos de red disponibles y de las funciones de la máquina en red; posiblemente, será necesario redirigir el trafico desde/hacia uno o más sitios.

- **Seguridad de red**: una red, sobre todo si es abierta a cualquier punto exterior, es una posible fuente de ataques y, por lo tanto, puede comprometer la seguridad de nuestros sistemas o los datos de nuestros usuarios. Hay que protegerse, detectar e impedir posibles ataques con una política de seguridad clara y eficaz.

- **Servicios de nombres**: en una red hay infinidad de recursos disponibles. Los servicios de nombres nos permiten nombrar objetos (como máquinas y servicios) para poderlos localizar y gestionar. Con servicios como el DNS, DHCP, LDAP, etc., se nos permitirá localizar servicios o equipos.

- **NIS (*network information service*)**: las grandes organizaciones han de tener mecanismos para poder organizar, de forma efectiva, los recursos y el acceso a ellos. Las formas habituales en UNIX estándar, como los *logins* de usuarios con control por *passwords* locales, son efectivos con pocas máquinas y usuarios, pero cuando tenemos grandes organizaciones, con estructuras jerárquicas, usuarios que pueden acceder a múltiples recursos de for-

ma unificada o separada por diferentes permisos, etc., los métodos UNIX sencillos se muestran claramente insuficientes o imposibles. Entonces se necesitan sistemas más eficaces para controlar toda esta estructura. Servicios como NIS, NIS+ y LDAP nos permiten organizar de modo adecuado toda esta complejidad.

- **NFS (*network fylesystems*)**: a menudo, en las estructuras de sistemas en red es necesario compartir informaciones (como los propios ficheros) por parte de todos o algunos de los usuarios. O sencillamente, debido a la distribución física de los usuarios, es necesario un acceso a los ficheros desde cualquier punto de la red. Los sistemas de ficheros por red (como NFS) permiten un acceso transparente a los ficheros, independientemente de nuestra situación en la red. Y en algunos casos, como Samba/CIFS, nos ofrecen soporte para el acceso por parte de plataformas hardware/software diferentes, e independientes de las configuraciones de clientes o servidores.

- **UNIX *remote commands***: UNIX dispone de comandos transparentes a la red, en el sentido de que, independientemente de la conexión física, es posible ejecutar comandos que muevan información por la red o permitan acceso a algunos servicios de las máquinas. Los comandos suelen tener una "*r*" delante, con el sentido de 'remoto', por ejemplo: *rcp*, *rlogin*, *rsh*, *rexec*, etc., que permiten las funcionalidades indicadas de forma remota en la red.

- **Aplicaciones de red**: aplicaciones de conexión a servicios de red, como telnet (acceso interactivo), ftp (transmisión de ficheros), en forma de aplicación cliente que se conecta a un servicio servido desde otra máquina. O bien que nosotros mismos podemos servir con el servidor adecuado: servidor de telnet, servidor ftp, servidor web, etc.

- **Impresión remota**: acceso a servidores de impresión remotos, ya sea directamente a impresoras remotas o bien a otras máquinas que ofrecen sus impresoras locales. Impresión en red de forma transparente al usuario o aplicación.

- **Correo electrónico**: uno de los primeros servicios proporcionados por las máquinas UNIX es el servidor de correo, que permite el almacenamiento de correo, o un punto de retransmisión de correo hacia otros servidores, si no iba dirigido a usuarios propios de su sistema. Para el caso web, también de forma parecida, un sistema UNIX con el servidor web adecuado ofrece una plataforma excelente para web. UNIX tiene la mayor cuota de mercado en cuanto a servidores de correo y web, y es uno de los principales mercados, donde tiene una posición dominante. Los sistemas GNU/Linux ofrecen soluciones de código abierto para correo y web y conforman uno de sus principales usos.

- **X Window**: un caso particular de interconexión es el sistema gráfico de los sistemas GNU/Linux (y la mayor parte de UNIX), X Window. Este sistema permite una transparencia total de red y funciona bajo modelos cliente servidor; permite que el procesamiento de una aplicación esté desligado de la visualización y de la interacción por medio de dispositivos de entrada, por lo que éstos se sitúan en cualquier parte de la red. Por ejemplo, podemos estar ejecutando una determinada aplicación en una máquina UNIX cuando desde otra visualizamos en pantalla los resultados gráficos, y entramos datos con el teclado y ratón locales de forma remota. Es más, el cliente, llamado cliente X, es tan sólo un componente software que puede ser portado a otros sistemas operativos, permitiendo ejecutar aplicaciones en una máquina UNIX y visualizarlas en cualquier otro sistema. Un caso particular son los llamados terminales X, que son básicamente una especie de terminales "tontos" gráficos que sólo permiten visualizar o interactuar (por teclado y ratón) con una aplicación en ejecución remota.

6. Distribuciones de GNU/Linux

Al hablar de los orígenes de los sistemas GNU/Linux, hemos comprobado que no había un único sistema claramente definido. Por una parte, hay tres elementos software principales que componen un sistema GNU/Linux:

1) El *kernel* Linux: como vimos, el *kernel* es tan sólo la pieza central del sistema. Pero sin las aplicaciones de utilidad, *shells*, compiladores, editores, etc. no podríamos tener un sistema completo.

2) Las aplicaciones GNU: en el desarrollo de Linux, éste se vio complementado con el software de la FSF existente del proyecto GNU, que le aportó editores (como *emacs*), compilador (*gcc*) y diferentes utilidades.

3) Software de terceros: normalmente de tipo de código abierto en su mayor parte. Todo sistema GNU/Linux se integra además con software de terceros que permite añadir una serie de aplicaciones de amplio uso, ya sea el propio sistema gráfico de X Windows, servidores como el Apache para web, navegadores, ofimática, etc. Asimismo, puede ser habitual incluir algún software propietario (para ámbitos no cubiertos por el software libre), dependiendo del carácter libre que en mayor o menor grado quieran disponer los creadores de la distribución.

Al ser la mayoría del software de tipo de código abierto o libre, ya sea el *kernel*, software GNU o de terceros, hay una evolución más o menos rápida de versiones, ya sea por medio de corrección de errores o nuevas prestaciones introducidas. Esto obliga a que, en el caso de querer crear un sistema GNU/Linux, tengamos que escoger qué software queremos instalar en el sistema y qué versiones concretas de este software.

> El mundo GNU/Linux no se limita a una empresa o comunidad particular, con lo que ofrece a cada uno la posibilidad de crear su propio sistema adaptado a sus necesidades.

Entre el conjunto de las versiones de los diferentes componentes, siempre se encuentran algunas que son estables y otras que están en desarrollo, en fases alfa o beta (posiblemente, con errores o funcionalidades no completas u optimizadas), por lo que habrá que tener cuidado con la elección de las versiones a la hora de crear un sistema GNU/Linux. Otro problema añadido es la selección de alternativas; el mundo de GNU/Linux es lo suficientemente rico para que

haya más de una alternativa para un mismo producto de software. Hay que elegir entre las alternativas posibles, incorporar algunas o todas, si queremos ofrecer al usuario libertad para escoger su software.

Un caso práctico son los gestores de escritorio de X Window en los que, por ejemplo, nos ofrecen (principalmente) dos entornos de escritorio diferentes como Gnome y KDE. Los dos tienen características parecidas y aplicaciones semejantes o complementarias.

En el caso de un distribuidor de sistemas GNU/Linux, ya sea comercial o bien una organización/comunidad sin beneficio propio, dicho distribuidor tiene como responsabilidad generar un sistema que funcione, seleccionando las mejores versiones y productos software que puedan conseguirse en el momento.

En este caso, una distribución GNU/Linux es una colección de software que forma un sistema operativo basado en el *kernel* Linux.

Un dato importante a tener en cuenta, y que provoca más de una confusión, es que, como cada uno de los paquetes de software de la distribución tendrá su propia versión (independiente de la distribución en la que esté ubicado), el número de distribución asignado no mantiene una relación con las versiones de los paquetes software.

La única función del número de distribución es comparar las distribuciones que genera un mismo distribuidor. No permite comparar entre otras distribuciones. Si queremos hacer comparaciones entre distribuciones, tendremos que examinar los paquetes software principales y sus versiones para poder determinar qué distribución aporta más novedades.

Ejemplo

Pongamos un ejemplo de algunas versiones que hemos encontrado en diferentes distribuciones GNU/Linux (las versiones dependerán de la situación actual de las distribuciones):

a) *Kernel* Linux: actualmente, podemos encontrar distribuciones que ofrecen uno o más *kernels*, como los de la serie antigua 2.4.x (ya obsoleta, pero puede encontrarse en algunas máquinas en producción) o generalmente los últimos 2.6.x en revisiones (el número x) de distinta actualidad.

b) La opción en el sistema gráfico X Window: en versión de código abierto, que podemos encontrar prácticamente en todos los sistemas GNU/Linux, ya sean algunas versiones residuales de Xfree86 como las que manejan versiones 4.x.y o bien el proyecto Xorg (siendo un *fork* del anterior en el 2003, generado por problemas de licencias en XFree), que goza de más popularidad en diferentes versiones 6.x o 7.x (en algunas distribuciones se numera el paquete del servidor X con 1.7.x). Actualmente, la mayoría de distribuciones han migrado desde XFree a Xorg, por distintos problemas existentes con cambios de licencias en XFree, siendo Xorg la implementación oficial de X Window en GNU/Linux. Lo mismo pasa en diversas variantes de BSD, UNIX propietarios como Solaris, y también en las últimas versiones de MacOS X.

c) Gestor de ventanas o escritorio: podemos disponer de Gnome o KDE, o ambos; Gnome con versiones 2.x/3.x o KDE 3.x.y. / 4.x.y.

Pudimos obtener en un momento determinado, por ejemplo, una distribución que incluyese *kernel* 2.4, con XFree 4.4 y Gnome 2.14; o bien otra, por ejemplo, *kernel* 2.6, Xorg 6.8, KDE 3.1. ¿Cuál es mejor? Es difícil compararlas, ya que suponen una mezcla de elementos, y dependiendo de cómo se haga la mezcla, el producto saldrá mejor o peor, y más o menos adaptado a las necesidades del usuario. Usualmente, el distribuidor mantiene un compromiso entre la estabilidad del sistema y la novedad de las versiones incluidas. También proporciona software de aplicación atrayente para los usuarios de la distribución, ya sea aquél generalista o especializado en algún campo concreto.

En general, podemos hacer un mejor análisis de distribuciones a partir de los siguientes apartados, que habría que comprobar en cada una de ellas:

a) Versión del núcleo Linux: la versión viene indicada por unos números *X*.*Y*.*Z*, donde *X* es la versión principal, que representa los cambios importantes del núcleo; *Y* es la versión secundaria, e implica mejoras en las prestaciones del núcleo: *Y* es par en los núcleos estables e impar en los desarrollos o pruebas; y *Z* es la versión de construcción, que indica el número de la revisión de *X*.*Y*, en cuanto a parches o correcciones hechas. Los distribuidores no suelen incluir la última versión del núcleo, sino la que ellos hayan probado con más frecuencia y pueden verificar que es estable para el software, y componentes que ellos incluyen. Este fue el esquema de numeración clásico (que se siguió durante las ramas 2.4.x, hasta los inicios de la 2.6), que tuvo algunas modificaciones, para adaptarse al hecho de que el *kernel* (rama 2.6.x) se volvió mas estable, y cada vez las revisiones son menores (para significar un salto de versión de los primeros números), debido al desarrollo continuo y frenético. En los últimos esquemas de numeración del *kernel*, se llegan a introducir cuartos números, para especificar de *Z* cambios menores, o diferentes posibilidades de la revisión (con diferentes parches añadidos). La versión así definida con cuatro números es la que se considera estable (*stable)*. También son usados otros esquemas para las versiones de test (no recomendables para entornos de producción), como sufijos *-rc* (*release candidate*), los *-mm* que son *kernels* experimentales con pruebas de diferentes técnicas, o los *-git* que son una especie de "foto" diaria del desarrollo del *kernel*. Estos esquemas de numeración están en constante cambio para adaptarse a la forma de trabajar de la comunidad del *kernel*, y a sus necesidades para acelerar el desarrollo del *kernel*.

b) Formato de empaquetado: es el mecanismo empleado para instalar y administrar el software de la distribución. Se conoce por el formato de los paquetes de software soportados. En este caso suelen estar los formatos *rpm*, *deb, tar.gz, mdk*, etc. Aunque cada distribución suele tener posibilidad de utilizar varios formatos, tiene uno por defecto. El software acostumbra a venir con sus archivos en un paquete que incluye información sobre su instalación y posibles dependencias con otros paquetes de software. El empaquetado es importante si se usa software de terceros que no venga con la distribución, ya que el software puede encontrarse sólo en algunos sistemas de paquetes, o incluso en uno sólo, y en algunos casos está pensado para versiones concretas de algunas distribuciones de GNU/Linux.

c) Estructura del sistema de archivos: la estructura del sistema de archivos principal (*/*) nos indica dónde podemos encontrar nuestros archivos (o los propios del sistema) dentro del sistema de ficheros. En GNU/Linux y UNIX hay algunos estándares de colocación de los archivos (como veremos en otros módulos), como por ejemplo el FHS (*filesystem hierarchy standard*) [Lin03b]. Así, si tenemos una idea del estándar, sabremos dónde encontrar la mayor parte de los archivos. Luego depende de que la distribución lo siga más o menos y de que nos avisen de los cambios que hayan realizado del estándar.

d) *Scripts* de arranque del sistema: los sistemas UNIX y GNU/Linux incorporan unos guiones de arranque (o *shell scripts*) que indican cómo debe arrancar la máquina y cuál será el proceso (o fases) que se van a seguir, así como lo que deberá hacerse en cada paso. Para este arranque hay dos modelos principales, los de SysV o BSD (es una de las diferencias de las dos ramas de UNIX principales), y cada distribución podría escoger uno o otro. Aunque los dos sistemas tienen la misma funcionalidad, son diferentes en los detalles, y esto será importante en los temas de administración (lo veremos en la administración local). En nuestro caso, los sistemas que analizaremos, tanto Fedora como Debian, utilizan el sistema de SysV (será el que veremos en la unidad local), pero existen otras distribuciones como Slackware que utilizan el otro sistema propio de BSD. Y actualmente existen determinadas propuestas para nuevas opciones en este aspecto de arranque, como por ejemplo la que ha obtenido peso importante en las distribuciones actuales, como el sistema *upstart* utilizado, entre otras, en la distribución Ubuntu.

e) Versiones de la biblioteca del sistema: todos los programas (o aplicaciones) que tenemos en el sistema dependen para su ejecución de un número (mayor o menor) de bibliotecas de sistema. Estas bibliotecas, normalmente de dos tipos, ya sean estáticas unidas al programa (archivos *libxxx.a*) o dinámicas que se cargan en tiempo de ejecución (archivos *libxxx.so*), proporcionan gran cantidad de código de utilidad o de sistema que utilizarán las aplicaciones. La ejecución de una aplicación puede depender de la existencia de unas bibliotecas adecuadas y del número de versión concreto de estas bibliotecas (no es lo recomendable, pero puede suceder). Un caso bastante habitual es la biblioteca GNU C *library*, la biblioteca estándar de C, también conocida como *glibc*. Puede suceder que una aplicación nos pida que dispongamos de una versión concreta de la *glibc* para poder ejecutarse o compilarse. Es un caso bastante problemático, y por ello uno de los parámetros que se valoran de la distribución es conocer qué versión de la *glibc dispone*, y los posibles paquetes adicionales de versiones de compatibilidad con versiones antiguas. El problema aparece al intentar ejecutar o compilar un producto de software muy antiguo en una distribución moderna, o bien un producto de software muy nuevo en una distribución antigua.

El mayor cambio llegó al pasar a una *glibc* 2.0, en que había que recompilar todos los programas para poder ejecutarlos correctamente. En las diferentes revisiones nuevas de numeración 2.x ha habido algunos cambios menores que

podían afectar a alguna aplicación. En muchos casos, los paquetes de software comprueban si se tiene la versión correcta de la *glibc*, o en el mismo nombre mencionan la versión que hay que utilizar (ejemplo: *paquete-xxx-glibc2.rpm*).

f) Escritorio X Window: el sistema X Window es el estándar gráfico para GNU/Linux como visualización de escritorio. Fue desarrollado en el MIT en 1984 y prácticamente todos los UNIX tienen una versión del mismo. Las distribuciones GNU/Linux disponen de diferentes versiones como la Xfree86 o la Xorg (como ya dijimos, esta última es el estándar de facto en estos momentos). El X Window es una capa gráfica intermedia que confía a otra capa denominada gestor de ventanas la visualización de sus elementos. Además, podemos combinar el gestor de ventanas con utilidades y programas de aplicación variados para formar lo que se denomina un entorno de escritorio.

Linux tiene, principalmente, dos entornos de escritorio: Gnome y KDE. Cada uno tiene la particularidad de basarse en una biblioteca de componentes propios (los diferentes elementos del entorno como ventanas, botones, listas, etc.): *gtk+* (en Gnome) y *Qt* (en KDE), que son las principales bibliotecas de componentes que se usan para programar aplicaciones en estos entornos. Pero además de estos entornos, hay muchos otros, gestores de ventanas o escritorios: XCFE, Fluxbox, Motif, Enlightement, BlackIce, FVWM, etc., de modo que la posibilidad de elección es amplia. Además, cada uno de ellos permite cambiar la apariencia (*look & feel*) de ventanas y componentes al gusto del usuario, o incluso crearse el suyo propio.

g) Software de usuario: software añadido por el distribuidor, en su mayoría de tipo Open Source, para las tareas más habituales (o por el contrario, no tanto como para algunos campos de aplicación muy especializados).

Las distribuciones habituales son tan grandes que pueden encontrarse de centenares a miles de estas aplicaciones (muchas de las distribuciones tienen desde unos pocos CD a varias decenas de ellos (o unas pocas unidades de DVD), de aplicaciones extra de partida. O pueden obtenerse a posteriori por red de los repositorios oficiales (o extras) de software de las distribuciones. Estas aplicaciones cubren casi todos los campos, desde el hogar hasta administrativos o científicos. Y en algunas distribuciones se añade software propietario de terceros (como, por ejemplo, alguna *suite* ofimática del tipo Office), software de servidor preparado por el distribuidor, como por ejemplo un servidor de correo, un servidor web seguro, etc.

Así es cómo cada distribuidor suele producir **diferentes versiones** de su distribución; por ejemplo, a veces hay distinciones entre una versión personal, profesional o de tipo servidor. Suele ser más habitual una distinción entre dos versiones: Desktop (escritorio) y Server (servidor). Aunque en muchas distri-

buciones suele incluirse el software complementando ambos aspectos, y es la instalación final que haga el usuario, de qué paquetes software y su configuración, la que decide el ámbito final de la máquina y su sistema GNU/Linux.

El sistema GNU/Linux de fondo es el mismo; sólo hay diferencias (que se pagan en algunos casos) en el software añadido (en general, obra de la misma casa distribuidora). Por ejemplo, en servidores web o en servidores correo, ya sean desarrollos propios, optimizados o mejorados. Otras diferencias pueden ser la inclusión de mejores herramientas, desarrolladas por el fabricante de la distribución, o el soporte adicional, en forma de contratos de mantenimiento que incluya el distribuidor comercial.

A menudo, asumir un coste económico extra no tiene mucho sentido, ya que el software estándar es suficiente (con un poco de trabajo extra de administración de sistemas); pero para las empresas puede ser interesante porque reduce tiempo de instalación y mantenimiento de los servidores y, además, optimiza algunas aplicaciones y servidores críticos para la gestión informática de la empresa.

6.1. Debian

El caso de la distribución Debian GNU/Linux es especial, en el sentido de que es una distribución guiada por una comunidad sin fines comerciales, aparte de mantener su distribución y promocionar el uso del software de código abierto y libre.

Nota

Los documentos "Contrato social Debian" son consultables en: http://www.debian.org/social_contract

Debian es una distribución apoyada por una comunidad entusiasta de usuarios y desarrolladores propios, basada en el compromiso de la utilización de software libre.

El proyecto Debian se fundó en 1993 para crear la distribución Debian GNU/Linux. Desde entonces, se ha vuelto bastante popular y rivaliza en uso con otras distribuciones comerciales como Red Hat o SUSE. Por ser un proyecto comunitario, el desarrollo de esta distribución se rige por una serie de normas o políticas: existen unos documentos llamados "Contrato social Debian", que mencionan la filosofía del proyecto en su conjunto, y las políticas Debian, que especifican en detalle cómo se implementa su distribución.

Figura 2. Logotipo de Debian

La distribución Debian está bastante relacionada con los objetivos de la FSF y su proyecto de Software Libre GNU; por esta razón, incluyen siempre en su nombre: "Debian GNU/Linux"; además, su texto del contrato social ha servido como base de las definiciones de código abierto. En cuanto a las políticas, todo

aquel que quiera participar en el proyecto de la distribución tiene que seguirlas. Aunque no se sea un colaborador, estas políticas pueden ser interesantes porque explican cómo es la distribución Debian.

Cabe mencionar también un aspecto práctico de cara a los usuarios finales: Debian ha sido siempre una distribución difícil (aunque esta percepción ha cambiado en las últimas versiones). Suele ser la distribución que usan los *hackers* de Linux, en el buen sentido de los que se encargan del desarrollo y test del *kernel*, aportan modificaciones, programadores de bajo nivel, los que desean estar a la última para probar software nuevo, los que quieren probar los desarrollos del *kernel* que todavía no han sido publicados... O sea, todo tipo de fauna de *hackers* de GNU/Linux.

Las versiones anteriores de Debian se habían hecho famosas por su dificultad de instalación. La verdad es que no se hacía demasiado para que fuese fácil de cara a los no expertos. Pero las cosas con el tiempo han mejorado. Ahora, la instalación, no sin ciertos conocimientos, puede hacerse guiada por un instalador gráfico (conocido como Debian Installer), mientras antes era una instalación puramente textual (de hecho todavía se mantiene, y es muy usada para la instalación en ambientes de servidor). Pero aun así, los primeros intentos pueden llegar ser un poco traumáticos por el grado de conocimiento inicial exigido para algunos aspectos de la instalación.

Debian GNU/Linux no es una única distribución, sino que suele diferenciarse en una serie de variantes, los llamados "sabores" de la distribución Debian. En este momento hay tres ramas (sabores) de la distribución: la ***stable***, la ***testing*** y la ***unstable***. Como sus nombres indican, la *stable* es la que está destinada a entornos de producción (o usuarios que desean estabilidad), la *testing* ofrece software más nuevo que ha sido comprobado mínimamente (podríamos decir que es una especie de versión beta de Debian) y que pronto van a ser incluidos en la *stable*. Y la *unstable* es la que presenta las últimas novedades de software, cuyos paquetes cambian en plazos muy cortos; en una semana, e incluso en el día a día pueden cambiar varios paquetes. Todas ellas son actualizables desde varias fuentes (CD, ftp, web) por un sistema denominado APT que maneja los paquetes software DEB de Debian. Las tres distribuciones tienen nombres más comunes asignados (por ejemplo, en un determinado momento de producción de Debian):

- Lenny (*stable*)
- Squeeze (*testing*)
- Sid (*unstable*)

La versión previa *stable* se denominaba Etch (4.0), anteriormente Sarge (era la 3.1). La más actual es la Debian GNU/Linux Lenny (5.0). Las versiones más extendidas (además de versiones antiguas que sigan en producción) son la

Lenny y la Sid, que son los dos extremos. La Sid no está recomendada para entornos (de producción) de trabajo diario, porque puede traer características a medias que aún se están probando y pueden fallar (aunque no es habitual); es la distribución que suelen usar los *hackers* de GNU/Linux. Además, esta versión cambia casi a diario. Suele ser normal que, si se quiere actualizar a diario, existan de diez a veinte paquetes de software nuevos por día (o incluso más en algunos momentos puntuales de desarrollo o cambios importantes).

La Lenny es quizás la mejor elección para el sistema de trabajo diario; se actualiza periódicamente para cubrir nuevo software o actualizaciones (como las de seguridad). No dispone del último software, y éste no se incluye hasta que la comunidad lo haya verificado en un amplio rango de pruebas.

Vamos a comentar brevemente algunas características de esta distribución; las versiones son las que se encuentran por defecto en la *stable* (Lenny) y en *unstable* (Sid) a día de hoy:

a) La distribución (*stable*) actual consta de entre 28 a 33 CD (o 4-5 DVD) de la última revisión disponible y dependiendo de la arquitectura (soporta más de 12 arquitecturas hardware diferentes). Hay diferentes posibilidades dependiendo del conjunto de software que nos encontremos en soporte físico (CD o DVD), o bien lo que deseamos posteriormente descargar desde la red, con lo cual sólo necesitamos un CD básico (*netinstall CD*), más el acceso a red, para descargar el resto según demanda. Esta distribución puede comprarse (a precios simbólicos de soporte físico, y de esta manera contribuimos a mantener la distribución) o puede descargarse desde debian.org o sus *mirrors*.

b) La *testing* y *unstable* no suelen tener CD/DVD oficiales estables, sino que puede convertirse una Debian *stable* a *testing* o *unstable* mediante cambios de configuración del sistema de paquetes APT. En algunos casos, Debian proporciona imágenes de CD/DVD que se generan semanalmente con el contenido de la distribución *testing*.

c) Núcleo Linux: utilizaban núcleos de la serie 2.6.x por defecto (algunas versiones previas de Debian incluían por defecto *kernels* de la serie 2.4.x). El enfoque de Debian en *stable* es potenciar la estabilidad y dejar a los usuarios la opción de otro producto más actualizado de software, si lo necesitan (en *unstable* o *testing*). Por ejemplo, en el momento de "congelar" la versión estable, se escoge la versión de *kernel* más estable existente, y en las revisiones posteriores se suelen corregir si se detectan problemas. Por su parte, *testing* y *unstable* suelen incluir con poca diferencia de tiempo las últimas versiones de *kernel* a medida que se producen, con el tiempo. Por ejemplo, en la rama 2.6.x suelen transcurrir varias subversiones de *kernel* (la numeración x) entre la distribución *stable* y las no estables.

d) Formato de empaquetado: Debian soporta uno de los que más prestaciones ofrece, el APT. Los paquetes de software tienen un formato denominado DEB. El APT es una herramienta de más alto nivel para gestionarlos y mantener una base de datos de los instalables y los disponibles en el momento. Además, el sistema APT puede obtener software de varias fuentes, ya sea desde CD, ftp, web.

e) El sistema con APT es actualizable en cualquier momento, mediante lista de repositorios de fuentes de software Debian (fuentes APT), que pueden ser los sitios Debian por defecto (debian.org) o de terceros. No estamos así ligados a una empresa única ni a ningún sistema de pago por suscripción.

f) Algunas de las versiones utilizadas, en un ejemplo de tiempo concreto son: para una *stable kernel* (2.6.26), Xorg (1.4.1), *glibc* (2.7)... Debian Sid tiene *kernel* (2.6.32), Xorg (1.7.7), *glibc* (2.10.2)...

g) En el escritorio acepta tanto Gnome (por defecto) como KDE (K Desktop Environment). *Unstable* suele disponer de las últimas versiones disponibles de estos entornos.

h) En cuanto a aplicaciones destacables, incluye la mayoría de las que solemos encontrar en las distribuciones de GNU/Linux: editores como *emacs* (y *xemacs*), compilador *gcc* y herramientas, servidor web Apache, navegador web (Firefox), software Samba para compartir archivos con Windows, etc.

i) Incluye también *suites* ofimáticas como OpenOffice y KOffice.

j) Incluye muchos ficheros de configuración personalizados para su distribución en directorios de /etc.

k) Debian usaba por defecto el **gestor de arranque** *lilo* en versiones previas, aunque en las últimas se ha movido a *Grub*.

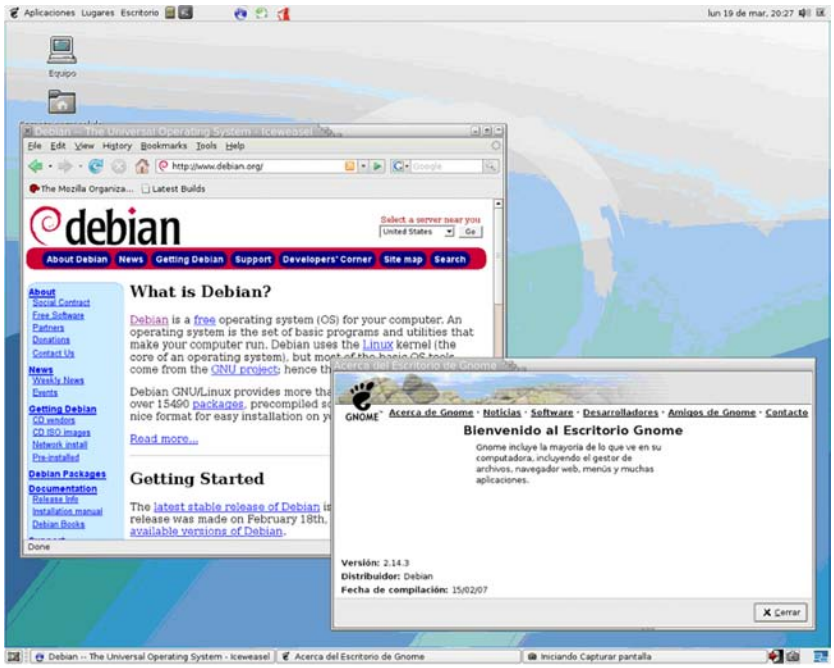
l) La configuración de la escucha de los servicios de red TCP/IP, que se realiza como en la mayoría de UNIX, con el servidor *inetd* (*/etc/inetd.conf*). Aunque dispone también *opcidonal* de *xinetd*, una opción que toma más peso.

m) Hay muchas **distribuciones GNU/Linux** más, basadas en Debian, ya que el sistema puede adaptarse fácilmente para hacer distribuciones más pequeñas o más grandes, o con más o menos software adaptado a un segmento. Una muy famosa es Knoppix (por ser una de las pioneras en el concepto LiveCD), una distribución de un único CD (o DVD), de tipo LiveCD (de ejecución directa en CD), que es muy usada para demos de GNU/Linux, o para probarla en una máquina sin hacer una instalación previa, ya que arranca y se ejecuta desde CD, aunque también puede instalarse en disco duro y convertirse en una Debian estándar. Linex es otra distribución que ha conseguido bastante fama por su desarrollo apoyado por una administración, la de la comunidad

autónoma de Extremadura. Por otra parte, encontramos a Ubuntu Linux (de la empresa Canonical), una de las distribuciones que ha obtenido más amplia repercusión (superando incluso a Debian en varios aspectos), por sus facilidades para construir una alternativa de escritorio. Ubuntu es un referente Linux en el ámbito de escritorio, y también dispone de una versión especial orientada a servidor, Ubuntu Server. Por otra parte, es interesante destacar que ha ido en aumento la colaboración en Canonical y la comunidad Debian, para reflejar los avances de cada una en las respectivas distribuciones.

Debian puede usarse como base para otras distribuciones; por ejemplo, Knoppix es una distribución basada en Debian que puede ejecutarse desde el CD sin necesidad de instalarse en disco. Linex es una distribución Debian adaptada por la administración de la comunidad de Extremadura, en su proyecto de adoptar software de código abierto. Y Ubuntu es una distribución especialmente optimizada para entornos de escritorio.

Figura 3. Entorno Debian con Gnome



6.2. Fedora

Red Hat Inc [Redh] es una de las principales firmas comerciales del mundo GNU/Linux, con una de las distribuciones con más éxito. **Bob Young** y **Marc Ewing** crearon Red Hat Inc en 1994. Estaban interesados en los modelos de software de código abierto y pensaron que sería una buena manera de hacer negocio. Su principal producto es su distribución Red Hat Linux (que abrevia-

remos como Red Hat), que está abierto a diferentes segmentos de mercado, tanto al usuario individual (versiones personal y profesional), como preferentemente hacia a las medianas o grandes empresas (con su versiones Enterprise).

Red Hat Linux es la principal distribución comercial de Linux, orientada tanto a mercado de oficina de escritorio como a servidores de gama alta. Además, Red Hat Inc es una de las empresas que más colaboran con el desarrollo de Linux, ya que varios miembros importantes de la comunidad trabajan para ella.

Figura 4. Logotipos de Red Hat Fedora

Aunque Red Hat trabaja con un modelo de código abierto, se trata de una empresa, y por lo tanto sus fines son comerciales; por ello, suele añadir a su distribución básica valores por medio de contratos de soporte, suscripciones de actualización y otros métodos. En el caso empresarial, añade software personalizado (o propio), para hacer que se adecue más el rendimiento a los fines de la empresa, ya sea por servidores optimizados o por software de utilidad propio de Red Hat.

A partir de cierto momento (finales del 2003), Red Hat Linux (versión 9.x), su versión de GNU/Linux para escritorio, se da por discontinuada, y aconseja a sus clientes a migrar a las versiones empresariales de la firma, que continuarán siendo las únicas versiones soportadas oficialmente por la firma.

En este momento Red Hat decide iniciar el proyecto abierto a la comunidad denominado Fedora, con el objetivo de realizar una distribución guiada por comunidad (al estilo Debian, aunque con fines diferentes), que denominará Fedora Core (más tarde, simplemente, Fedora). De hecho, se persigue crear un laboratorio de desarrollo abierto a la comunidad que permita probar la distribución, y a su vez guiar los desarrollos comerciales de la empresa en sus distribuciones empresariales.

Nota

Ver: http://fedoraproject.org/

En cierto modo, algunos críticos señalan que se usa a la comunidad como *betatesters* de las tecnologías que se incluirán en productos comerciales. Además, este modelo es utilizado posteriormente por otras compañías para crear modelos duales de distribuciones de comunidad a la vez que comerciales. Entonces aparecen ejemplos como OpenSuse (a partir de la comercial Novell SuSe).

El duo Red Hat y la comunidad Fedora presenta una cierta visión conservadora (menos acentuada en Fedora) de los elementos software que añade a su distribución, ya que su principal mercado de destino es el empresarial, e intenta hacer su distribución lo más estable posible, a pesar de que no cuentan con las últimas versiones. Lo que sí hace como valor añadido es depurar extensamente el *kernel* de Linux con su distribución, y genera correcciones y parches (*patchs*) para mejorar su estabilidad. A veces, pueden llegar a deshabilitar alguna funcionalidad (*drivers*) del *kernel*, si considera que éstos no son lo suficientemente

estables. También ofrece muchas utilidades en el entorno gráfico y programas gráficos propios, incluidas unas cuantas herramientas de administración. En cuanto a los entornos gráficos, utiliza tanto Gnome (por defecto) como KDE, pero mediante un entorno modificado propio mediante temas de escritorio propios, que hace que los dos escritorios sean prácticamente iguales (ventanas, menús, etc.).

La versión que utilizaremos será la última Fedora Linux disponible, que denominaremos simplemente Fedora. En general, los desarrollos y prestaciones que se mantienen suelen ser bastante parecidos en las versiones que salen a posteriori, con lo cual la mayoría de comentarios serían aplicables a las diferentes versiones a lo largo del tiempo. Tenemos que tener en cuenta que la comunidad Fedora, intenta cumplir un calendario de aproximadamente seis meses para cada nueva versión. Y se produce un cierto consenso sobre las prestaciones nuevas a introducir.

Red Hat, por contra, dejó en gran parte el mercado de versiones de escritorio en manos de la comunidad Fedora, y se centra en sus negocios en las versiones empresariales RHEL (Red Hat Enterprise Linux en varias ediciones). Cabe destacar también la existencia de versiones libres compatibles con Red Hat Empresarial, como la distribución CentOS Linux , y en otra medida con orientación al ámbito científico, como Scientific Linux .

Nota

Distribuciones compatibles RHEL:
http://www.scientificlinux.org/
http://centos.org/

Vamos a comentar brevemente algunas características de esta distribución Fedora:

a) La distribución actual consiste o bien en un DVD con el sistema completo, o un CD con una versión LiveCD instalable. También existen diferentes ediciones (denominadas *spins*) orientadas a sectores determinados (juegos, educación, científicos) o bien a escritorios concretos (Gnome, KDE, XFCE).

b) Núcleo Linux: utiliza núcleos de la serie 2.6.x, que pueden irse actualizando con el sistema de paquetes rpm (por medio de la utilidad *yum*, por ejemplo). Red Hat, por su parte, somete el *kernel* a muchas pruebas y crea parches para solucionar problemas, que normalmente también son integrados en la versión de la comunidad Linux, ya que bastantes de los colaboradores importantes en la comunidad del *kernel* de Linux trabajan para Red Hat.

c) Formato de empaquetado: Red Hat distribuye su software mediante el sistema de paquetes RPM (*red hat package manager*), los cuales se gestionan mediante el comando *rpm* o las utilidades *yum*. RPM es uno de los mejores sistemas de empaquetado existentes (al estilo del *deb* Debian), y algunos UNIX propietarios lo están incluyendo. El sistema RPM mantiene una pequeña base de datos con los paquetes instalados, y verifica que el paquete que se va instalar con el comando *rpm* no esté ya instalado, o entre en conflicto con algún otro paquete de software o por contra falte algún paquete software o versión de éste, necesaria para la instalación. El paquete RPM es un conjunto de ficheros

comprimidos junto con información de sus dependencias o del software que necesita. El sistema de gestión de paquetes RPM se ha implementado desde las versiones de Red Hat a Fedora, pero también a otras distribuciones como SUSE, y Mandriva.

d) En cuanto al arranque, utiliza *scripts* de tipo SysV (*sysvinit*). En las últimas versiones se ha reemplazado por *upstart* (proveniente de Ubuntu), aunque se mantiene su compatibilidad con los *scripts sysvinit*.

e) En el escritorio acepta tanto Gnome (escritorio por defecto) como KDE de forma opcional.

f) En cuanto a aplicaciones destacables, incluye las que solemos encontrar en la mayoría de distribuciones de GNU/Linux: editores como *emacs* (y *xemacs*), compilador *gcc* y herramientas, servidor web Apache, navegador web Firefox/Mozilla, software Samba para compartir archivos con Windows, etc.

g) Incluye también *suites* ofimáticas como OpenOffice y KOffice.

h) El software adicional puede obtenerse por los servicios de actualización *yum* (entre otros) de forma parecida al sistema APT en Debian o con diferentes herramientas de *update* incluidas, o bien por Internet mediante paquetes RPM pensados para la distribución.

i) Fedora usa el **cargador de arranque Grub** para arrancar la máquina por defecto.

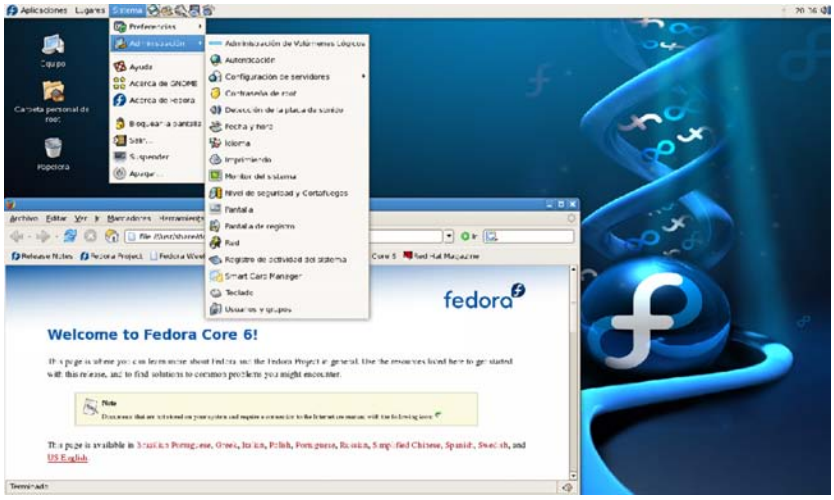
j) La configuración de escucha de los servicios de red TCP/IP, que se lleva a cabo en la mayoría de UNIX con el servidor *inetd* (*/etc/inetd.conf*), en Red Hat ha sido sustituido por *xinetd*, que tiene una configuración más modular (directorio */etc/xinetd.d*).

k) Dispuso en arranque de un programa denominado Kudzu que se encargaba de **verificar cambios de hardware y detectar el hardware nuevo instalado**. En las versiones actuales queda obsoleto, debido a la existencia de una nueva API denominada HAL que permite realizar esta función (que ha sido progresivamente integrada en las distribuciones GNU/Linux).

l) Hay varias **distribuciones más** basadas en el Red Hat original, que siguen muchas de sus características; cabe destacar Mandriva (antes Mandrake): una distribución francesa que en su origen se basó en Red Hat y que sigue en los primeros puestos junto con Red Hat en las preferencias de los usuarios (sobre todo en trabajo de escritorio). Mandriva desarrolla software propio y multitud de asistentes para ayudar a la instalación y administración de las tareas más comunes, separándose de su origen con base en Red Hat. Por otra parte, las versiones empresariales de Red Hat también han originado una serie de distribuciones libres muy populares en entornos de servidor, como CentOS [Cen]

(que intenta mantener una compatibilidad 100% con el Red Hat empresarial), y Scientific Linux (especializada en el cómputo científico en proyectos de investigación científica).

Figura 5. Un escritorio Fedora con Gnome



Con respecto a la distribución comunitaria Fedora y sus orígenes comerciales en Red Hat:

a) Es una distribución **creada por la comunidad voluntaria** de programadores y usuarios basada en desarrollo que no cuenta con soporte ni de actualizaciones ni de mantenimiento por parte del fabricante. Este aspecto pasa a depender de la comunidad, de forma semejante al caso de la distribución Debian GNU/Linux.

Nota

Ved http://docs.fedoraproject.org/

b) Las versiones se van a producir con bastante rapidez; se esperan nuevas versiones de la distribución aproximadamente cada seis meses.

c) Para la gestión de paquetes, también utiliza el sistema de paquetes RPM. Respecto al proceso de la actualización de los paquetes de la distribución o a la instalación de otros nuevos, pueden obtenerse por diferentes herramientas, con los canales de actualización de Fedora (repositorios de software), por los nuevos sistemas de actualización *yum* (basados en el sistema *rpm*).

d) Otras cuestiones más técnicas (algunas de las cuales veremos en los siguientes apartados) pueden encontrarse en las notas de la versión de cada lanzamiento de Fedora.

7. Qué veremos...

Una vez hemos examinado esta introducción "filosófica" al mundo del código abierto y la historia de los sistemas UNIX y GNU/Linux, así como definido cuáles serán las tareas de la figura del administrador del sistema, pasaremos a tratar las diferentes tareas típicas que nos encontraremos durante la administración de sistemas GNU/Linux.

A continuación, examinaremos las diferentes áreas que implica la administración de sistemas GNU/Linux. En cada módulo, examinaremos un mínimo de fundamentos teóricos que nos permitan explicar las tareas a realizar y entender el funcionamiento de las herramientas que utilizaremos. Cada módulo vendrá acompañado de algún tipo de taller, donde veremos una pequeña sesión de trabajo de una tarea o el uso de algunas herramientas. Sólo recordaremos que, como dijimos en la presentación, el tema de la administración es amplio y cualquier intento de abarcarlo completamente (como este) tiene que fallar por las dimensiones limitadas; por ello, en cada tema encontraréis abundante bibliografía (en forma libros, sitios web, howto's, etc.) en el que ampliar la "pequeña" introducción que habremos hecho del asunto.

Los temas que veremos son los siguientes:

- **En el módulo de nivel de usuario**, daremos una perspectiva de los sistemas de GNU/Linux desde la perspectiva del usuario final. Observaremos procedimientos básicos de arranque e instalación del sistema, así como configuraciones básicas de dispositivos, entorno gráfico de escritorio y gestión de software.

- **En el módulo de *shells scripts***, veremos una herramienta básica para el administrador, que es la posibilidad de automatizar tareas mediante lenguajes interpretados proporcionados por el sistema. Analizaremos su sintaxis y posibilidades básicas, así como algunas utilidades de sistema básicas que complementarán la programación de los *scripts*.

- **En el módulo de migración**, obtendremos una perspectiva del tipo de sistemas informáticos que se están utilizando y en qué ambientes de trabajo se usan. Veremos, asimismo, cómo los sistemas GNU/Linux se adaptan mejor o peor a cada uno de ellos, y plantearemos una primera disyuntiva a la hora de introducir un sistema GNU/Linux: ¿cambiamos el sistema que teníamos o lo hacemos por etapas, coexistiendo ambos?

- **En el módulo de herramientas de administración local**, estudiaremos (básicamente) aquel conjunto de útiles con el que el administrador tendrá que "vivir" (y/o sufrir) a diario, y que podrían formar la "caja de he-

rramientas" del administrador. Hablaremos de los estándares GNU/Linux, que nos permitirán conocer aspectos comunes a todas las distribuciones GNU/Linux, es decir, lo que esperamos poder encontrar en cualquier sistema. Otra herramienta básica serán: los editores simples (o no tan simples); algunos comandos básicos para conocer el estado del sistema u obtener información filtrada según nos interese; procesos básicos de compilación de programas a partir de los códigos fuente; herramientas de gestión del software instalado, al mismo tiempo que comentaremos la disyuntiva de uso de herramientas gráficas o las de línea de comandos. En general, en la administración local, trataremos aquellos aspectos de administración que podríamos considerar "locales" en nuestro sistema. Estos aspectos pueden conformar la mayor parte de las tareas típicas del administrador a la hora de manejar elementos tales como usuarios, impresoras, discos, software, procesos, etc.

- Finalmente, **en el módulo dedicado a red**, examinaremos todas aquellas tareas de administración que engloben nuestro sistema con su "vecindario" en la red, sea cual sea su tipo, y veremos los diferentes tipos de conectividad que podemos tener con los sistemas vecinos, así como los servicios que les podemos ofrecer o recibir de ellos.

Actividades

1. Leed el manifiesto Debian en:

http://www.debian.org/social_contract

2. Documentaos sobre las diferentes distribuciones basadas en Debian: Knoppix, Linex, variedades Ubuntu. Aparte de los sitios de cada distribución, en la dirección http://www.distrowatch.com hay una buena guía de las distribuciones y su estado, así como del software que incluyen. En esta web, o bien accediendo a las comunidades o fabricantes, podéis obtener las imágenes ISO de las distribuciones.

Bibliografía

[Bar] Barrapunto. *barrapunto site*. Noticias Open Source. http://barrapunto.com

[Bul] Bulma. "Bulma Linux User Group". Documentación general y comunidades de usuarios. <http://bulmalug.net>

[Debb] Comunidad Debian. "Distribución Debian". http://www.debian.org

[Debba] Debian. "Software Libre vs Software Abierto". http://www.debian.org/intro/free.es.html

[Dis] Distrowatch. "Distribuciones Linux disponibles". Seguimiento de las distribuciones GNU/Linux y novedades de los paquetes software. Y enlaces a los sitios de descarga de las imágenes ISO de los CD/DVD de las distribuciones GNU/Linux. http://www.distrowatch.com

[PS02] Enríquez, Ricardo; Sierra, Pío (2002). *Open Source*. Anaya Multimedia.

[FHS] *FHS Standard*, 2003. http://www.pathname.com/fhs

[Fre] Freshmeat. "Freshmeat site". Listado de proyectos Open Source. http://freshmeat.org

[FSF] FSF. "Free Software Foundation y Proyecto GNU". http://www.gnu.org

[His] HispaLinux. "Comunidad Hispana de Linux". Documentación general y comunidades de usuarios. http://www.hispalinux.es

[Joh08] Johnson, Michael K. (1998). "Linux Information Sheet". *The Linux Documentation Project*.

[Lev] Levenez, Eric. "UNIX History". http://www.levenez.com/unix

Linux Journal. *Linux Journal [Revista Linux]*. Revista GNU/Linux. http://www.linuxjournal.com

Linux Magazine. *Linux Magazine*. Revista GNU/Linux. http://www.linux-mag.com/

[New] Newsforge. "Newsforge site". Noticias Open Source. http://newsforge.org

[OSDb] OSDN. "Open Source Development Network". Comunidad de varios sitios web, noticias, desarrollos, proyectos, etc. http://osdn.com

[OSIa] OSI. "Listado de licencias Open Source". http://www.opensource.org/licenses/index.html

[OSIb] OSI (2003). "Open Source Definition". http://www.opensource.org/docs/definition.php

[OSIc] OSI (2003). "Open Source Iniciative". http://www.opensource.org

[Ray98] Raymond, Eric (1998). *La catedral y el bazar*. <http://es.tldp.org/Otros/catedral-bazar/cathedral-es-paper-00.html>

[Ray02a] Raymond, Eric (2002). "UNIX and Internet Fundamentals". *The Linux Documentation Project*.

Red Hat Inc. "Distribución Red Hat". http://www.redhat.com

[Sal94] Salus, Peter H. (1994, noviembre). "25 aniversario de UNIX". *Byte España* (núm. 1).

Scientific Linux. http://www.scientificlinux.org>

[Sla] Slashdot. "Slashdot site". Sitio de noticias comunidad Open Source y generales informática e Internet. http://slashdot.org

[Sou] Sourceforge. "Sourceforge site". Listado de proyectos Open Source. <http://sourceforge.org>

[Sta02] Stallman, Richard (2002). "Discusión por Richard Stallman sobre la relación de GNU y Linux". http://www.gnu.org/gnu/linux-and-gnu.html

[Tan87] Tanenbaum, Andrew (1987). *Sistemas operativos: diseño e implementación*. Prentice Hall.

[Tan06] Tanenbaum, Andrew; Woodhull, Albert S. (2006). *The Minix Book: Operating Systems Design and Implementation* (3.ª ed.). Prentice Hall.

[Cen] The Comunity ENTerprise Operatyng System. http://www.centos.org

[Fed] The Fedora Project. http://fedoraproject.org/

[LPD] The Linux Documentation Project (LDP). Colección de Howto's, manuales y guías que cubren cualquiera de los aspectos de GNU/Linux. Documentación general y comunidades de usuarios. http://www.tldp.org

Nivel usuario

Remo Suppi Boldrito

PID_00167540

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

Como ya se ha visto en el módulo anterior, **GNU/Linux** es uno de los términos empleados para referirse a la combinación del núcleo (kernel) –equivalente desde el punto de vista de prestaciones y funcionalidad (y en algunos casos superior) a Unix– denominado **Linux**, y de herramientas de sistema GNU, todo bajo licencia GPL (licencia pública de GNU) y otra serie de licencias libres.

A pesar de que Linux es, en sentido estricto, el sistema operativo, parte fundamental de la interacción entre el núcleo y el usuario (o los programas de aplicación), se maneja usualmente con las herramientas GNU, como por ejemplo el intérprete de órdenes o comandos *bash*, que permite la comunicación con el núcleo mediante un completo conjunto de órdenes e instrucciones. Existen otros núcleos disponibles para el proyecto GNU, el conocido como Hurd y que muchos desarrolladores consideran que es el auténtico núcleo del proyecto GNU. En http://www.linuxdriver.co.il/kernel_map se puede consultar un mapa interactivo del núcleo donde se demuestra la complejidad que posee un sistema de estas características.

En este módulo veremos conceptos de nivel 0 para aprender desde el inicio los diferentes conceptos de GNU/Linux, e iremos avanzando hasta ver aspectos de inicialización y configuraciones para adecuar el sistema operativo a nuestras necesidades.

1. Introducción al sistema GNU/Linux

El sistema GNU/Linux es un sistema multitarea, es decir, permite la ejecución de centenares de tareas al mismo tiempo independientemente de la cantidad de *cores* o CPU que tenga para ejecutarse utilizando una característica común en todos los sistemas operativos modernos llamada multiprogramación. Esta característica permite ejecutar una tarea durante un determinado tiempo, suspenderla, pasar a la siguiente y así sucesivamente, y cuando se llega al final, volver a comenzar por la primera sin afectar su comportamiento o ejecución.

Normalmente esta ejecución se denomina *"round robin"*, ya que distribuye un *"quantum"* de tiempo (que oscila entre 15 milisegundos a 150 milisegundos, dependiendo del operativo) para cada tarea en espera y volviendo a comenzar por la primera cuando se llega a la última de la cola. Si el sistema posee más de un *core* o procesador, GNU/Linux tiene capacidad para distribuir estas tareas en los diferentes elementos de cómputo obteniendo las consiguientes mejoras en las prestaciones. Puede parecer que 15 milisegundos = 0,015 segundos es un tiempo pequeño, pero cualquier procesador actual puede como mínimo ejecutar en este tiempo ¡unas 2,2 millones de instrucciones máquina! Además, Gnu/Linux es un sistema operativo multiusuario que permite que más de un usuario a la vez pueda estar trabajando con el sistema y que éste con su trabajo no pueda afectar a ninguna de las tareas de los otros usuarios en el sistema.

Gnu/Linux es un sistema multitarea y multiusuario que permite (aun con un solo procesador/*core*) atender las necesidades simultáneas de múltiples usuarios, utilizando un técnica habitual de los sistemas operativos modernos denominada multiprogramación.

Todos los sistemas Unix, y Gnu/Linux no es una excepción, consideran dos tipos de usuarios diferenciados: el **superusuario** (llamado **root**), que tiene todos los permisos sobre el sistema, y el resto de los usuarios que disponen de un directorio de trabajo (*home*) del que tienen todos los permisos, pero en el resto del sistema lo que pueden hacer está en función de su importancia y nivel de seguridad para el propio sistema. Generalmente, en cualquiera de estos sistemas ***nix** (Unix, Linux...) un usuario puede "mirar" (y en algunos casos ejecutar) todo aquello que no implique información confidencial pero tiene generalmente restringido el resto de acciones.

El segundo concepto interesante en los sistemas *****nix es que se puede trabajar interactivamente en dos modos diferenciados: modo texto y modo gráfico (y en este último se puede abrir una ventana especial llamada *Terminal*, que permite trabajar en modo texto dentro del modo gráfico). Normalmente, los

modos gráficos son los más utilizados en sistemas de escritorio o de usuario doméstico, mientras que los de texto son adecuados para servidores. No obstante, como no se impone ninguna restricción se puede cambiar fácilmente de uno a otro con una secuencia de teclas o incluso estar en modo gráfico y desarrollando código en modo texto sobre un terminal, o conectado con un terminal a otra máquina o al disco de otra máquina.

El tercer concepto interesante es que la interacción entre el usuario y el núcleo se realiza a través de un intérprete de comandos/órdenes llamado *shell* y que puede ser escogido por el usuario entre los diferentes que hay. Estos *shell* permiten la ejecución (interpretación, mejor dicho, ya que el código está escrito en lenguaje texto ASCII) de pequeños programas llamados *shell scripts*, que son muy potentes para realizar secuencias de comandos (y que pueden llegar a ser muy complejas). Obviamente, al igual que otros sistemas operativos, Gnu/Linux en su interfaz gráfica soporta la interacción gráfica sobre los diferentes acciones del sistema, incluso permitiendo ejecutar *shell scripts* como si de otro programa se tratara.

Dos conceptos interesantes más, en los sistemas *nix, son la idea de tarea de usuario o del sistema operativo y la estructura del sistema de archivos. En cuanto al primer concepto, una tarea es una actividad que debe realizar el sistema operativo que bien puede ser la ejecución de un comando, una orden, editar un archivo, etc. Para ello, el sistema operativo debe ejecutar un programa adecuado para realizar esta tarea que normalmente se denomina ejecutable, ya que contiene las instrucciones máquina para realizar esta tarea. Cuando el programa ejecutable se carga en memoria se le llama proceso, o programa en ejecución, ya que contiene, además del ejecutable, todas las estructuras de datos para que se pueda realizar esta tarea, y se libera toda la memoria cuando finaliza su ejecución.

Este proceso se puede suspender, bloquear, continuar su ejecución de acuerdo a las necesidades del usuario y a lo que ordene el sistema operativo. Una derivación de este concepto de proceso es que en los procesadores modernos un proceso puede ser dividido entre varia subtareas (llamados hilos o *threads*). ¿Qué ventajas tiene un programa multithread? Que el código que ejecuta cada *thread* lo define el programador, por lo cual se puede tener un thread atendiendo una lectura de disco y otro haciendo un refresco de una imagen en pantalla simultáneamente dentro del mismo proceso. Aun teniendo un solo *core*, este tipo de programación es más eficiente que si hiciera una subtarea primero y otra después.

Nota

Los hilos de ejecución o *threads* pueden ser ejecutados independientemente por diferentes *cores*.

Por último, los sistema *nix disponen de una estructura de archivos estándar donde se ubican los archivos del sistema con total independencia a los dispositivos físicos. Es decir, a partir de una raíz (llamada root y definida por la "/") se ubican los diferentes directorios en función de sus objetivos, y donde cada usuario dispone de un directorio propio de trabajo, generalmente en el direc-

torio */home/nombre-usuario*, donde su dueño tendrá total capacidad de decisión, mientras que no así en el resto del árbol (el superusuario tiene su propio directorio en */root*).

2. Conceptos y órdenes básicas

Entrando con un poco más de detalle, en este apartado discutiremos las ideas básicas y las instrucciones necesarias para "movernos" en el sistema. La forma más simple es a través de órdenes al intérprete de órdenes (y ya veréis como es lo más eficiente cuando se adquiere un poco de práctica aunque al principio pueda parecer anticuado o complicado o las dos cosas). Este método de trabajo nos permitirá trabajar rápido con cualquier máquina en forma local o remota, ya que se tiene repetición/texto predictivo de los comandos, ejecutando órdenes complejas o programar *shell scripts* simplemente con un terminal texto. Se debe tener en cuenta que con una interfaz gráfica para un usuario novel es sumamente útil, pero extremadamente ineficiente para un usuario avanzado.

La mayoría de las órdenes, o comandos, que veremos en este apartado forman parte del estándar y son comunes a todos los sistemas GNU/Linux y a Unix (son normas IEEE POSIX). Aunque cada distribución tiene sus propias aplicaciones de administración y gestión, generalmente todas las acciones que se hacen a partir de ellas también se pueden hacer con las órdenes que veremos. A partir de éstas, podremos manipular casi todos los aspectos del sistema y movernos eficientemente.

En este apartado tenemos por objetivo aprender a utilizar correctamente estas órdenes y a navegar por cualquier sistema basado en GNU/Linux y sin que importe qué distribución estemos usando. Cada una de las órdenes del sistema suele tener multitud de parámetros diferentes. Con la utilización de los parámetros podemos, con una misma orden, ejecutar acciones diferentes, aunque todas sean de un mismo estilo. En este documento no especificaremos los diferentes parámetros de cada una de las órdenes que veremos, ya que se puede consultar el manual incluido en todo sistema *nix con la orden **man <nombre_de_la_orden>**. Es interesante comentar que si no se sabe el nombre de la orden/comando, se puede utilizar el comando **apropos acción**, que nos listará todas las órdenes, y en la que la palabra pasada como acción sale en la especificación de la orden. Por ejemplo, si ponemos **apropos copy** nos dará:

```
cp (1) - copy files and directories
cpgr (8) - copy with locking the given file to the password or gr...
cpio (1) - copy files to and from archives
cppw (8) - copy with locking the given file to the password or gr...
dd (1) - convert and copy a file
...
```

Esto indica los comandos que permiten copiar algún elemento. El parámetro de una orden/comando está precedido por un espacio o muchas veces por un "–", como por ejemplo:

```
cp -dpR /home/juan /usr/local/backup
```

Esto permite hacer una copia de respaldo de los archivos de */home/juan* en */usr/local/backup* indicando con -d que copia los enlaces simbólicos tal cual son, en lugar de copiar los archivos a los que apuntan, -p que preserva los permisos, el usuario y el grupo del archivo a copiar y -R para copiar los directorios recursivamente.

2.1. Usuario y grupos

Como hemos dicho en la introducción, todos los *nix son multiusuario y multitarea. Por este motivo es muy importante que el mismo sistema operativo incorpore mecanismos para manipular y controlar correctamente a los usuarios: el sistema de entrada e identificación (*login*), los programas que puede ejecutar, mecanismos de seguridad para proteger el hardware del ordenador, protección para los ficheros de los usuarios, etc.

Para identificar a los usuarios delante del sistema operativo, generalmente se utiliza una política de nombres estándar que suele ser poner como *login* la primera inicial del nombre del usuario seguido de su apellido y se debe recordar que se diferencian mayúsculas y minúsculas y, por tanto, *abc* es diferente de *ABC*. Los sistemas *nix organizan toda esta información por usuarios y grupos. Para entrar a trabajar interactivamente con el sistema, éste nos pedirá un *login* y una contraseña.

El *login* suele ser un nombre que identifica de manera inequívoca el usuario y si bien existen otros métodos de identificación, por ejemplo, mediante certificados digitales, es el método más habitual. Para validar el *login*, se solicita una palabra que sólo conoce el usuario y se llama contraseña (*password*). La contraseña debe ser una combinación de letras, números y caracteres especiales y no debe estar formada por ninguna palabra de diccionario o similares porque puede representar un problema de seguridad importante.

Ejemplo

Por ejemplo, si ponemos una contraseña como .MBAqcytvav34 podría parecer algo imposible de recordar fácilmente pero es un punto y las primera letras de un tango "Mi Buenos Aires querido cuando yo te vuelva a ver" y el año que se escribió el tango formando una palabra clave que será mucho más compleja de averiguar que una palabra que esté en el diccionario, ya sea al derecho o al revés.

Nota

Existen diferentes tipos de instalaciones de un sistema operativo en función de su objetivo/rol que cumplirá y las aplicaciones que tendrá instaladas: servidor, escritorio, ofimática, ocio, gateway, firewall, etc. El tipo **servidor** es aquella máquina que contiene programas que se encargan de proporcionar algún tipo de servicio (como servir páginas web, dejar que los usuarios se conecten remotamente, etc.), que en la mayoría de los casos están vinculados a una red de comunicaciones, pero no necesariamente.

El sistema de contraseñas es de tipo unidireccional, lo cual significa que nuestra contraseña no es almacenada como texto, sino que es cifrada y guardada. Cuando entramos en el sistema y escribimos nuestra contraseña, ésta se cifra y se compara con la que hay almacenada. Si coinciden, la identificación es po-

sitiva, si no coinciden, es negativa. La seguridad del sistema se basa en el algoritmo de cifrado, del cual, si es seguro, no se podrá conseguir la llave original haciendo el procedimiento inverso. Los programas que intentan romper las contraseñas de los usuarios realizan millones de cifrados de palabras a partir de diccionarios (con sistemas automáticos para derivarlas y buscar variantes) y probar si coinciden con el cifrado de alguna de las contraseñas de usuario. Por este motivo, se deben escoger cuidadosamente las contraseñas, cuidando dónde se guardan/apuntan (intentando en lo posible no hacerlo).

Actualmente, en los sistemas GNU/Linux podemos seleccionar dos tipos de cifrado posibles para las contraseñas de usuario. El que se utiliza desde los inicios de UNIX es el 3DES. El único inconveniente de este tipo de cifrado es que sólo nos permite contraseñas de 8 letras (si escribimos más, se ignoran), a diferencia del otro tipo de cifrado, denominado MD5, con el cual podemos usar contraseñas de la longitud que queramos (de hecho, MD5 es un sistema de *hashing*, pero también se puede utilizar para cifrar contraseñas de manera unidireccional). Cuanto más larga sea la contraseña, resulta más segura, con lo cual, se recomienda utilizar el segundo tipo de cifrado en sistemas de alta seguridad. De todos modos, debemos considerar que, si necesitamos usar algunos programas especiales para la gestión de usuarios, como el NIS, puede que no sean compatibles con MD5.

Nota

NIS son una serie de aplicaciones que nos permiten gestionar todos los usuarios de una misma red de manera centralizada en un solo servidor.

Los grupos de usuarios es un conjunto de usuarios con acceso al sistema que comparten unas mismas características, de forma que nos es útil agruparlos para poderles dar una serie de permisos especiales en el sistema. Un usuario debe pertenecer, al menos, a un grupo, aunque puede ser de más de uno. El sistema también utiliza todo este mecanismo de usuarios y grupos para gestionar los servidores de aplicaciones instalados y otros mecanismos. Por esta razón, además de los usuarios reales, en un sistema habrá usuarios y grupos que no tienen acceso interactivo pero están vinculados a otras tareas que se deben hacer en el operativo.

Ejemplo

Por ejemplo, el usuario Debian-Exim es un usuario definido para las funcionalidades del gestor de correo Exim que tiene ID de usuario (101) y de grupo (105), pero que no se puede conectar interactivamente (tal como lo indica el */bin/false* de la definición del usuario en el archivo */etc/passwd*).

```
Debian-exim:x:101:105::/var/spool/exim4:/bin/false.
```

Como ya hemos explicado, en todo sistema operativo hay un superusuario (*root*) que tiene privilegios máximos para efectuar cualquier operación sobre el sistema. Es necesario que este exista, puesto que será quien se encargará de toda la administración y gestión de servidores, grupos, etc. Este usuario no se debe utilizar para trabajar normalmente en el sistema. Sólo deberemos entrar como *root* cuando sea realmente necesario (en la actualidad, la mayoría de las distribuciones no permiten entrar como *root*, sino entrar como un usuario normal y luego "elevarse" como root a través del comando **su** o ejecutar coman-

dos a través del comando **sudo**) y utilizaremos otras cuentas o usuarios para el trabajo normal. De este modo, nunca podremos dañar el sistema con operaciones erróneas o con la prueba de programas que pueden ser maliciosos, etc.

Toda la información de usuarios y grupos se encuentra en los archivos siguientes:

- */etc/passwd*: información (nombre, directorio *home*, etc.) del usuario.
- */etc/group*: información sobre los grupos de usuarios.
- */etc/shadow*: contraseñas cifradas de los usuarios y configuración para su validez, cambio, etc.

Utilizar el archivo de *shadow* es opcional pero recomendable, ya que es un archivo que solo puede leer el *root* y, por lo tanto, el resto de usuarios no tendrá acceso a las contraseñas cifradas, reduciendo así la probabilidad que un usuario normal puede utilizar herramientas de fuerza bruta para averiguar las contraseñas de otros usuarios o del *root*.

Todos estos ficheros están organizados por líneas, cada una de las cuales identifica un usuario o grupo (dependiente del fichero). En cada línea hay varios campos separados por el carácter ":" y es importante saber qué son estos campos, por lo cual los exploraremos con algo más de detalle:

Nota

Información de usuarios y grupos:
- /etc/passwd: información de los usuarios.
- /etc/group: información sobre los grupos de usuarios.
- /etc/shadow: contraseñas cifradas de los usuarios y configuración para su validez, cambio, etc.

1) /etc/passwd: Por ejemplo: *games:x:5:60:games:/usr/games:/bin/sh*.

- Campo 1) Login: el nombre del usuario. No puede haber dos nombres iguales, aunque sí alguno que coincida con un grupo del sistema.

- Campo 2) Contraseña cifrada: si no se utiliza el fichero de *shadow*, las contraseñas cifradas se almacenan en este campo. Si utilizamos el fichero de *shadow*, todos los usuarios existentes deben estar también en el de *shadow* y en este campo se identifica con el carácter "x".

- Campo 3) User ID: número de identificación del usuario. Es el número con el cual el sistema identifica el usuario. El 0 es el reservado para el *root*.

- Campo 4) Group ID: el número de grupo al cual pertenece el usuario. Como que un usuario puede pertenecer además de un grupo, este grupo se denomina primario.

- Campo 5) Comentarios: campo reservado para introducir los comentarios sobre el usuario. Se suele utilizar para poner el nombre completo o algún tipo de identificación personal.

- Campo 6) Directorio de usuario: el directorio *home* del usuario que es donde este puede dejar todos sus ficheros. Se suelen poner en una carpeta del sistema (generalmente /home/login_usuario).

- Campo 7) Intérprete de órdenes: un intérprete de órdenes (*shell*) es un programa que se encarga de leer todo el que escribimos en el teclado y ejecutar los programas u órdenes que indicamos. En el mundo Gnu/linux el más utilizado es el *bash* (GNU Bourne-Again Shell), si bien hay otros (*csh*, *ksh*, etc.). Si en este campo escribimos /bin/false, no permitiremos que el usuario ejecute ninguna orden en el sistema, aunque esté dado de alta.

2) /etc/group: Por ejemplo: netdev:x:110:Debian.

- Campo 1) Nombre del grupo.

- Campo 2) Contraseña cifrada: la contraseña de un grupo se utiliza para permitir que los usuarios de un determinado grupo se puedan cambiar a otro o para ejecutar algunos programas con permisos de otro grupo (siempre que se disponga de la contraseña).

- Campo 3) Group ID: número de identificación del grupo. Es el número con el cual el sistema identifica internamente los grupos. El 0 está reservado para el grupo del root (los administradores).

- Campo 4) Lista de usuarios: los nombres de los usuarios que pertenecen al grupo, separados por comas. Aunque todos los usuarios deben pertenecer a un grupo determinado (especificado en el cuarto campo del fichero de *passwd*), este campo se puede utilizar para que usuarios de otros grupos también dispongan de los mismos permisos que tiene el usuario a quien se está haciendo referencia.

3) /etc/shadow: Por ejemplo:
root:1oE9iKzko$n9imGERnPDayiyat0XQjm.:14409:0:99999:7:::

- Campo 1) Login: debe ser el mismo nombre que se utiliza en el fichero de */etc/passwd*.
- Campo 2) Contraseña cifrada.
- Campo 3) Días que han pasado, desde el 1/1/1970, hasta que la contraseña ha sido cambiada por última vez.
- Campo 4) Días que deben pasar hasta que la contraseña se pueda cambiar.
- Campo 5) Días que deben pasar hasta que la contraseña se deba cambiar.
- Campo 6) Días antes de que caduque la contraseña en qué se avisará al usuario que la debe cambiar.
- Campo 7) Días que pueden pasar después de que la contraseña caduque, antes de deshabilitar la cuenta del usuario (si no se cambia la contraseña).
- Campo 8) Días, desde el 1/1/1970, que la cuenta es válida, pasados estos días será deshabilitada.

- Campo 9) reservado.

Cuando un usuario entra en el sistema, se le sitúa en su directorio *home* y se ejecuta el intérprete de órdenes (*shell*) configurado en */etc/passwd* en la línea correspondiente al usuario (campo 7). De este modo, ya puede empezar a trabajar. Sólo el *root* del sistema (o los usuarios de su grupo) tienen permiso para manipular la información de los usuarios y grupos, darlos de alta, de baja, etc. Cada orden para manejar a los usuarios tiene varios parámetros diferentes para gestionar todos los campos que hemos visto anteriormente.

A modo de ejemplo, podemos mencionar:

- **adduser**: nos sirve para añadir un nuevo usuario al sistema. La manera en que este se añade (si no le especificamos nada) se puede configurar en el fichero */etc/adduser.conf*. Admite un conjunto de opciones diferentes para especificar el directorio *home*, el *shell* a utilizar, etc.

- **useradd**: crea un nuevo usuario o cambia la configuración por defecto. Esta orden y la anterior nos pueden servir para efectuar las mismas acciones, aunque con diferentes parámetros.

- **usermod**: con esta orden podemos modificar la mayoría de los campos que se encuentran en el fichero de *passwd* y *shadow*, como el directorio *home*, el *shell*, la caducidad de la contraseña, etc.

- **chfn**: cambia la información personal del usuario, contenida en el campo de comentarios del fichero de *passwd* (campo 5).

- **chsh**: cambia el *shell* del usuario.

- **deluser**: elimina a un usuario del sistema, y borra todos sus ficheros según los parámetros que se le pase, hace copia de seguridad o no, etc. La configuración que se utilizará por defecto con esta orden se especifica en el fichero */etc/deluser.conf*.

- **userdel**: orden con las mismas posibilidades que la anterior.

- **passwd**: sirve para cambiar la contraseña de un usuario, su información de caducidad o para bloquear o desbloquear una determinada cuenta.

- **addgroup**: permite añadir un grupo al sistema.

- **groupadd**: lo mismo que la orden anterior, pero con diferentes parámetros.

Nota

Comandos básicos de gestión de usuarios y grupos:

- useradd
- userdef
- passwd
- groupadd
- groupdef
- gpasswd
- whoami
- groups
- id
- who
- w
- su

- **groupmod**: permite modificar la información (nombre y GID) de un grupo determinado.

- **delgroup**: elimina un grupo determinado. Si algún usuario todavía lo tiene como primario, no se podrá eliminar.

- **groupdel**: igual que en el caso anterior.

- **gpasswd**: sirve para cambiar la contraseña del grupo. Para saber qué usuario somos, podemos utilizar la orden **whoami**, que nos mostrará nuestro login.

- **groups**: sirve para saber a qué grupos pertenecemos e **id** nos mostrará usuario y grupo.

También es interesante podernos convertir en otro usuario sin salir de la sesión (orden **login** o **su**) o cambiarnos de grupo con la orden **newgrp**. Esta última orden se debe utilizar sólo cuando no se pertenece al grupo en cuestión y se sabe su contraseña (que ha de estar activada en el fichero de *group*). Si sólo necesitamos los permisos del grupo en cuestión para ejecutar una orden determinada, también podemos utilizar **sg**.

Como vemos, en GNU/Linux tenemos más de una manera para ejecutar una acción determinada. Esta es la tónica general que se sigue en el sistema: podemos editar directamente los ficheros y modificar, utilizar algunas de las órdenes que existen, crearlas nosotros mismos, etc. En definitiva, tenemos la posibilidad de seleccionar cuál es la opción que más nos satisface.

Por otro lado, y como decíamos anteriormente, GNU/Linux es un sistema operativo multiusuario, por lo cual, en un mismo momento puede haber varios usuarios conectados al sistema de manera simultánea. Para averiguar quiénes son, se puede utilizar la orden **who**, que nos muestra la lista de usuarios dentro del sistema y **w**, además, nos muestra qué es lo que están haciendo. Nos podemos comunicar con otro usuario utilizando la orden **write**, con la cual aparece el mensaje que hemos escrito en la pantalla del usuario indicado o **wall**, que escribe el contenido del fichero que hemos especificado en todos los usuarios dentro del sistema. Para activar o desactivar la opción de recibir mensajes, tenemos la orden **mesg**. También podemos hacer un chat personal con algún usuario a partir de la orden **talk**.

2.2. El sistema de ficheros y jerarquía

Todo sistema operativo necesita guardar multitud de archivos: configuración del sistema, registro de actividades, de usuarios, etc. Existen diferentes sistemas de archivos caracterizados por su estructura, fiabilidad, arquitectura, rendimiento, etc. y GNU/Linux es capaz de leer/escribir archivos en la casi totalidad de los sistemas de archivos aunque tiene su propio sistema optimizado para sus funcionalidades, como por ejemplo ext4 o ReiserFS.

El ext4 es una mejora compatible con ext3 que a su vez es una evolución del ext2 que es el más típico y extendido. Su rendimiento es muy bueno, incorpora todo tipo de mecanismos de seguridad y adaptación, es muy fiable e incorpora una tecnología denominada journaling, que permite recuperar fácilmente errores en el sistema cuando, por ejemplo, hay un corte de luz o el ordenador sufre una parada no prevista. ext4 soporta volúmenes de hasta 1024 PiB (PiB pebibyte = 2^{50} bytes ≈ 1.000.000.000.000.000 bytes), mejora el uso de CPU y el tiempo de lectura y escritura.

ReiserFS es otro de los sistemas utilizados en Linux que incorpora nuevas tecnologías de diseño y que le permiten obtener mejores prestaciones y utilización del espacio libre. Cualquiera de estos tipos de sistemas de archivos puede ser seleccionado en el proceso de instalación y es recomendable etx3/ext4 para la mayoría de las instalaciones.

Una característica muy importante de todos los sistemas *nix es que todos los dispositivos del sistema se pueden tratar como si fueran archivos (independencia del dispositivo físico). Es decir, existe un procedimiento de "montar el dispositivo" a través de la orden **mount** en un directorio del sistema y luego, accediendo a este directorio, se accede al dispositivo (incluso si el dispositivo es remoto), por lo cual no existe una identificación del dispositivo físico para trabajar con los ficheros como en otros operativos como A:, C:, etc.).

El sistema de ficheros ext2/3/4 ha sido diseñado para manejar de forma óptima ficheros pequeños (los más comunes) y de forma aceptable los ficheros grandes (por ejemplo, archivos multimedia) si bien se pueden configurar los parámetros del sistema de ficheros para optimizar el trabajo con este tipo de archivos. Como hemos mencionado anteriormente, el sistema de archivos parte de una raíz indicada por "/" y se organizan las carpetas (directorios) y subcarpetas (subdirectorios) a partir de ésta, presentando una organización jerárquica (visualizada por el comando *tree -L 1*) como:

```
/
|-- bin
|-- boot
|-- cdrom -> media/cdrom
|-- dev
```

```
|-- etc
|-- home
|-- initrd.img -> boot/initrd.img-2.6.26-2-686
|-- lib
|-- lost+found
|-- media
|-- mnt
|-- opt
|-- proc
|-- root
|-- sbin
|-- selinux
|-- srv
|-- sys
|-- tmp
|-- usr
|-- var
`-- vmlinuz -> boot/vmlinuz-2.6.26-2-686
```

Donde se muestra el primer nivel de directorios a partir del /. Todos ellos son directorios, excepto los que figuran con el carácter "->", que son enlaces a archivos y el archivo original se encuentra a la derecha de la flecha.

2.3. Directorios del sistema

En las distribuciones GNU/Linux se sigue el estándar FHS y tenemos como directorios en el raíz (los más importantes):

- **/bin:** órdenes básicas para todos los usuarios del sistema.
- **/boot**: archivos necesarios para el arranque del sistema.
- **/dev**: dispositivos del sistema.
- **/etc:** archivos de configuración del sistema y de las aplicaciones que hay instaladas.
- **/home**: directorio de las carpetas *home* de los usuarios.
- **/lib**: bibliotecas esenciales para el núcleo del sistema y sus módulos.
- **/mnt**: punto de montaje temporal para dispositivos.
- **/proc**: procesos y variables del núcleo del sistema.
- **/root**: directorio *home* para el root del sistema.
- **/sbin**: comandos especiales para el root del sistema.
- **/tmp**: archivos temporales.
- **/usr**: segunda estructura jerárquica, utilizada para almacenar todo el software instalado en el sistema.
- **/var**: directorio para los gestores de colas o *spoolers* de impresión, archivos de registro (logs), etc.

No se deben borrar estos directorios a pesar de que parezca que no se utilizan para el buen funcionamiento del sistema (muchas aplicaciones pueden no instalarse o dar errores si los directorios estándar no se encuentran definidos).

Para movernos por la estructura de directorios debemos utilizar las órdenes para hacer una lista de los contenidos y cambiar de carpeta. Cuando entramos en el sistema, es usual que el login nos sitúe en nuestro directorio *home*, que generalmente se suele indicar con el carácter "~". Si queremos ver lo que hay en el directorio en el que estamos situados, podemos hacer una lista de los contenidos utilizando la orden **ls** o en su versión más completa **ls -la**, que implica todos los ficheros (incluidos los que comienzan por un "." que no se muestran normalmente) -a y en formato largo -l. En todos los directorios existen dos entradas indicadas con "." y ".." donde la primera hace referencia al directorio/carpeta actual y la segunda al directorio/carpeta superior.

Ejemplo

Por ejemplo, si hacemos **ls .** mostrará el listado de directorio actual y si hacemos **ls ..** mostrará el listado del directorio inmediato superior.

Para cambiar de directorio podemos utilizar la orden **cd**, la cual si no le pasamos ningún parámetro nos situará en nuestro directorio *home* del usuario que la ha ejecutado. Todos los comandos aceptan direcciones relativas, por ejemplo, *ls ./grub* si estamos en el directorio /boot nos listará el contenido del directorio /boot/grub (forma relativa) o *ls /boot/grub* también nos listará el contenido del directorio /boot/grub, pero desde cualquier lugar que estemos (forma absoluta). Otro comando útil para saber dónde estamos parados es el **pwd**, que nos indicará en qué directorio estamos.

2.4. Enlaces

Un elemento muy utilizado en los archivos son los enlaces o vínculos. Un enlace es un puente a un archivo o directorio y representa una referencia que podemos poner en cualquier lugar que nos interese y que actúa como un acceso directo a cualquier otro.

Este mecanismo nos permite acceder a carpetas o ficheros de forma segura y cómoda, sin tener que desplazarse por la jerarquía de directorios.

Ejemplo

Si necesitamos acceder frecuentemente al archivo */etc/network/if-up/mountnfs*, por ejemplo, podemos utilizar un enlace en nuestro directorio con la orden *ln -s /etc/network/if-up /mountnfs nfs-conf* y haciendo un *cat nfs-conf*, tendremos el mismo resultado que *cat /etc/network/if-up/mountnfs* evitando tener que introducir cada vez toda la ruta completa.

En este caso hemos creado un enlace simbólico (parámetro -s) o sea, que si borramos el archivo el enlace quedará apuntando a nada y dará un error cuando ejecutemos el comando *cat nfs-conf*, pero este tipo de enlace se puede hacer a cualquier recurso y en cualquiera de las particiones del disco. La otra posibilidad es hacer un enlace fuerte (*hard link*) permitido sólo para recurso en la misma partición y que si borramos el archivo, el enlace queda activo hasta que no haya más enlaces apuntando a este archivo (momento en el cual se

borrará el archivo destino). Este recurso se debe utilizar con cuidado (sólo el *root* puede hacer enlaces fuertes a directorios) ya que permite ocultar a qué archivo está apuntando, mientras que con un enlace simbólico se puede ver el archivo destino.

2.5. Permisos

Dado que los sistemas *nix son multiusuario, necesitamos que los archivos almacenados deban tener un serie de propiedades que permitan leer, escribir y ejecutar (parámetros r,w,x *read, write, execute*). Para ello Gnu/Linux puede trabajar con un método simplificado (llamado *Access Control List* reducidas) en los que para cada elemento en el sistema de archivo se consideran tres bits (que representan los atributos para rwx) y se aplican para el dueño del elemento (*owner*), tres para el grupo y tres para el resto de usuarios. Por lo cual, para cada elemento (archivo, directorio, dispositivo, enlace, etc.) existen 9 bits además de la identificación a quien pertenece el elemento (uid) y a qué grupo pertenece (gid). Cuando hacemos ls -l tendremos para cada elemento una salida como:

```
-rwxr-xr-x 1 root root 4297 2008-01-18 06:09 mountnfs
```

Los primeros diez caracteres (empezando por la izquierda) nos indican los permisos del fichero de la siguiente manera:

- Carácter 1: indica el tipo de archivo, lo más comunes "–" para un archivo, "d" para un directorio, "l" para un enlace.

- Caracteres 2, 3, 4: nos indican, respectivamente, los permisos de lectura, escritura y ejecución para el propietario del fichero. En el caso de no tener el permiso correspondiente activado, se encuentra el carácter "-" y si no "r", "w" o "x". En el tercer carácter, además, nos podemos encontrar una "s", que nos indica si el archivo es de tipo SetUserId, que significa que en ejecutarlo obtendrá los permisos del propietario del fichero. Si sólo tiene el permiso "x", cuando el programa ejecuta lo hace con los permisos de quien la haya lanzado.

- Caracteres 5, 6, 7: estos caracteres tienen exactamente el mismo significado que los anteriores, pero hacen referencia a los permisos concedidos a los usuarios del grupo al que pertenece el archivo.

- Caracteres 8, 9, 10: igual que en el caso anterior, pero para los otros usuarios del sistema.

La siguiente cifra (1 en este caso) nos indica el número de enlaces fuertes que tiene el archivo. Para los directorios, este número indica cuántas carpetas hay en su interior además de los enlaces fuertes que tiene. A continuación se en-

cuentra el propietario y el grupo del archivo, seguido del tamaño (en bytes) que ocupa y la fecha de la última modificación. En todos los archivos se guarda su fecha de creación, del último acceso y de la última modificación, que podemos manipular con la orden **touch**. Al final se encuentra el nombre del fichero, en el que se diferencian minúsculas de mayúsculas y podemos tener todo tipo de caracteres sin ningún problema (aunque luego será más complicado ejecutar órdenes con ellas ya que se deberá poner entre comillas ("") para que no se interpreten estos caracteres). Se recomienda utilizar: a-z A-Z . - _ 0-9.

El mecanismo de SetUserId es muy útil cuando un programa necesita tener los permisos de su propietario para acceder a ciertos archivos o hacer algún tipo de operación en el sistema. Se debe vigilar este tipo de archivos porque pueden generar problemas de seguridad en el sistema si son mal utilizados. Para cambiar los permisos de un archivo determinado podemos utilizar la orden **chmod** y esta acción sólo la puede realizar el propietario del archivo. Las dos formas más comunes de utilizar el *chmod* son: *chmod XXX nombreArchivo*, donde XXX puede estar entre 0 y 7 correspondiendo al valor en octal de los permisos rwx para el propietario (primer número), grupo segundo y público el tercero. Para sacar el número debemos considerar que 1 es el permiso concedido y 0 quitado, por ejemplo, r-x se traduce como 101 que en octal (o binario) es 5. Por lo cual, r-x--xr-- se traduce como 101001100 lo cual queda 514.

Nota

Para cambiar las protecciones de file a rwxr - - r - -:

```
chmod 744 file
```

Para cambiar de dueño y grupo:

```
chown user file
chgrp group file
```

La otra manera de utilizar la orden es indicar de manera explícita qué permiso queremos dar o eliminar al archivo indicando con las letras u,g,o al usuario, grupo o resto respectivamente, un + para agregar el permiso y un – para quitarlo y "r", "w", "x" o "s" (este último para el SetUserId) para el permiso en concreto. Por ejemplo, *chmod go +r mountfs* concedería el permiso de lectura al grupo y los otros usuarios para el archivo mountnfs. Para cambiar el propietario de un fichero, se utiliza la orden **chown**[1], y para cambiar el grupo de un archivo se puede utilizar la orden **chgrp**. Los permisos por defecto para los archivos se pueden definir con el comando *umask*, con la misma notación que para la primera forma del **chmod** pero complementado (es decir, si queremos rw-r- -r-- el valor debería ser 133).

[(1)]Comando que sólo puede utilizar el root, ya que un usuario podría hacer una acción maliciosa y luego cambiar el dueño del archivo responsabilizando a otro usuario de la acción realizada.

2.6. Manipulación, patrones, búsquedas y contenidos

La orden **rm** permite eliminar los archivos y para eliminar un directorio podemos utilizar la orden **rmdir**, aunque sólo lo borrará cuando éste esté vacío (si quisiéramos borrar completamente un directorio y todo su contenido, podríamos utilizar *rm -r* que funciona de modo recursivo). Para copiar archivos de un lugar a otro tenemos la orden **cp**, indicando el fichero o directorio origen y el lugar o nombre de destino, aunque sea en el directorio actual (si queremos mover archivo/directorio, se puede utilizar el comando **mv**).

Podemos utilizar modificadores en los nombres de los archivos/directorios como por ejemplo "*", para referirse a cualquier cadena y "?" para indicar un carácter cualquiera. Por ejemplo, si queremos listar todos los archivos que comiencen por a y terminen por x será *ls a*x*, pero si queremos listar todos los archivos de tres letras que comiencen por a y terminen por x, será *ls a?x*. Se puede utilizar "[]" para una selección de caracteres, por ejemplo ls [Yy]* indicaría todos los archivos que comiencen por Y o por y, y después cualquier cosa, y podemos agregar también "!" que significa la negación de lo indicado [! Yy] es decir, que no comiencen por Y/y. Finalmente, para facilitar ciertas búsquedas, dentro de "[]" podemos especificar clases de caracteres como [:clase:], en que la clase puede ser cualquiera de:

Nota

* cualquier cosa
? un carácter
[Yy] uno u otro
[A-Z] un rango
[:clase:] una clase

- alnum [A-Za-z0-9] alpha [A-Za-z]
- blank [\] cntrl caracteres de control
- digit [0-9A-Fae-f] graph caracteres imprimibles (sin espacios)
- lower [a-z] print caracteres imprimibles (con espacios)
- punct [.,¡!¿?:;] ... space []
- upper [A-Z] xdigit [0-9A-Fa-f]

Otro tipo de operación muy útil es la búsqueda de archivos. Existen varias órdenes para hacer búsquedas de diferentes tipos: **find** es la orden más versátil buscar información sobre los archivos/directorios (por nombre, tamaño, fecha, protecciones, etc.), **locate** permite utilizar una base de datos del sistema para hacer búsquedas más rápidas que el **find** pero se debe tener en cuenta que puede dar información no actualizada (y que se puede actualizar con **updatedb**), y **whereis** indica dónde se encuentra el archivo especificado.

Como los archivos pueden ser de muchos tipos (ejecutables, texto, datos, música, etc.) en los sistemas *nix no se utiliza la extensión para identificar el tipo de archivo sino un número interno en la cabecera del archivo llamado *magic number*, que determina el tipo de archivo según sus datos (se puede utilizar la orden **file** para leer y determinar el tipo de archivo). Si necesitamos ver el contenido de un archivo, una de las órdenes básicas es **cat**, o **more** si el archivo es Ascii y lo mostrará paginado. Si hacemos un **cat** de un archivo ejecutable algunos de los caracteres pueden desconfigurar el terminal y se puede reinicializar con la orden **reset** y **clear** para borrar la pantalla. La orden **less** nos permite movernos de forma más eficiente (adelante y atrás por el archivo). Si el archivo es binario y queremos ver qué contiene, podemos utilizar las órdenes **hexdump** para ver el contenido de forma hexadecimal o **strings** para buscar las cadenas de caracteres. Con la orden **grep** le podemos pasar como segundo parámetro el nombre del archivo y como primero, el patrón que queramos buscar (con base en la sintaxis que hemos visto anteriormente, extendida a otras opciones). Además, la orden nos permite múltiples acciones más, como contar el número de líneas en las que aparece el patrón (parámetro -c), etc. Con **cut** podemos separar en campos el contenido de cada línea del fichero especificando qué carácter es el separador, muy útil en tareas de administración. También podemos visualizar un determinado número de líneas del comienzo

o del final de un archivo con las órdenes **head** y **tail**, respectivamente, y con **wc** podemos contar el número de líneas o palabras, la máxima longitud de línea de un fichero, etc. Finalmente, para comparar diferentes archivos existen varias órdenes para hacerlo: **diff**, **cmp** y **comm** hacen comparaciones de diferentes maneras y métodos en los ficheros que indicamos o **sdiff**, que además permite mezclar los datos de acuerdo a los parámetros indicados.

2.7. Procesos

Como hemos dicho en la introducción, los *nix son sistemas operativos multitareas que ejecutan procesos e hilos (threads) mediante una técnica llamada multiprogramación, que permite ejecutar más de un proceso/hilo a la vez en forma concurrente y de forma más eficiente que si lo ejecutáramos en forma secuencial, ya que se puede solapar la ejecución de entrada/salida de un proceso con la ejecución en CPU de otro proceso. Para identificar de manera inequívoca cada proceso, el núcleo del sistema les asigna un número denominado PID (*process identification*) necesario para administrar y referenciar el proceso.

Para saber qué procesos se están ejecutando, podemos utilizar la orden **ps**. Otro comando interesante para mirar la ejecución interactiva de los procesos es **top**, que mostrará la carga y estado de sistema en forma dinámica (hacer q -quit- para salir del comando).

Orden ps

Por ejemplo, *ps -edaf* muestra un conjunto de información sobre todos los procesos en ejecución y en diferentes estados.

Además de esto, podemos enviar unas señales a los procesos a fin de informarles de algún evento, los podemos sacar de la cola de ejecución, eliminarlos, darles más prioridad, etc. Saber manipular correctamente todos estos aspectos también es muy importante, ya que nos permitirá utilizar nuestro ordenador de manera más eficiente. La orden **kill** nos permite enviar señales a los procesos que nos interesen.

En general, todos los programas se diseñan para que puedan recibir este tipo de señales (incluso los scripts pueden capturar las señales). De este modo, según el tipo de señal recibida saben que deben hacer unas operaciones u otras (por ejemplo, suspender la ejecución cuando un usuario hace un Crtl-d). Para ver los tipos de señales, consultar el man **kill**. **killall** es una orden para hacer lo mismo, pero utiliza el nombre del proceso en lugar del pid, y **skill** es similar, pero con una sintaxis diferente: por ejemplo, para detener todas las ejecuciones de un usuario determinado, podríamos utilizar *skill -STOP -u login*, con lo que se terminará la ejecución de los procesos de este usuario. Además de **Crtl-d** o **Crtl-c** para finalizar un proceso (la primera es con espera hasta que el proceso termine su E/S y la segunda es en el momento que la recibe), con **Ctrl-z** podemos interrumpir un programa y revivirlo con **fg**.

Kill

Por ejemplo, con la señal TERM –que es 15– si hacemos *kill -15 PID*, que es equivalente a hacer Crtl-C en un proceso interactivo sobre un terminal, indicamos al proceso que queremos que acabe, de manera que al recibir la señal deberá guardar todo lo necesario y terminar su ejecución, si el programa no está preparado para recibir este tipo de señal, podemos utilizar la -9 (que todos los procesos la obedecen) y termina independientemente de lo que están haciendo *kill -9 PID*.

La orden **pstree** permite ver esta jerarquía de manera gráfica, y veremos que el padre de todos los procesos es un llamado *init*, ya que es el proceso inicial para poner en marcha los restantes procesos del sistema, y a partir de este nacen todos los demás, que a su vez pueden tener más hijos. Esta estructura es

muy útil para identificar de donde vienen los procesos (quien los ha puesto en marcha) y para eliminar un conjunto de ellos, ya que, al eliminar un proceso padre también se eliminan todos sus hijos.

Nota

Órdenes para procesos:

- kill -9 <pid>
- *skill -STOP -u <login>*
- *ps -edaf*
- *top*
- *pstree*
- nice [-n increment] <orden>

Un parámetro importante de los procesos es un valor llamado prioridad, que está relacionado a la CPU que recibirán este proceso (a mayor prioridad mayor tiempo de CPU). El rango de prioridades va desde el -20 al 19 (las negativas sólo las puede utilizar el root), de mayor a menor. Para lanzar un proceso con una prioridad determinada, podemos utilizar la orden **nice** y **renice** si queremos dar una prioridad diferente a un proceso que ya esté en ejecución. Por defecto, la prioridad con que ejecutan los programas es la 0. La orden **time** permite calcular el tiempo que utiliza un proceso para ejecutarse (generalmente con fines contables, por ejemplo si tenemos que facturar por el tiempo de CPU que gasta un proceso).

2.8. Órdenes complementarias

Nota

Órdenes complementarias:

a) Manuales:

- man <comando>
- man -k <palabra>

b) Comprimir:

- tar cvf <destino> <origen>
- tar zcvf <destino> <origen>
- gzip <file>
- bzip <file>

c) Espacio de disco:

- df -k
- du -k <directorio/file>

d) Parametros de sistema archivos:

- dumpe2fs <partición>

e) Sincronizar sistema de archivo:

- sync

Todas las órdenes disponen en Gnu/Linux (y en todo los *nix) de un manual completo que indica todos los parámetros y opciones (***man orden***) y se utiliza el comando *less* para visualizarlas, es por ello que podemos ir adelante y atrás con las teclas de "AvPág" y "RePág", buscar una palabra con el carácter "/" seguido de la palabra ("n" nos sirve para buscar las ocurrencias siguientes y "N", para las anteriores), "q" para salir, etc. Los manuales del sistema se dividen en diferentes secciones según su naturaleza:

1) Programas ejecutables (aplicaciones, órdenes, etc.).
2) Llamadas al sistema proporcionadas por el *shell*.
3) Llamadas a librerías del sistema.
4) Archivos especiales (generalmente los de dispositivo).
5) Formato de los archivos de configuración.
6) Juegos.
7) Paquetes de macro.
8) Órdenes de administración del sistema (generalmente aquellas que sólo el *root* puede utilizar).
9) Rutinas del núcleo.

Si hay más de un manual disponible para una misma palabra, lo podemos especificar indicando el número correspondiente de la sección que nos interesa antes de la palabra, por ejemplo *man 3 printf* (**man -k palabra** buscará entre las páginas del manual aquellas que tengan la "palabra" pasada como argumento, equivalente al **apropos** y **mandb** permitirá actualizar la base de

datos de los manuales). Si el manual no nos proporciona toda la información que necesitamos, podemos utilizar el comando **info**, que es lo mismo que el manual pero con información extendida.

Otra orden útil es la utilizada para comprimir un archivo, agrupar varios en uno solo o ver qué contiene un archivo comprimido. Si bien existen decenas de programas diferentes en todos los sistemas GNU/Linux encontraremos la orden **tar**. Este programa nos permite manipular de cualquier manera uno o varios archivos para comprimirlos, agruparlos, etc. Su sintaxis es ***tar opciones archivoDestino archivosOrigen***, donde si el archivo origen es un carpeta trabajará en forma recursiva sobre ella guardando en el archivo destino el contenido de toda la carpeta. Los parámetros comunes son c para crear y f si lo queremos guardar en un archivo (*tar cf archivol.tar o**** empaquetará todos los archivos del directorio actual que comiencen por "o"). Si además quisiéramos comprimir, podríamos utilizar czf, con lo cual se utilizaría el programa *gzip* después de empaquetarlos. Para desempaquetar un archivo determinado, el parámetro necesario es el x, de manera que deberíamos escribir *tar xf* para indicar el archivo empaquetado. Si estuviera comprimido, deberíamos pasar xzf.

El programa **gzip** (utilizado por el tar para comprimir) usa un formato de compresión propio y diferente del zip tan popular o del compress (estándar en los *nix pero obsoleto) y que se puede utilizar para comprimir un archivo en forma independiente (**gunzip** para hacer el proceso inverso o *gzip -u*). Otra aplicación de compresión bastante utilizada y que proporciona muy buenos resultados es el **bzip2**.

La gestión y manipulación de los discos duros del ordenador es otro aspecto fundamental en las tareas de administración del sistema. Más adelante se verá todo un apartado para discos, pero ahora trataremos las órdenes útiles para obtener información de los discos. El disco duro se divide en particiones, a las que podemos acceder como si se tratara de un dispositivo independiente, y las denominaremos unidad. Esto es muy útil porque nos permite separar de forma adecuada la información que tengamos en el sistema, tener más de un sistema operativo instalado en el mismo disco, etc. La orden **df** nos mostrará, de cada unidad montada en el sistema, el espacio que se ha utilizado y lo que queda libre y la orden **du** nos muestra realmente lo que nos ocupa un fichero en disco o un directorio (estos nos mostrará en bloques de discos pero con el parámetro -k en kilobytes).

Un disco está organizado en pistas y dentro de las pistas en sectores (zonas donde se guardará la información) y como este último valor es configurable, cuando se crea el sistema de archivo con fines de optimizar las prestaciones de disco/espacio utilizado nos puede interesar ver estos parámetros (para sistemas ext2/3/4), por lo que podemos utilizar la orden **dumpe2fs partición** y averiguar los parámetros con los cuales ha sido creado el disco.

Otro concepto muy extendido es la desfragmentación de un disco que no es más que la reorganización de los bloques de los ficheros para que queden en lugares consecutivos y su acceso sea más rápido. En los sistemas de ficheros que utilizamos con GNU/Linux no es necesario desfragmentar los discos (aunque hay programas con este fin), ya que el sistema se encarga automáticamente de tener el disco siempre desfragmentado, y además, en el proceso de arranque siempre se comprueban los errores y se realiza una desfragmentación total si es necesaria.

La optimización del sistema de archivos de GNU/Linux proviene de que todas las funciones del núcleo que se encargan de la gestión de ficheros utilizan unos métodos para agilizar los procesos de lectura y escritura. Uno de ellos es la utilización de una caché de disco para evitar estar constantemente leyendo y escribiendo en el disco físico (proceso lento y costoso). Esto puede representar problemas si tenemos un corte de alimentación, ya que las últimas operaciones de lectura/escritura no se habrán salvado por estar en memoria. El programa **fsck** comprueba y arregla un sistema de ficheros que haya quedado en este estado. Aunque lo podemos ejecutar cuando se desee, el mismo sistema operativo lo ejecuta cuando en el proceso de arranque detecta que el sistema no se cerró adecuadamente. Por eso, para apagar el ordenador correctamente debemos ejecutar la orden **shutdown**, que se encarga de lanzar todos los procesos necesarios para que los programas terminen, se desmonte el sistema de ficheros, etc. En este sentido, el sistema de ficheros ext3/4 es más eficaz que el ext2, ya que el journaling le permite recuperar más información de los archivos perdidos y en forma más eficiente (la integridad física de una partición se puede comprobar con la orden **badblocks**). Si bien no es aconsejable, se puede desactivar la caché de disco y si queremos en algún momento volcar de caché a disco, para evitar problemas podemos ejecutar la orden **sync**.

Se debe tener en cuenta que la mayoría de las órdenes mencionadas se deben ejecutar como *root* (o algunos de los usuarios que formen parte del grupo de *root*).

3. Instalación y arranque de GNU/Linux (conceptos básicos)

En este apartado veremos los pasos esenciales que se siguen en la mayoría de los procesos de instalación de GNU/Linux y que serán complementados posteriormente con los talleres de instalación. Si bien cada distribución tiene su propio entorno de instalación, en todas ellas hay unos pasos básicos para instalar el sistema operativo y que se describirán de modo resumido. Es importante notar que hoy en día cualquiera de las distribuciones tiene una instalación muy optimizada que necesita muy poca atención del usuario, ya que obtiene información del hardware subyacente, por lo cual, generalmente, para un usuario novel no es necesario tomar decisiones importantes (distribuciones como Ubuntu o Fedora, por ejemplo, son prácticamente instalaciones automáticas).

También debemos tener en cuenta que un usuario novel puede iniciar su camino en el mundo Linux con otro tipo de ejecuciones de GNU/Linux que no modifican el ordenador y permiten trabajar en el sistema operativo sin tener que instalar otro. Es altamente recomendable iniciar los primeros pasos sobre un **GNU/Linux Live**: el usuario debe bajarse la imagen del sistema operativo, crear un CD/DVD con ella y arrancar desde este dispositivo sin tocar el disco duro de la máquina. Este tipo de distribuciones (es recomendable utilizar Knoppix por su eficiencia y versatilidad en sus versiones para CD, DVD o USB) tienen el "inconveniente" de que para salvar el trabajo del usuario se debe hacer sobre un dispositivo de disco USB, ya que si se salva sobre el sistema de archivo, al estar éste en RAM se perderán los datos salvados.

Nota

VirtualBox permite arrancar una imagen o instalar un SO sobre otro SO huésped (*host*) tanto en 32 bits como en 64. Si no se desea hacer una instalación desde 0 se puede encontrar gran cantidad de distribuciones con sus imágenes ya realizadas, como por ejemplo en http://virtualboxes.org/images/

Otra opción totalmente recomendable como primeros pasos (o como forma de trabajo habitual) sin tener que tocar el sistema operativo de una máquina es trabajar con máquina virtualizadas. Para ello, se recomienda utilizar **VirtualBox**, que permite arrancar una imagen o instalar una sobre un sistema operativo huésped (*host*), tanto en 32 bits como en 64 bits es altamente configurable y si no se desea hacer una instalación se puede encontrar gran cantidad de distribuciones con sus imágenes ya realizadas, como por ejemplo en http://virtualboxes.org/images/, que cuenta con aproximadamente 30 distribuciones de Linux y 15 distribuciones de otros sistemas *nix o incluso Android o sistemas no *nix.

Es importante que antes de instalar un nuevo sistema conozcamos adecuadamente los componentes hardware que tenemos instalados en nuestro ordenador para poder configurarlo adecuadamente, aunque la distribución que utilizamos incorpore detección de *hardware*. Es posible que en un solo disco duro tengamos instalados dos sistemas operativos (*dualboot*) o más, totalmente

independientes, y si bien el proceso de instalación de otro sistema operativo en el mismo disco no debería interferir con las particiones de los demás, es aconsejable hacer copias de seguridad de todos los documentos importantes.

Es necesario, antes de comenzar, tener información de la marca y el modelo de la tarjeta gráfica, la de sonido, la de red, la marca, el tipo y las características del monitor, así como cualquier otro hardware especial que tengamos (el resto del hardware será detectado por el sistema: placa base, la CPU y la memoria RAM).

Generalmente, todas las distribuciones de GNU/Linux proporcionan algún tipo de medio para el arranque del proceso de instalación, siendo el más común tener un CD o DVD de arranque, para lo cual es necesario configurar la BIOS para que pueda arrancar (*boot*) desde CD/DVD. La instalaciones son autoguiadas y es importante prestar atención a la selección del idioma y del teclado para evitar problemas desde el inicio (si bien se podrá configurar posteriormente).

Para usuarios ya avanzados, existe alguna otra forma de instalar GNU/Linux que permite hacer la instalación desde cualquier medio: FTP, HTTP, disco duro, NFS, USB, pero es recomendable no utilizarlas como primera experiencia.

La partición del disco duro es una de las partes más críticas de todo el proceso, ya que significa dividir el disco duro en varias secciones que serán consideradas independientes. Si ya tenemos un sistema operativo instalado en nuestro ordenador, el disco estará particionado en una o varias particiones, pero si el disco es nuevo, tendrá una única partición. Para instalar GNU/Linux debemos disponer, al menos, de una partición para uso propio (si bien es posible instalarlo sobre otros sistemas de archivos, no es recomendable esta opción por cuestiones de rendimiento y fiabilidad) y otra más pequeña para una extensión de la memoria RAM del ordenador llamada partición de SWAP (generalmente del doble de la memoria RAM instalada).

El procedimiento más común para reducir, crear o cambiar el tamaño de las particiones es utilizar herramientas como las disponibles en Windows Vista, 7 (administración de Discos o la aplicación **fips** con licencia GPL y para sistemas FAT) o en cualquier Linux Live. Se puede utilizar **gparted** para modificar el tamaño de una partición ya creada sin perder el contenido de la misma (si bien se recomienda hacer copias de seguridad de los archivos más importantes). El procedimiento recomendable es arrancar con una Linux Live y utilizar el comando *gparted*, que es muy eficiente y seguro.

Nota

La información de particiones y el programa de carga de un sistema operativo se guardan en una zona de datos reservada llamada MBR (*master boot record*) sobre el primer disco.

Como primer paso es recomendable que GNU/Linux utilice dos particiones en el disco duro (una para el sistema de ficheros y la otra para la swap). Si bien todas las distribuciones tienen un particionamiento guiado, se puede hacer en forma manual con diferentes utilidades (*fdisk, cfdisk, diskDruid,* etc). La forma

en que GNU/Linux identifica los discos es con */dev/hdX* para los discos IDE y */dev/sdX* para los SCSI y Serial ATA, en los cuales X es una letra, correspondiente al disco al que nos queramos referir: */dev/hda* es el maestro del primer canal IDE, */dev/hd*b el segundo y así sucesivamente (o */dev/sda* el primer disco SCSI o SATA y */dev/sdb* el segundo, etc.). La aplicación de instalación nos hará un listado de los discos y deberemos escoger sobre cual de ellos queremos realizar la instalación.

Cuando creamos una partición podremos escoger entre primaria o lógica. En un disco duro podemos tener hasta 4 particiones primarias y hasta 64 lógicas. Si no necesitamos más de 4 particiones, podemos elegir cualquiera de los dos tipos. Si necesitamos más, deberemos tener en cuenta que las lógicas se sitúan dentro de una primaria (hasta un máximo de 16 para cada una), de manera que no podemos tener 4 particiones primarias creadas y luego añadir otras lógicas. En este caso, deberíamos crear 3 de primarias y hasta 16 lógicas en la cuarta partición primaria.

Nota

De todas las particiones de un disco duro podemos elegir una para que sea la activa. Este flag sirve para indicar a la BIOS o al EFI del sistema (sistema de inicialización y carga del sistema operativo) cuál es la partición que debe iniciar si en el MBR no encuentra ningún programa de arranque.

Cuando se crea una partición se debe indicar qué sistema de ficheros utilizará (Linux ext3, Linux ext4, Linux swap u otro) y una vez hechas las particiones, guardaremos la configuración y tenemos que indicar al proceso de instalación dónde queremos situar la raíz del sistema de ficheros (*root filesystem*) y el *swap* del sistema, a partir de este momento se podrá continuar con la instalación del mismo.

Una parte importante de la instalación son los módulos del núcleo que son partes de software especializadas que trabajan con alguna parte del hardware o del sistema. En las distribuciones actuales simplemente se debe seleccionar que dispositivo se tiene (monitor, red, sonido gráficos) y la instalación cargará prácticamente todos los módulos necesarios aunque en la mayoría tienen procesos de autodetección, por lo cual no será necesario seleccionar prácticamente nada. Si algún módulo no se incluye durante la instalación, es posible hacerlo luego con comandos como ***insmod*** o ***modprobe*** (para añadir un nuevo módulo), ***lsmod*** (para hacer una lista de los instalados), ***rmmod*** (para eliminar alguno) y también ***modprobe*** (para probar alguno y, si funciona correctamente, incluirlo en el núcleo). Todos estos módulos son ficheros binarios que solemos encontrar en el directorio */lib/modules/versión-del-operativo/* del sistema.

Después de configurar los módulos que se incluirán en el núcleo del sistema operativo, tendremos que configurar la red (si tenemos la tarjeta necesaria). Aunque en este documento no entraremos en detalle sobre redes, describiremos los conceptos necesarios para poder dar este paso de forma básica. El primer dato que solicitará la instalación es el nombre del sistema (para referirnos a él de forma amigable) y a continuación pedirá si en nuestra red utilizamos un mecanismo llamado DHCP (consiste en tener un servidor especial que se encarga de asignar automáticamente las IP a los ordenadores que arrancan). Si utilizamos este mecanismo, debemos indicarlo, y si no, se nos preguntará por la IP (cuatro números separados por punto entre 0 y 255) y máscara de nuestro

ordenador (cuatro números entre 0-255 siendo común utilizar 255.255.0.0). Si no conocemos estos datos, debemos dirigirnos al administrador de nuestra red. Seguidamente deberemos introducir la IP del gateway de nuestra red (gateway es el dispositivo u ordenador que actúa de puente entre nuestra red local e Internet y si no tenemos ningún dispositivo de este tipo, podemos dejar en blanco este campo).

A continuación, debemos especificar el servidor (o servidores) de nombres que utilizamos llamado DNS, que es una máquina que nos proporciona la equivalencia entre un nombre y una dirección IP (es decir, nos permitirá conocer por ejemplo la IP de www.uoc.es en forma transparente). Si no sabemos cuáles son, deberemos recurrir al administrador de la red.

Nota

Una dirección IP es la identificación de un ordenador dentro de una red cuando utilizamos el protocolo TCP/IP (protocolo utilizado en Internet).

Si estamos en una red local podemos consultar al administrador para que nos proporcione toda la información necesaria o, si tenemos otro sistema operativo instalado en el ordenador, podremos obtener esta información de él, pero en ningún caso debemos inventar estos valores, ya que si el ordenador está conectado a una red local puede generar problemas a otros ordenadores.

Una vez configurados estos aspectos, deberemos seleccionar si queremos instalar un pequeño programa en el disco duro para que en el proceso de arranque del ordenador podamos elegir qué sistema operativo de los que tenemos instalados queremos arrancar (incluso si sólo hemos instalado GNU/Linux). Las aplicaciones más usuales son el **Lilo** (*Linux loader*) o el **Grub** (recomendado) (*GNU, Grand Unified Bootloader*), que tienen por objetivo iniciar el proceso de carga y ejecución del núcleo del sistema operativo que le indiquemos interactivamente o por defecto después de un tiempo de espera. Todas las distribuciones (si no hay problemas) detectan si tenemos algún otro sistema operativo instalado en los discos duros y configuran automáticamente el sistema de arranque. Este programa generalmente se instala en la MBR del disco maestro del primer canal IDE o SCSI, que es el primer lugar que la BIOS o EFI del ordenador inspecciona buscando un programa de estas características.

El último paso en la instalación es la selección de paquetes a instalar además de los estrictamente necesarios para el funcionamiento básico del sistema operativo. La mayoría de los procesos de instalación incluyen dos formas de seleccionar los programas del sistema: básico o experto. Con el proceso de selección básico, se agrupan los paquetes disponibles para grandes grupos de programas: administración, desarrollo de software, ofimática, matemáticas, etc. opción recomendable para dar los primeros pasos. Si no seleccionamos un paquete luego se podrá hacer una instalación posterior con la herramienta de que disponen todas las distribuciones para instalar o desinstalar paquetes. Debian GNU/Linux fue una de las primeras en incluir aplicaciones para gestionar los paquetes, se llama **apt** y es muy útil para hacer el mantenimiento y actualización de todos los paquetes instalados incluso en el mismo sistema operativo.

Si la instalación no ha funcionado correctamente, puede pasar que no podamos arrancar ninguno de los sistemas operativos instalados (ni el nuevo ni el anterior), pero todas las distribuciones tienen un apartado en el arranque de "rescate" (*rescue mode*), que nos permitirá arrancar el sistema GNU/Linux desde el CD/DVD, acceder al disco duro y arreglar aquellas cosas que no han funcionado o recuperar el sistema operativo inicial, si bien para algunos casos son necesarios una serie de conocimientos avanzados en función de cuál haya sido la causa de error.

4. Configuraciones básicas

4.1. El sistema de *login*

Tanto en modo gráfico como en modo texto, el procedimiento de identificación y entrada se denomina login. Generalmente, el sistema arrancará en modo gráfico, pero podemos pasar a modo texto seleccionando el tipo de sesión en el panel de entrada o con *Crtl-Alt-Backspace* (terminando la ejecución del servidor gráfico).

En modo texto se lanzan 5 terminales independientes, a las que se puede acceder mediante *Alt-F1*, *Alt-F2*, etc., que permitirá trabajar simultáneamente con diferentes cuentas al mismo tiempo. El proceso de *login* pone en pantalla, primero, un mensaje que se puede modificar desde el archivo */etc/issue* y que admite diferentes variables (*\d fecha actual, \s nombre del SO, \t hora actual*, etc.) y en segundo lugar el mensaje del día desde */etc/motd* (creando un fichero vacío llamado *.hushlogin*, en su directorio *home* se anula este mensaje). A continuación el proceso de *login* lanza el *shell* por defecto para el usuario (indicado en el último campo de */etc/passwd*).

El *shell* ejecuta el archivo *.profile* del directorio home del usuario para las opciones por defecto de este mismo usuario y se complementa con el */etc/profile*, que configura opciones por defecto para todos los usuarios. Cada *shell* además posee archivos de configuración propios, como por ejemplo el *shell Bash*, y ejecuta además dos ficheros más llamados *.bashprofile* (que se ejecuta en cada *login*) y *.bashrc* (que se ejecuta cada vez que abrimos un nuevo terminal). Veremos algunas de las instrucciones que podemos encontrar en estos archivos:

```
# ~/.profile: ejecutado por el shell durante el login
# No será leído por bash si existe ~/.bash_profile o ~/.bash_login
# Ver ejemplos en /usr/share/doc/bash/examples/startup-files

# EL valor por defecto de umask
umask 022

# Si está ejecutando bash ejecuta .bashrc si existe
if [ -n "$BASH_VERSION" ]; then
if [ -f "$HOME/.bashrc" ]; then
. "$HOME/.bashrc"
fi
```

```
fi

# Incluye en PATH un directorio bin del usuario
if [ -d "$HOME/bin" ] ; then
PATH="$HOME/bin:$PATH"
fi
# Cambia el prompt
export PS1='\h:\w\$ '

# ~/.bashrc: ejecutado por bash(1) para non-login shells.
# Si no se ejecuta interactivamente no hace nada.
[ -z "$PS1" ] && return

# Habilita el soporte de color para el comando ls
if [ -x /usr/bin/dircolors ]; then
eval "`dircolors -b`"
alias ls='ls --color=auto'
alias dir='dir --color=auto'
fi

# otros alias
alias ll='ls -l'

# habilita programmable completion features
if [ -f /etc/bash_completion ]; then
. /etc/bash_completion
fi
```

Como se puede observar en estos ficheros (que son un ejemplo de *shell script* y que veremos más adelante), se incluyen diferentes definiciones de variables (PATH, por ejemplo, que es la variable donde se buscarán los programas para su ejecución sin tener que incluir todo el camino; PS1, que es la variable que almacena el prompt, que es el carácter que sale a la izquierda de la línea de comando acabado en $ o % para el usuario normal y en # para el *root*). También tenemos alias de órdenes o ejecución condicional de otros archivos (en el primero si se está ejecutando el bash, ejecuta el *$HOME/.bashrc*). Si se quiere ejecutar programas del directorio desde el que estamos situados sin necesidad de poner "./" al principio, podríamos añadir esta entrada en la declaración del PATH, pero generalmente no se incluye, ya que puede representar un agujero de seguridad. Para el **prompt** hemos utilizado la orden **export** para definir las llamadas variables de entorno, que se mantienen durante toda la sesión y se pueden consultar con la misma orden.

Con **set** y **unset** también podemos inicializar o quitar otras variables/atributos representados por defecto (con **echo $variable** podremos consultar el valor de cada variable). Algunas de bash son:

PWD: directorio actual.

LOGNAME: nombre del usuario conectado.

BASH_VERSION: versión del bash que utilizamos.

SHELL: *Shell* utilizado.

RANDOM: genera un número aleatorio diferente cada vez que mostramos su contenido.

SECONDS: número de segundos que han pasado desde que hemos abierto el *shell*.

HOSTNAME: nombre del sistema.

OSTYPE: tipo de sistema operativo que estamos utilizando.

MACHTYPE: arquitectura del ordenador.

HOME: directorio home del usuario.

HISTFILESIZE: tamaño de archivo de historia (número de órdenes que se guardan).

HISTCMD: número de orden actual en la historia.

HISTFILE: fichero en el que se guarda la historia de órdenes.

La configuración adicional del login gráfico se realizará por medio del entorno gráfico y dependerá del tipo de entorno.

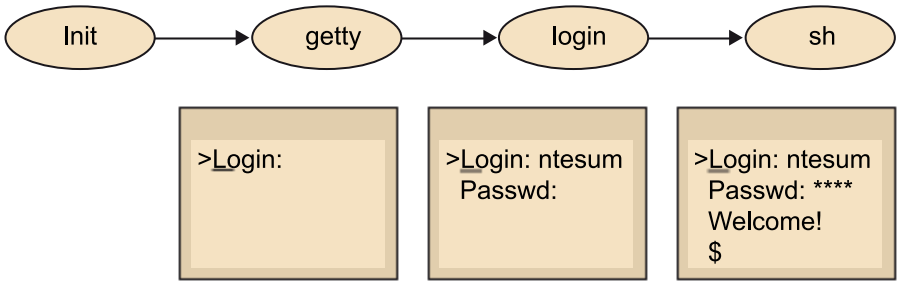
4.2. El intérprete de comandos (*shell*)

Como ya hemos avanzado, el término genérico *shell* se utiliza para denominar un programa que sirve de interfaz entre el usuario y el núcleo (*kernel*) del sistema GNU/Linux. En este apartado veremos alguna características básicas de los *shells* interactivos de texto, que es el programa que verá el usuario (y lo atenderá) una vez realizado el procedimiento de *login*.

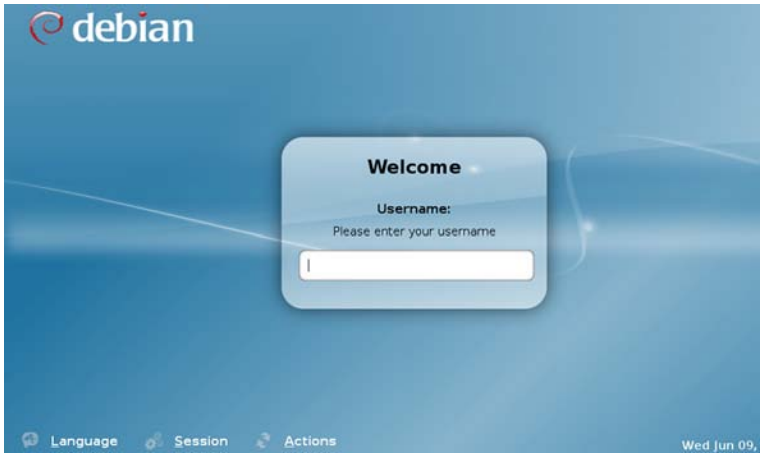
El *shell* es lo que los usuarios ven del sistema, ya que el resto del sistema operativo permanece esencialmente oculto a sus ojos. El *shell* está escrito de la misma forma que un proceso (programa) de usuario; no está integrado en el *kernel* y se ejecuta como un programa más del usuario.

Cuando el sistema GNU/Linux arranca, suele presentar a los usuarios una interfaz de entrada que puede ser gráfica o en modo texto. Estas diferentes entradas dependerán de los modos (o niveles) de arranque del sistema (*runlevels*), que permitirá configurar los modos de arranque del sistema operativo.

La siguiente figura muestra el arranque *shell* en modo texto y los procesos de sistema involucrados (OKe).



En el modo de arranque gráfico, la interfaz está compuesta por algún gestor de acceso que administra el proceso de *login* del usuario desde una pantalla (caratula) gráfica, en la que se solicita la información de entrada correspondiente: su identificador como usuario y su palabra de paso (o *passsword*). En GNU/Linux suelen ser habituales los gestores de acceso: xdm (propio de X Window), gdm (Gnome) y kdm (KDE), así como algún otro asociado a diferentes gestores de ventanas (*window managers*) como muestra la figura siguiente:



Una vez validado el acceso, el usuario encontrará una interfaz gráfica de X *Window* con algún gestor de ventanas, como Gnome o KDE (ver último punto del capítulo). Desde el modo gráfico también es posible "interactuar" con el sistema a través de un entorno de trabajo en modo texto simplemente abriendo un "emulador de terminal" o simplemente terminal (p. ej., Xterm) desde los menús de la interfaz gráfica (en Gnome → Aplicaciones → Accesorios → Terminal).

Si el acceso es por modo texto (llamado también modo consola) y una vez identificados obtendremos el acceso al *shell* en forma interactiva (se puede pasar del modo gráfico al texto con Crtl+Alt+F1 a F5, que permitirá tener 5 consolas diferentes y volver con Crtl+Alt+F7). Otra forma de trabajo con un *shell* interactivo es a través de una conexión remota desde otra máquina conectada mediante red y con aplicaciones tales como telnet o rlogin (poco utilizadas por inseguras), ssh, o gráficas como los emuladores X Window.

Una vez iniciado el *shell* interactivo [Qui01], se muestra un *prompt* (símbolo o secuencia de caracteres como $, %, # –utilizado generalmente para identificar el usuario *root* –o también algo configurable por el usuario como MySys) indicándole al usuario que puede introducir una línea de comando. Tras la introducción, el *shell* asume la responsabilidad de validar la sintaxis y poner los procesos necesarios en ejecución, mediante una serie de secuencias/fases:

1) Leer e interpretar la línea de comandos.

2) Evaluar los caracteres "comodín" como $ * ? u otros.

3) Gestionar las redirecciones de E/S necesarias, los *pipes* y los procesos en segundo plano (*background*) necesarios (&).

4) Manejar señales.

5) Preparar la ejecución de los programas.

Normalmente, las líneas de comandos podrán ser ejecuciones de comandos del sistema, comandos propios del *shell* interactivo, puesta en marcha de aplicaciones o *shell scripts* (secuencias de comandos, variables, sentencias de control, etc., que generalmente se encuentran en un archivo ASCII).

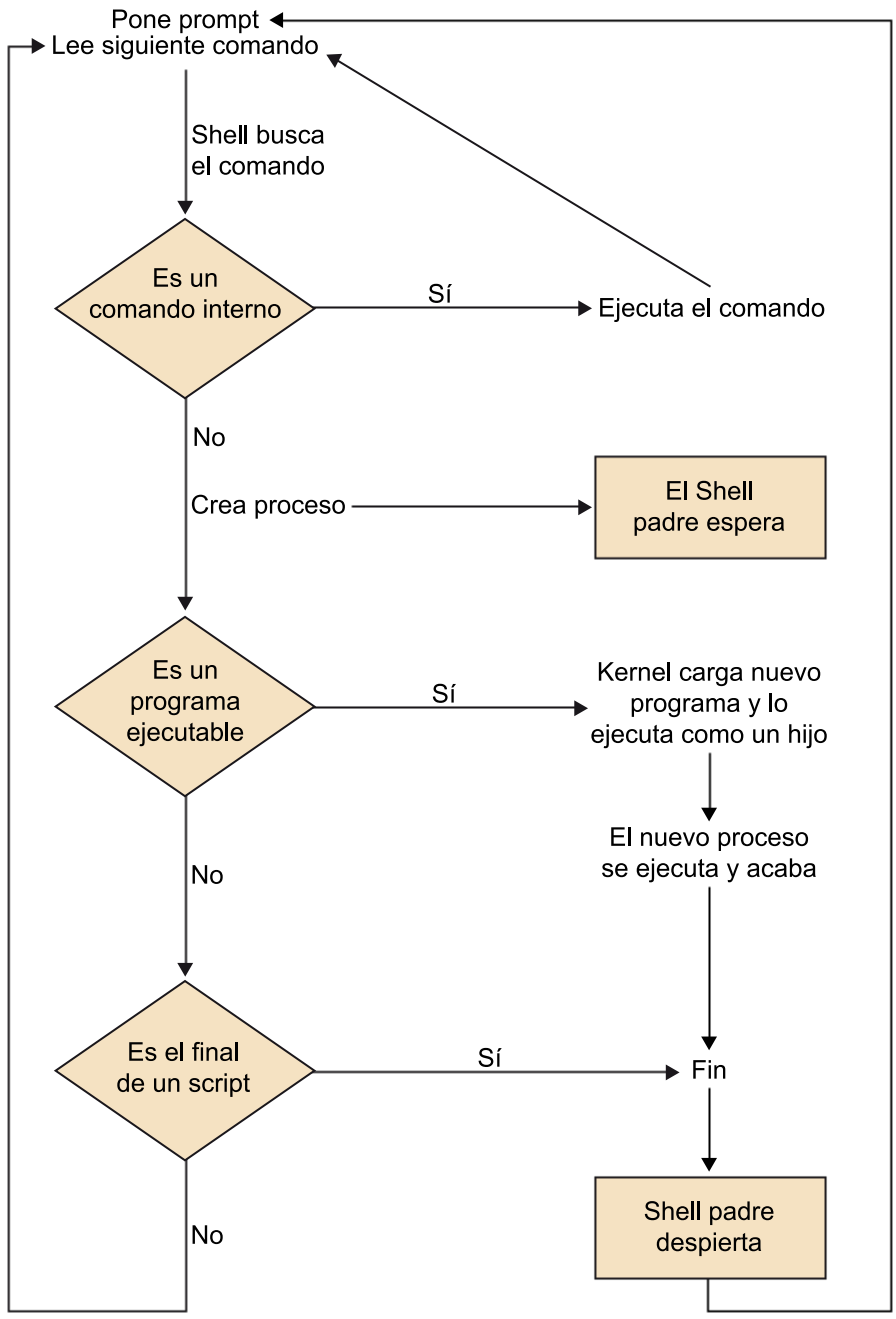
Los ficheros de *script* son directamente ejecutables por el sistema bajo el nombre que se haya dado al fichero. Para ejecutarlos, se invoca el *shell* junto con el nombre del fichero, o bien se dan permisos de ejecución al *shell script*.

En cierta manera, podemos ver el *shell script* como código de un lenguaje interpretado que se ejecuta sobre el *shell* interactivo correspondiente. Para el administrador, los *shell scripts* son muy importantes básicamente por dos razones:

1) La configuración del sistema y de la mayoría de los servicios proporcionados se hacen mediante herramientas en forma de **shell scripts**.

2) La principal forma de automatizar procesos de administración es mediante la creación de *shell scripts* por parte del administrador.

La figura siguiente muestra el flujo de control básico de un *shell*:



Todos los programas invocados mediante un *shell* poseen tres ficheros predefinidos, especificados por los correspondientes descriptores de ficheros (*file handles*). Por defecto, estos ficheros son:

1) standard input (entrada estándar): normalmente asignada al teclado del terminal (consola); usa el descriptor número 0 (en UNIX los ficheros utilizan descriptores enteros).

2) standard output (salida estándar): normalmente asignada a la pantalla del terminal; usa el descriptor 1.

3) ***standard error*** (salida estándar de errores): normalmente asignada a la pantalla del terminal; utiliza el descriptor 2.

Esto nos indica que cualquier programa ejecutado desde el *shell* tendrá por defecto la entrada asociada al teclado del terminal, su salida hacia la pantalla y, en el caso de producirse errores, también los envía a la pantalla.

Además, los *shells* pueden proporcionar tres mecanismos siguientes de gestión de la E/S:

1) Redirección: dado que los sistemas *nix tratan de la misma forma a los dispositivos de E/S y los ficheros, el *shell* los trata a todos simplemente como ficheros. Desde el punto de vista del usuario, se pueden reasignar los descriptores de los ficheros para que los flujos de datos de un descriptor vayan a cualquier otro descriptor; a esto se le llama redirección. Por ejemplo, nos referiremos a la redirección de los descriptores 0 o 1 como a la redirección de la E/S estándar. Para ello se utilizan los símbolos › » ‹ «, por ejemplo ls › dir.txt redirecciona la salida del comando ls a un fichero llamado dir.txt, que si existe se borra y guarda la salida del comando y si no, se crea y se guarda la salida. Si fuera ls » dir.txt añade al final si el fichero existe.

2) Tuberías (*pipes*): la salida estándar de un programa puede usarse como entrada estándar de otro por medio de *pipes*. Varios programas pueden ser conectados entre sí mediante *pipes* para formar lo que se denomina un *pipeline*. Por ejemplo ls | sort | more donde la salida del comando ls se ordena (*sort*) y la salida de este se muestra por pantalla en forma paginada.

3) Concurrencia de programas de usuario: los usuarios pueden ejecutar varios programas simultáneamente, indicando que su ejecución se va a producir en segundo plano (*background*), en términos opuestos a primer plano (o *foreground*), donde se tiene un control exclusivo de pantalla. Otra utilización consiste en permitir trabajos largos en segundo plano cuando interactuamos el *shell* con otros programas en primer plano. Por ejemplo, ls ›dir.txt & se ejecutará en *background* permitiendo al usuario continuar interactuando con el *shell* mientras el comando se va ejecutando.

El *shell* por defecto es el que se indica en el último campo del fichero /etc/passwd para el usuario y se puede averiguar desde la línea de comando por el valor de la variable de entorno con:

```
echo $SHELL
```

Algunas consideraciones comunes a todos los *shells*:

- Todos permiten la escritura de *shell scripts*, que son luego interpretados ejecutándolos bien por el nombre (si el fichero tiene permiso de ejecución) o bien pasándolo como parámetro al comando del *shell*.

- Los usuarios del sistema tienen un *shell* por defecto asociado a ellos. Esta información se proporciona al crear las cuentas de los usuarios. El administrador asigna un *shell* a cada usuario, o bien si no se asigna el *shell* por defecto (bash en GNU/Linux). Esta información se guarda en el fichero de */etc/passwd*, y puede cambiarse con la orden **chsh**, esta misma orden con opción -l nos lista los *shells* disponibles en el sistema (o ver también */etc/shells*).

- Cada *shell* es en realidad un comando ejecutable, normalmente presente en los directorios /bin en GNU/Linux (o /usr/bin).

- Se pueden escribir *shell scripts* en cualquiera de ellos, pero ajustándose a la sintaxis de cada uno, que es normalmente diferente (a veces hay sólo pequeñas diferencias). La sintaxis de las construcciones, así como los comandos internos, están documentados en la página *man* de cada *shell* (*man bash*, por ejemplo).

- Cada *shell* tiene algunos ficheros de arranque asociados (ficheros de inicialización), cada usuario puede adaptarlos a sus necesidades, incluyendo código, variables, caminos (*path*)...

- La potencia en la programación está en combinar la sintaxis de cada *shell* (de sus construcciones), con los comandos internos de cada *shell*, y una serie de comandos UNIX muy utilizados en los *scripts*, como por ejemplo los *cut, sort, cat, more, echo, grep, wc, awk, sed, mv, ls, cp...*

- Si como usuarios estamos utilizando un *shell* determinado, nada impide arrancar una copia nueva de *shell* (lo llamamos *subshell*), ya sea el mismo u otro diferente. Sencillamente, lo invocamos por el nombre del ejecutable, ya sea el **sh, bash, csh o ksh**. También cuando ejecutamos un *shell script* se lanza un *subshell* con el *shell* que corresponda para ejecutar el *script* pedido.

Dada la importancia de los *shell scripts* en las tareas de administración de un sistema *nix, en el capítulo siguiente se verán algunos detalles más sobre el *shell* y ejemplos de sintaxis y de programación.

4.3. El sistema de arranque

Ya hemos visto en el punto anterior la forma de configurar durante la instalación del arranque del sistema a través de un gestor como Lilo o Grub (recomendable). Como ya sabemos, a partir de la BIOS o EFI, el ordenador lee el MBR del primer disco máster y ejecuta el gestor de arranque (si no se encuen-

tra este programa, se inspecciona el sector de arranque de la partición activa del disco) aunque recomendamos instalar grub en el MBR, que es el primer lugar que se inspecciona.

Grub nos permite múltiples configuraciones, permite tener un pequeño intérprete de órdenes al arrancar el ordenador, y acceder a los archivos de las particiones del disco sin cargar ningún sistema operativo, etc. En este subapartado sólo veremos algunas configuraciones básicas, pero si se necesitan opciones avanzadas se puede consultar la documentación existente en: http://www.gnu.org/software/grub/manual/grub.html.

Grub se carga en dos fases y durante el procedimiento de instalación ya se cargan dos ficheros correspondientes a cada fase. Si queremos instalar el grub para que no nos muestre ningún menú para seleccionar el sistema operativo que queremos cargar, sólo tenemos que ejecutar la orden **grub** y desde el prompt de éste, ejecutar:

```
install (hd0, 0)/boot/grub/stage1 d (hd0) (hd0, 0)/boot/grub/stage2
```

Esta instrucción instala el Grub en el MBR del disco máster y, como podemos observar, la forma como se refieren los discos difieren a como se hace referencia en GNU/Linux (o en Lilo). En hdX la X, en lugar de a, b. ..., es 0, 1, ... y para las particiones también se empieza con el número 0 para la primera, por lo cual hda1 será en grub (hd0, 0). El parámetro **d (hd0)** indica que la primera fase del grub se instalará en el MBR del primer disco. La última opción especifica dónde se sitúa el fichero para la segunda fase de carga, que es ejecutada por la primera (otra forma de hacer la instalación del grub es con la orden **grub-install**).

El paso de parámetros al núcleo Linux en el momento del arranque es muy simple. Por ejemplo, pasando *single* o *1* iniciaría el sistema en el *runlevel 1*, con *root=/dev/hda3* especificaremos la raíz del sistema de ficheros, etc. Si bien se pueden introducir las órdenes interactivamente, durante el arranque es recomendable utilizar un archivo de configuración como (los comentarios empiezan por "#"):

```
# Archivo menu.lst en /boot/grub
# Especificación del sistema operativo que se carga por defecto
default 0
# Espera de 10 segundos antes de cargar el sistema por defecto
timeout 10
# Configuración de arranque para un sistema GNU/Linux
title Debian GNU/Linux
kernel (hd0, 0) /vmlinuz root=/dev/hda1
# Configuración de arranque para un sistema Windows
title Windows
```

```
root (hd0, 2)
makeactive
```

Para instalar Grub con este menú de arranque, deberíamos ejecutar la misma instrucción que anteriormente, pero añadiendo el parámetro
p (hd0, 0)/boot/grub/menu.lst

Normalmente es la forma habitual de trabajar, por lo cual solo deberemos entrar en */boot/grub* y modificar este archivo. El archivo de configuración acepta varios parámetros, además de los mencionados, como:

- **Colores:**

```
color cyan/blue white/blue
```

- **Passwd** (para evitar que se pueda editar el archivo durante el arranque):

```
password ['--md5'] passwd
```

- **Carga de un kernel específico con initramfs:**

```
title Debian GNU/Linux, kernel 2.6.26-2-686
root (hd0,0)
kernel /boot/vmlinuz-2.6.26-2-686 root=/dev/hda1 ro quiet
initrd /boot/initrd.img-2.6.26-2-686
```

- **Carga de un kernel específico con initramfs y en modo Single User:**

```
title Debian GNU/Linux, kernel2.6.26-2-686 (single-user mode)
root (hd0,0)
kernel /boot/vmlinuz-2.6.26-2-686 root=/dev/hda1 ro single
initrd /boot/initrd.img-2.6.26-2-686
```

La carga de un *kernel* con una imagen llamada *initrd* hace referencia a un disco RAM (generado con **mkinitramfs**), que se carga en el momento del arranque para acceder a diferentes módulos que se anexarán al *kernel*. Este tipo de configuración permite tener un *kernel* general y agregar los dispositivos en el momento de la carga (módulos) por lo cual podríamos utilizar el *kernel* en diferentes máquinas y con diferente hardware sólo cambiando la imagen de los módulos del initrd.

4.4. Acceso a particiones y dispositivos

Los sistemas tipo UNIX tratan todos los dispositivos del ordenador como si fueran archivos. Esto permite total flexibilidad, ya que se pueden aprovechar todos los mecanismos y las funciones que se utilizan con ficheros para los dispositivos. En el directorio */dev/* se tienen todos los dispositivos reconocidos por el sistema. Si el sistema no reconoce adecuadamente un dispositivo o queremos crear uno especial, se puede utilizar la orden **mknod**, pero se debe utilizar con cuidado ya que su mal uso podría dañar partes del sistema.

Para las unidades de almacenamiento, el sistema provee comandos como **mount** y **umount**, que sitúan (montan) o desmontan todo el sistema de archivos de un determinado dispositivo/unidad en un directorio existente del sistema.

La forma básica de utilizar el orden es **mount dispositivo directorio**, donde el dispositivo puede ser cualquiera del canal IDE o SCSI (*/dev/hdXX,/dev/sdXX*), la disquetera (*/dev/FDX*), memorias USB, etc., y directorio es la ubicación en que montaremos la estructura de ficheros del dispositivo, y si el sistema de archivo no es el mismo con el cual estamos trabajando, se deberá agregar el parámetro *-t filesystem*, donde *fylesystem* es una sigla que se puede consultar en la página del manual del *mount*. Es recomendable que el directorio en el que montamos estos dispositivos esté vacío, ya que cuando se utiliza como punto de montaje no se puede acceder a él. Para desmontar uno de estos dispositivos, podemos utilizar **umount directorio**, donde el directorio debe ser el punto de montaje utilizado. Si montamos dispositivos movibles como CD/USB, es importante no sacar el dispositivo del soporte, ya que antes debemos avisar al sistema para que actualice la caché del sistema de ficheros del dispositivo (o actualizar las tablas internas en el caso que el dispositivo sólo sea de lectura). Igualmente, tampoco podemos desmontar el dispositivo si algún usuario o aplicación está utilizando alguno de sus archivos o directorios (al intentarlo, el sistema daría un mensaje de error).

Por defecto, para poder montar/desmontar sistemas de archivo, se necesitan privilegios de superusuario, pero se pueden ceder estos permisos a usuarios "administradores" añadiendo una entrada en el fichero *sudoers*, de manera que con la orden *sudo mount ...* el usuario habilitado podrá hacer tareas permitidas solo al root (consultar el manual de **mount** para otras opciones en el trabajo con dispositivos/particiones, como por ejemplo rw, suid, dev, exec, auto, nouser, async, que son las consideradas por defecto).

Todo el montaje de particiones y dispositivos se puede hacer durante la inicialización del sistema operativo a través de entradas en el archivo */etc/fstab*. Un ejemplo típico de este archivo es:

```
# /etc/fstab: static file system information.
```

```
#
# <file system>  <mount point>  <type>       <options>          <dump>  <pass>
  proc           /proc          proc         defaults           0       0
  /dev/hda1      /              ext3         errors=remount-ro  0       1
  /dev/hda5      none           swap         sw                 0       0
  /dev/hdc       /media/cdrom0  udf,iso9660  user,noauto        0       0
```

Cada línea es un *mount* y los parámetros son (en orden) dispositivo a montar, donde se monta, tipo de *filesystem* (auto para que lo detecte automáticamente), opciones de montado, especificación si queremos hacer copias de seguridad (dump) y el último campo sirve para indicar el orden de montaje (0, indica que el orden no es importante). La raíz del sistema de archivos es lo primero que se debe montar, con lo que en este campo debería haber un 1. Una entrada que siempre veremos en este fichero y que nos puede sorprender es el directorio /proc/, que tiene un significado especial. Realmente, lo que hay en este directorio no son ficheros, sino el valor de muchas de las variables que utiliza el núcleo del sistema y que permitirá ajustar parámetros del kernel como si se tratara de un archivo.

Es interesante consultar la orden **autofs**, que permite montar automáticamente un sistema de archivo cuando se detecta la inserción de un dispositivo en el sistema.

4.5. Configuración de dispositivos

Es importante, antes de intentar configurar algún dispositivo, buscar información al respecto e incluso antes de comprar uno, asegurarse de que dispone de drivers compatibles con la versión que pretendemos trabajar.

1) Teclado

Si comenzamos por el teclado, es importante que funcione de acuerdo al mapa de caracteres configurado en el proceso de instalación. Este mapa de caracteres se encuentra en */etc/console/boottime.kmap.gz* y si cambiamos de teclado, sólo se debe cambiar este fichero por el adecuado, que está en /usr/share/keymaps/ ordenado por arquitecturas de ordenadores y países (de estos archivos se puede cambiar alguna de sus entradas para adaptarlas a nuestras necesidades). Se puede ver el número que corresponde a cada tecla ejecutando como root la orden **showkey**. Si no se desea reiniciar el sistema al cambiar el archivo de configuración de teclado, podemos utilizar la orden **loadkeys**, **dumpkeys** nos muestra las que están configuradas y **consolechars** permite cargar la fuente que se desee para el terminal. Otro aspecto relacionado con el teclado es el tema de los acentos, las diéresis, que se pueden configurar a partir del fichero */etc/inputrc* (todas las directivas posibles de este fichero las tenemos especifica-

das en el manual de **readline**). La que puede ser más útil es la de **convert-meta**, que al desactivarla (*set convert-meta off*) nos permite utilizar los acentos y las diéresis.

Finalmente, otra configuración importante (indirectamente relacionada con el teclado) es la de *locales*, donde se puede configurar la zona geográfica en la que estamos para poder utilizar teclas especiales del teclado, ver las fechas en el formato correcto, etc. Esta configuración es utilizada por muchas de las librerías del sistema, de manera que en muchas órdenes y aplicaciones del mismo se utilizará su configuración para adaptar algunas funciones del entorno local. Su configuración se encuentra en */etc/locale.gen* y se pueden utilizar las órdenes **locale-gen** y **locale**, verlas o actualizarlas.

Nota

Otra manera de reconfigurar el keymap y las locales en Debian es *apt-reconfigure console-data apt-reconfigure locales* respectivamente.

2) Tarjeta de red (Ethernet)

Para configurar una nueva tarjeta de red (tipo Ethernet) se debe, en primer lugar, añadir el módulo necesario para que se reconozca adecuadamente. Si bien no es necesario para algunas tarjetas, debemos asegurarnos (antes de comprar la tarjeta) de que existe el driver o módulo necesario para ella.

Con la orden **discover** podemos saber qué tipo de hardware tenemos y encontrar el módulo correspondiente. Si se quiere dejar configurado para que se cargue, se deberá incluir en */etc/modules* (o también se puede utilizar **modprobe** o **insmode)**. Para configurarla en el */etc/network/interfaces*, se podrá especificar (en Debian) la configuración de las interfaces del sistema. Una interfaz es un dispositivo (real o lógico) relacionado con la red a partir del cual el sistema podrá comunicarse. Un ejemplo podría ser:

```
# Interfaz de loopback
auto lo
iface lo inet loopback
# NIC auto eth0
iface eth0 inet static
      address 192.168.0.10
      netmask 2 55.255.255.0
   network 192.168.0.0 # opcional
   broadcast 192.168.0.255 # opcional
   gateway 192.168.0.1 # opcional
```

La primera entrada de este archivo es la interfaz de *loopback*. Esta interfaz no se corresponde con ninguna tarjeta ni dispositivo real del ordenador y permite utilizar los mecanismos de comunicación de forma interna. La directiva auto, delante del dispositivo, indica que se puede montar automáticamente cuando el sistema arranca. La directiva de iface especifica el tipo de tarjeta y protocolo que se utilizará con ella mediante la sintaxis *iface dispositivo protocolo configuración*. Para las tarjetas Ethernet, el dispositivo será ethX, donde la X

será un número empezando por 0, que indica el número de tarjeta instalada en el ordenador. La familia del protocolo de comunicación utilizado con la tarjeta suele ser inet para el protocolo Ipv4 utilizado en Internet y inet6 para la nueva versión de Ipv6, y en el último campo se indica cómo se obtiene la configuración de red de la tarjeta (su dirección IP, la red donde está, el gateway a utilizar, etc.) con las siguientes posibilidades (básicas):

- **static**: deberemos proveer *address* (dirección IP de la interfaz), *netmask* (máscara de la dirección IP), *gateway* (dirección IP del *gateway* que utilizamos para esta interfaz).

- **dhcp**: para configurar de forma remota la IP de todos los ordenadores de una red local (*dynamic host configuration protocol*).

Para manejar la configuración de red tenemos diferentes órdenes, como **ifconfig** (visualiza y configura los dispositivos), **ifdown** y **ifup** (para apagar o encender la interfaz) o **route** (que nos muestra la tabla de encaminamiento).

3) Tarjeta inalámbrica (Wifi)

La red inalámbrica (wireless LAN o WiFi), y para configurarla deberemos añadir al núcleo los módulos necesarios o utilizar los que ya están compilados en el mismo núcleo. Se sigue la misma norma de nombre de dispositivos que para las tarjetas Ethernet y en */etc/network/interfaces* (en Debian) deberemos añadir la configuración necesaria para que se asigne una IP como en el siguiente ejemplo (consideramos que es el segundo dispositivo, por lo cual será eth1):

```
# Wireless network
auto eth1
iface eth1 inet dhcp
wireless_essid miwifi
wireless_channel 6
wireless_mode managed
wireless_keymode open
wireless_key1 millavehexadecimal
wireless_key2 s: millaveascii
wireless_defaultkey 1
```

Donde los parámetros subsiguientes a *iface* los podremos encontrar en la página del manual de la orden **iwconfig**, que es la que se utiliza para configurar los dispositivo inalámbricos.

4) Tarjeta de sonido

Al igual que en los casos anteriores, la tarjeta de sonido también necesita el módulo del núcleo para poder funcionar correctamente. Con la aplicación **discover**, podemos descubrir qué módulo es el que se corresponde con nuestra tarjeta o con la orden **lspci | grep audio**. Para instalar el módulo, procederemos como con la tarjeta de red con **insmod** o **modprobe**, para configurarlo permanentemente en */etc/modules*. Si bien con el módulo correspondiente ya podremos utilizar la tarjeta de sonido adecuadamente, generalmente también se suele instalar la infraestructura de sonido ALSA (*advanced linux sound architecture*) incluida por defecto en la mayoría de las distribuciones.

5) Impresora

En GNU/Linux, la configuración de impresoras se puede hacer de diferentes maneras, si bien el **lpd** (*line printer daemon*) fue uno de los primeros programas de gestión de impresión, en la actualidad hay otros más fáciles de configurar y gestionar. Los más básicos son:

- **lpd**: uno de los primeros *daemons* de impresión de los sistemas tipo UNIX. Su configuración se debe hacer manualmente.

- **lpr**: la versión de BSD del *lpd*. Es muy recomendable utilizar algún tipo de filtro automático como *magicfilter* o *apsfilter* para configurarlas.

- **LPRng**: aplicaciones basadas en *lpr*, con la ventaja de que incorporan una herramienta de configuración denominada *lprngtool*, que permite configurarla de manera gráfica y sencilla.

- **gnulpr**: la versión de GNU del sistema de impresión *lpr*. También incorpora herramientas gráficas de configuración, gestión de los servicios, etc.

- **CUPS (recomendado)**: de *Common UNIX printing systems*, este conjunto de aplicaciones es compatible con las órdenes de *lpr* y también sirve para redes Windows.

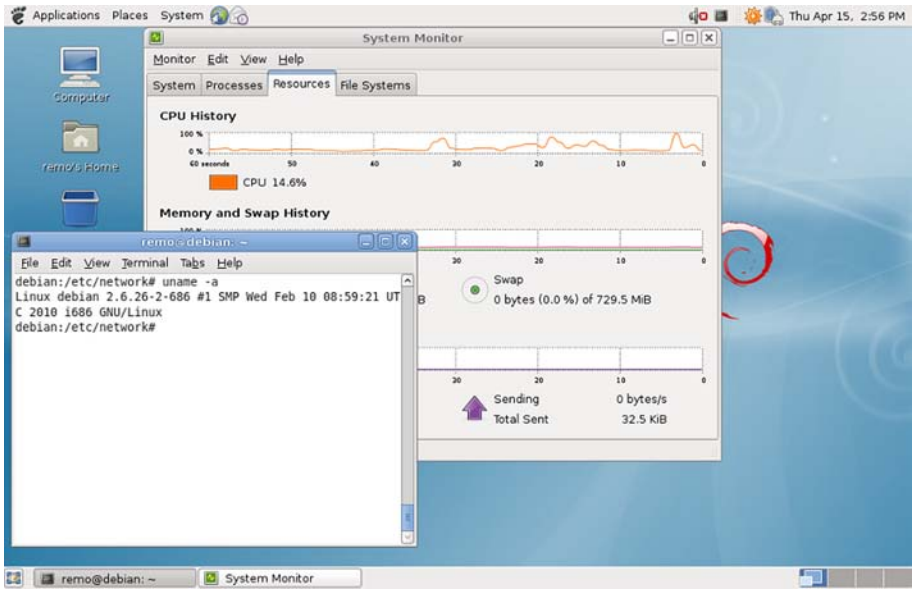
Normalmente, todas estas órdenes tienen sus propios métodos de configuración, pero utilizan el fichero */etc/printcap* para guardarla y utilizan un daemon (proceso que se ejecuta indefinidamente) para que el sistema de impresión sea operativo e incluso se pueda imprimir desde otros ordenadores.

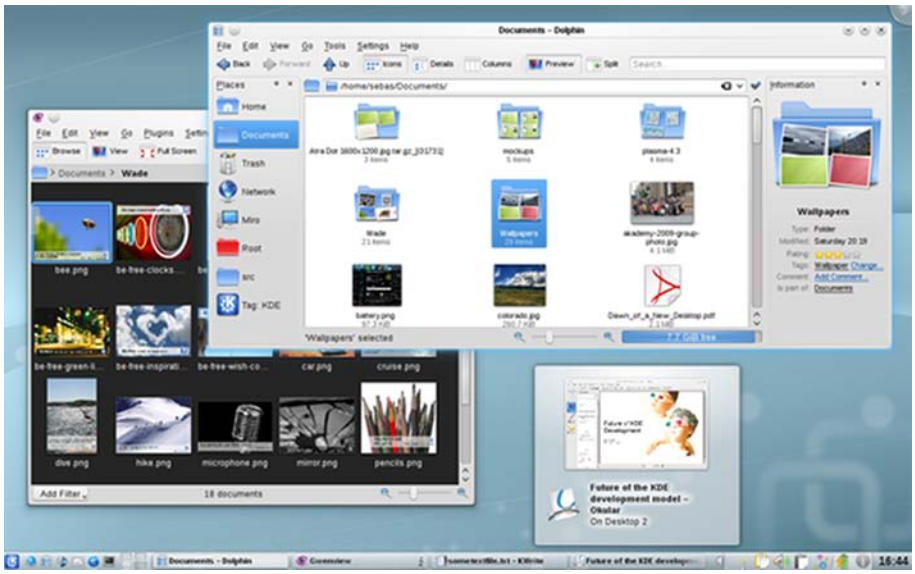
5. El entorno gráfico

Los sistemas *nix utilizan una arquitectura de entorno gráfico llamada X-Window diseñada en la década de los ochenta, que es independiente de la plataforma y sirve para cualquier tipo de *nix. X.Org es una implementación de código abierto del sistema X-Window (que surge como bifurcación de proyecto Xfree86) y funciona en modo cliente/servidor de manera que no podemos conectar gráficamente a un servidor remoto o ejecutar sobre un ordenador local aplicaciones gráficas de un ordenador remoto. Hoy en día se trabaja conjuntamente con una infraestructura de DRI (*direct rendering infrastructure*), que permite aprovechar los chips de procesamiento de las tarjetas para ahorrar trabajo de visualización al cliente X-Window.

El sistema X-Window (basado en una librería llamada *xlibs*) proporciona los métodos para crear las interfaces gráficas de usuarios (GUI), pero no implementa ninguna sino que éstas son proporcionadas por *toolkits*, que son bibliotecas generalmente implementadas con *xlibs* y que proporcionan un GUI particular. El *window manager* es un servidor especial de X-Window, que se encarga de gestionar todas las ventanas, los escritorios, las pantallas virtuales, etc. y obviamente, todas las aplicaciones pueden funcionar con cualquier *window manager*, ya que éste sólo se encarga de gestionar la ventana donde está ubicado el programa y hay decenas de ellos (Wmaker, sawmill, olvwm, etc.) siendo el usuario quien puede elegir el que más le agrade.

Otro tipo de software muy relacionado con X-Window es el que se encarga de proporcionar un entorno integrado para las aplicaciones, el escritorio, las herramientas de administración del sistema, etc. Los más populares en la actualidad son KDE (*the K desktop environment*) y GNOME (*GNU, object model environment*). Ambos proporcionan un toolkit particular, un entorno de escritorio con muchas funcionalidades y configuraciones diferentes y una lista de aplicaciones integradas, que cada vez va creciendo más, y son los más distribuidos en todas las distribuciones GNU/Linux. En las siguientes figuras podemos ver en primer lugar el aspecto de KDE y en segundo GNOME (se pueden ver diferentes WM y entornos en http://www.xwinman.org/):





En la actualidad, la mayoría de las tarjetas gráficas del mercado están soportadas y muchos fabricantes ya dan soporte para GNU/Linux proporcionando su propios *drivers*.

Para instalar Xorg en nuestro ordenador, es necesario bajar los paquetes que contienen las herramientas básicas y el software para el cliente y el servidor. Generalmente, estos paquetes se suelen denominar xorg, xserver-xorg, etc. y llevan implícitas diversas dependencias de fuentes y algunas utilidades básicas para el manejo de X-Window (se puede utilizar en Debian **apt-cache search Xorg** para ver los servidores disponibles y **apt-get install <Xorg.server>** para instalar uno en particular). Una vez instalados estos paquetes, debemos configurar adecuadamente los dispositivos de los que disponemos para poder arrancar correctamente el cliente y el servidor X-Window. En Debian podemos uti-

lizar la orden **dexconf**, que utilizará la base de datos **debconf** para generar el archivo de configuración del entorno gráfico (ubicado en */etc/X11/xorg.conf* normalmente).

Para probar la configuración podemos ejecutar **startx** y una vez que tenemos la pantalla en modo gráfico, podemos cambiar su resolución con las teclas *CtrlAlt+ CtrlAlt-* o volver a los terminales texto con *CtrlAltF1-5* y con *CtrlAltF7* volveríamos a la gráfica. Con *CrtlAltBackspace* terminaríamos la sesión del WM y volveremos al terminal texto.

Actividades

1. Analizad las siguientes órdenes y sus principales parámetros:

passwd, pwd, cd, ls, chmod, chown, chgrp, vi, man, cat, cp, date, df, du, find, mail, mkdir, more, page, mv, rm, rmdir, who, w, ps, cal, clear, zip, gzip, compress, uncompress, dc,diff, env, expr, exit, file, grep, kill, ln, lp, mesg, nice, nohup, quota, sed, awk, ftp,telnet, sleep, sort, su, sudo, tail, tar, tee, pstree, test, users, wc, write, cut, head.

2. Instalad Virtual Box sobre un sistema operativo huésped que se disponga y cargad una imagen ya configurada verificando que el sistema funcione (incluyendo sistema gráfico, red, acceso a disco del huésped, etc.).

3. Repetid el punto anterior instalando el sistema desde los CD/DVD enviados por la UOC.

Programación de comandos combinados (*shell scripts*)

Remo Suppi Boldrito

PID_00167541

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En este módulo veremos la importancia fundamental que tiene el intérprete de órdenes o comandos (*shell*) y sobre todo analizaremos con cierto detalle las posibilidades de estos para ejecutar ficheros de órdenes secuenciales y escritos en texto plano (ASCII) que serán interpretados por el shell. Estos ficheros, llamados *shell scripts*, son la herramienta fundamental de cualquier usuario avanzado e imprescindible para un administrador de sistemas *nix.

1. Introducción: el shell

El *shell* es una pieza de software que proporciona una interfaz para los usuarios en un sistema operativo y que provee acceso a los servicios del núcleo. Su nombre proviene de la envoltura externa de algunos moluscos, ya que es la parte externa que protege al núcleo.

Los *shells* se dividen en dos categorías: línea de comandos y gráficos, ya sea si la interacción se realiza mediante una línea de comandos (CLI) o en forma gráfica a través de una GUI. En cualquier categoría el objetivo principal del *shell* es invocar o "lanzar" otro programa, sin embargo, suelen tener capacidades adicionales, tales como ver el contenido de los directorios, interpretar órdenes condicionales, trabajar con variables internas, gestionar interrupciones, redirigir entrada/salida, etc.

Si bien un *shell* gráfico es agradable para trabajar y permite a usuarios sin muchos conocimientos desempeñarse con cierta facilidad, los usuarios avanzados prefieren los de modo texto ya que permiten una forma más rápida y eficiente de trabajar. Todo es relativo, ya que en un servidor es probable que un usuario administrador no utilice ni siquiera interfaz gráfica, mientras que para un usuario de edición de vídeo nunca trabajará en modo texto. El principal atractivo de los sistema *nix es que también en modo gráfico se puede abrir un terminal y trabajar en modo texto como si estuviera en un *shell* texto, o cambiar interactivamente del modo gráfico y trabajar en modo texto (hasta con 5 terminales diferentes) y luego regresar al modo gráfico con una simple combinación de teclas. En este apartado nos dedicaremos a *shell* en modo texto y las características avanzadas en la programación de *shell scripts*.

Entre los *shells* más populares (o históricos) en los sistemas *nix tenemos:

- Bourne shell (sh).
- Almquist shell (ash) o su versión en Debian (dash).
- Bourne-Again shell (bash).
- Korn shell (ksh).
- Z shell (zsh).
- C shell (csh) o la versión Tenex C shell (tcsh).

El Bourne shell ha sido el estándar *de facto* en los sistemas *nix, ya que fue distribuido por Unix Version 7 en 1977 y Bourne-Again (Bash) es una versión mejorada del primero escrita por el proyecto GNU bajo licencia GPL, la cual se ha transformado en el estándar de los sistemas GNU/Linux. Bash será el intérprete que analizaremos, ya que, además de ser el estándar, es muy potente,

posee características avanzadas, recoge las innovaciones planteadas en otros *shells* y permite ejecutar sin modificaciones (generalmente) *scripts* realizados para cualquier *shell* con sintaxis Bourne compatible. La orden **cat /etc/shells** nos proporciona los *shells* conocidos por el sistema Linux, independientemente de si están instalados o no. El *shell* por defecto para un usuario se obtendrá del último campo de la línea correspondiente al usuario del fichero */etc/passwd* y cambiar de *shell* simplemente significa ejecutar el nombre del *shell* sobre un terminal activo, por ejemplo:

```
RS@DebSyS:~$ echo $SHELL
/bin/bash
RS@DebSyS:~$ tcsh
DebSyS:~>
```

Una parte importante es lo que se refiere a los modos de ejecución de un *shell*. Se llama *login* interactivo cuando proviene de la ejecución de un *login*, lo cual implicará que se ejecutarán los archivos */etc/profile, ~/.bash_profile, ~/.bash_login o ~/.profile* (la primera de las dos que exista y se pueda leer) y *~/.bash_logout* cuando terminemos la sesión. Cuando es de *no-login* interactivo (se ejecuta un terminal nuevo) se ejecutará *~/.bashrc*, dado que un bash interactivo tiene un conjunto de opciones habilitadas a si no lo es (consultar el manual). Para saber si el shell es interactivo, ejecutar **echo $-** y nos deberá responder **himBH** donde la ***i*** indica que sí lo es.

Existen un conjunto de comandos internos[1] a *shell*, es decir, integrados con el código de *shell*, que para Bourne son:

(1)Consultar el manual para una descripción completa.

:, **.**, **break**, **cd**, **continue**, **eval**, **exec**, **exit**, **export**, **getopts**, **hash**, **pwd**, **readonly**, **return**, **set**, **shift**, **test**, **[**, **times**, **trap**, **umask**, **unset**.

Y además Bash incluye:

alias, **bind**, **builtin**, **command**, **declare**, **echo**, **enable**, **help**, **let**, **local**, **logout**, **printf**, **read**, **shopt**, **type**, **typeset**, **ulimit**, **unalias**.

Cuando Bash ejecuta un *script shell*, crea un proceso hijo que ejecuta otro Bash, el cual lee las líneas del archivo (una línea por vez), las interpreta y ejecuta como si vinieran de teclado. El proceso Bash padre espera mientras el Bash hijo ejecuta el *script* hasta el final que el control vuelve al proceso padre, el cual vuelve a poner el *prompt* nuevamente. Un comando de *shell* será algo tan simple como **touch archivo1 archivo2 archivo3** consiste en el propio comando seguido de argumentos, separados por espacios considerando *archivo1* el primer argumento y así sucesivamente.

1.1. Redirecciones y pipes

Existen 3 descriptores de ficheros: *stdin*, *stdout* y *stderr* (la abreviatura std significa estándar) y en la mayoría de los *shells* (incluso en Bash) se puede redirigir *stdout* y *stderr* (juntas o separadas) a un fichero, *stdout* a *stderr* y viceversa, donde el número 0 representa a *stdin*, 1 a *stdout*, y 2 a *stderr*.

Por ejemplo, enviar *stdout* del comando *ls* a un fichero será: **ls -l > dir.txt**, donde se creará un fichero llamado 'dir.txt', que contendrá lo que se vería en la pantalla si se ejecutara **ls -l**.

Para enviar la salida *stderr* de un programa a un fichero, haremos **grep xx yy 2> error.txt**. Para enviar la *stdout* a la *stderr*, haremos **grep xx yy 1>&2** y a la inversa simplemente intercambiando el 1 por 2, es decir, **grep xx yy 2>&1**.

Si queremos que la ejecución de un comando no genere actividad por pantalla, lo que se denomina ejecución silenciosa, solamente debemos redirigir todas sus salidas a */dev/null*, por ejemplo pensando en un comando del *cron* que queremos que borre todos los archivos acabados en .mov del sistema: **rm -f $(find / -name "*.mov") &> /dev/null** pero se debe ir con cuidado y estar muy seguro, ya que no tendremos ninguna salida por pantalla.

Los *pipes* permiten utilizar en forma simple tanto la salida de una orden como la entrada de otra, por ejemplo, **ls -l | sed -e "s/[aeio]/u/g"**, donde se ejecuta el comando ls y su salida, en vez de imprimirse en la pantalla, se envía (por un tubo o *pipe*) al programa **sed**, que imprime su salida correspondiente. Por ejemplo, para buscar en el fichero */etc/passwd* todas las líneas que acaben con false podríamos hacer **cat /etc/passwd | grep false$**, donde se ejecuta el **cat**, y su salida se pasa al **grep** donde el $ al final de la palabra le está indicando al **grep** que es final de línea.

1.2. Aspectos generales

Si la entrada no es un comentario, es decir (dejando de lado los espacios en blanco y tabulador) la cadena no comienza por #, el shell lee y lo divide en palabras y operadores, que se convierten en comandos, operadores y otras construcciones. A partir de este momento, se realizan las expansiones y sustituciones, las redirecciones, y finalmente, la ejecución de los comandos.

En Bash se podrán tener funciones, que son una agrupación de comandos que se pueden invocar posteriormente. Y cuando se invoca el nombre de la función de shell (se usa como un nombre de comando simple), la lista de comandos

relacionados con el nombre de la función se ejecutará teniendo en cuenta que las funciones se ejecutan en el contexto del *shell* en curso, es decir, no se crea ningún proceso nuevo para ello.

Un parámetro es una entidad que almacena valores, que puede ser un nombre, un número o un valor especial. Una variable es un parámetro que almacena un nombre y tiene atributos (1 o más o ninguno) y se crean con la sentencia **declare** y se remueven con **unset** y si no se les da valor, se les asigna la cadena nula.

Por último, existen un conjunto de expansiones que realiza el shell que se lleva a cabo en cada línea y que pueden ser enumeradas como: de tilde/comillas, parámetros y variables, substitución de comandos, de aritmética, de separación de palabras y de nombres de archivos.

2. Elementos básicos de un shell script

2.1. ¿Qué es un *shell script*?

Un *shell script* es simplemente un archivo (mysystem.sh para nosotros) que tiene el siguiente contenido (hemos numerado las líneas para referirnos a ellas con el comando el cat -n pero no forman parte del archivo):

```
RS@debian:~$ cat -n mysys.sh
1 #!/bin/bash
2 clear; echo "Información dada por el shell script mysys.sh. "
3 echo "Hola, $USER"
4 echo
5 echo "La fecha es `date`, y esta semana `date +"%V"`."
6 echo
7 echo "Usuarios conectados:"
8 w | cut -d " " -f 1 - | grep -v USER | sort -u
9 echo
10 echo "El sistema es `uname -s` y el procesador es `uname -m`."
11 echo
12 echo "El sistema está encendido desde hace:"
13 uptime
14 echo
15 echo "¡Esto es todo amigos!"
```

Para ejecutarlo podemos hacer **bash mysys.sh** o bien cambiarle los atributos para hacerlo ejecutable y ejecutarlo como una orden (al final se muestra la salida después de la ejecución):

```
RS@debian:~$ chmod 744 mysys.sh

RS@debian:~$ ./mysys.sh

Información dada por el shell script mysys.sh.
Hola, RS

La fecha es Sat Apr 17 07:04:36 EDT 2010, y esta semana 15.

Usuarios conectados:

RS
```

```
El sistema es Linux y el procesador es i686.

El sistema está encendido desde hace:
07:04:36 up 1:45, 2 users, load average: 0.29, 0.10, 0.03

¡Esto es todo amigos!
```

El script comienza con "#!", que es una línea especial para indicarle con qué shell se debe interpretar este script. La línea 2 es un ejemplo de dos comandos (uno borra la pantalla y otro imprime lo que está a la derecha) en la misma línea, por lo que deben estar separados por ";". El mensaje se imprime con la sentencia ***echo*** (comando interno) pero también se podría haber utilizado la sentencia ***printf*** (también comando interno) para una salida con formato. En cuanto a los aspectos más interesantes del resto: la línea 3 muestra el valor de una variable, la 5 forma un string con la salida de un comando (reemplazo de valores), la 8 ejecuta la secuencia de 4 comandos con parámetros encadenados por pipes "|" y la 10 (similar a la 5) también reemplaza valores de comandos para formar una cadena de caracteres antes de imprimirse por pantalla.

#!/bin/bash

Así evitamos que si el usuario tiene otro shell, su ejecución dé errores de sintaxis, es decir, garantizamos que este script siempre se ejecutará con bash.

Para depurar un shell script podemos ejecutar el script con **bash -x mysys.sh** o incluir en la primera línea el -x: **#!/bin/bash -x**.

También se puede depurar una sección de código incluyendo:

```
set -x # activa debugging desde aquí
código a depurar
set +x # para el debugging
```

2.2. Variables y *arrays*

Se pueden usar variables pero no existen tipos de datos. Una variable de bash puede contener un número, un carácter o una cadena de caracteres; no se necesita declarar una variable, ya que se creará con sólo asignarle un valor. También se puede declarar con *declare* y después asignarle un valor:

```
#!/bin/bash
a="Hola UOC"
echo $a
declare b
echo $b
b="Cómo están Uds.?"
echo $B
```

Como se puede ver, se crea una variable *a* y se le asigna un valor (que para delimitarlo se deben poner entre " ") y se recupera el VALOR de esta variable poniéndole un '$' al principio, y si no se antepone el $, sólo imprimirá el nombre de la variable no su valor. Por ejemplo, para hacer un script que haga una copia (*backup*) de un directorio, incluyendo la fecha y hora en el momento de hacerlo en el nombre del archivo (intentad hacer esto tan fácil sobre un sistema W y solo con el SO y terminaréis frustrados):

```
#!/bin/bash
OF=/home/$USER-$(date +%d%m%Y).tgz
tar -cZf $OF /home/$USER
```

Este *script* introduce algo nuevo, ya que estamos creando una variable y asignando un valor (resultado de la ejecución de un comando en el momento de la ejecución). Fijaos en la expresión $(date +%d%m%Y) que se pone entre () para capturar su valor. USER es una variable de entorno y solamente se reemplazará por su valor. Si quisiéramos agregar (concatenar) a la variable USER, por ejemplo, una string más, deberíamos delimitar la variable con {}. Por ejemplo:

```
RS@debian:~$OF=/home/${USER}uppi-$(date +%d%m%Y).tgz
RS@debian:~$ echo $OF
/home/RSuppi-17042010.tgz
```

Así, el nombre del fichero será distinto cada día. Es interesante ver el reemplazo de comandos que hace el bash con los (), por ejemplo, echo $(ls).

Las variable locales pueden declararse anteponiendo *local* y eso delimita su ámbito.

```
#!/bin/bash
HOLA=Hola
function uni {
local HOLA=UOC
echo -n $HOLA
}
echo -n $HOLA
uni
echo $HOLA
```

La salida será *HolaUOCHola*, donde hemos definido una función con una variable local que recupera su valor cuando se termina de ejecutar la función.

Las variables las podemos declarar como **ARRAY[INDEXNR]=valor**, donde *INDEXNR* es un número positivo (comenzando desde 0) y también podemos declarar un array como **declare -a ARRAYNAME** y se pueden hacer asignaciones múltiples con: **ARRAY=(value1 value2 ... valueN)**

```
RS@debian:~$ array=( 1 2 3)
RS@debian:~$ echo $array
1
RS@debian:~$ echo ${array[*]}
1 2 3
RS@debian:~$ echo ${array[2]}
3
RS@debian:~$ array[3]=4
RS@debian:~$ echo ${array[*]}
1 2 3 4
RS@debian:~$ echo ${array[3]}
4
RS@debian:~$ unset array[1]
RS@debian:~$ echo ${array[*]}
1 3 4
```

2.3. Estructuras condicionales

Las estructuras condicionales le permiten decidir si se realiza una acción o no; esta decisión se toma evaluando una expresión.

- La más básica es: **if** *expresión* **then** *sentencia* donde 'sentencia' sólo se ejecuta si 'expresión' se evalúa como verdadera. '2<1' es una expresión que se evalúa falsa, mientras que '2>1' se evalúa verdadera.
- Los condicionales tienen otras formas, como: **if** *expresión* **then** *sentencia1* **else** *sentencia2*. Aquí 'sentencia1' se ejecuta si 'expresión' es verdadera. De otra manera se ejecuta 'sentencia2'.
- Otra forma más de condicional es: **if** *expresión1* **then** *sentencia1* **else if** *expresión2* **then** *sentencia2* **else** *sentencia3*. En esta forma sólo se añade "else if 'expresión2' then 'sentencia2'", que hace que sentencia2 se ejecute si expresión2 se evalúa verdadera. La sintaxis es:

```
if [expresión];
then
código si 'expresión' es verdadera.
fi
```

Un ejemplo en el que el código que se ejecutará si la expresión entre corchetes es verdadera se encuentra entre la palabra *then* y la palabra *fi*, que indica el final del código ejecutado condicionalmente:

```
#!/bin/bash
if [ "pirulo" = "pirulo" ]; then
echo expresión evaluada como verdadera
fi
```

Otro ejemplo con variables y comparación por igualdad y por diferencia (= o !=):

```
#!/bin/bash
T1="pirulo"
T2="pirulon"
if [ "$T1" = "$T2" ]; then
echo expresión evaluada como verdadera
else
echo expresión evaluada como falsa
fi
if [ "$T1" != "$T2" ]; then
echo expresión evaluada como verdadera
else
echo expresión evaluada como falsa
fi
```

Un ejemplo de expresión de comprobación si existe un fichero (vigilar con los espacios después de [y antes de]):

```
FILE=~/.basrc
if [ -f $FILE ]; then
echo el fichero $FILE existe
else
echo fichero no encontrado
fi
if [ 'test -f $FILE' ]
echo Otra forma de expresión para saber si existe o no
fi
```

Existen versiones cortas del **if**, como por ejemplo:

```
[ -z "${COLUMNS:-}" ] && COLUMNS=80
```

Es la versión corta de:

```
if [ -z "${COLUMNS:-}" ]; then
COLUMNS=80
fi
```

2.4. Los bucles

El bucle **for** es distinto a los de otros lenguajes de programación, ya que permite iterar sobre una serie de 'palabras' contenidas dentro de una cadena. El bucle **while** ejecuta una sección de código si la expresión de control es verdadera, y sólo se para cuando es falsa (o se encuentra una interrupción explícita dentro del código en ejecución). El bucle **until** es casi idéntico al bucle **while**, excepto en que el código se ejecuta mientras la expresión de control se evalúe como falsa. Veamos unos ejemplos y observemos la sintaxis:

```
#!/bin/bash
for i in $( ls ); do
echo elemento: $i
done
```

En la segunda línea declaramos *i* como la variable que recibirá los diferentes valores contenidos en *$(ls)*, que serán los elementos del directorio obtenido en el momento de la ejecución. El bucle se repetirá (entre do y done) tantas veces como elementos tenga la variable y en cada iteración la variable *i* adquirirá un valor diferente de la lista. Otra sintaxis aceptada podrá ser:

```
#!/bin/bash
for i in `seq 1 10`;
do
echo $i
done
```

La sintaxis del **while** es:

```
#!/bin/bash
contador=0
while [ $contador -lt 10 ]; do
echo El contador es $contador
let contador=contador+1
done
```

Como podemos ver, además de **while** hemos introducido una comparación en la expresión por menor "-lt" y una operación numérica con let contador=contador+1. La sintaxis del **until** es equivalente:

```
#!/bin/bash
contador=20
until [ $contador -lt 10 ]; do
echo Contador-Until $contador
let contador-=1
done
```

En este caso vemos la equivalencia de la sentencia y además otra forma de hacer operaciones sobre las mismas variables con el "-=".

2.5. Funciones, select, case, argumentos y otras cuestiones

Las funciones son la forma más fácil de agrupar secciones de código que se podrán volver a utilizar. La sintaxis es *function nombre { mi_código }*, y para llamarlas sólo es necesario que estén dentro de mismo archivo y escribir su nombre.

```
#!/bin/bash
function salir {
exit
}
function hola {
echo Hola UOC!
}
hola
salir
echo Nunca llegará a esta línea
```

Como se puede ver tenemos dos funciones, *salir* y *hola*, que luego se ejecutarán (no es necesario declararlas en un orden específico) y como la función *salir* ejecuta un **exit** en shell nunca llegará a ejecutar el echo final. También podemos pasar argumentos a las funciones, que serán leídos por orden (de la misma forma como se leen los argumentos de un *script*) por $1, el primero, $2 el segundo y así sucesivamente:

```
#!/bin/bash
function salir {
exit
}
function hola-con-parametros {
```

```
    echo $1
    }
    hola-con-parametros Hola
    hola-con-parametros UOC
    salir
    echo Nunca llegará a esta línea
```

El **select** es para hacer opciones y leer desde teclado:

```
    #!/bin/bash
    OPCIONES="Salir Opción1"
    select opt in $OPCIONES; do
    if [ "$opt" = "Salir" ]; then
    echo Salir. Adiós.
    exit
    elif [ "$opt" = "Opción1" ]; then
    echo Hola UOC
    else
    clear
    echo opción no permitida
    fi
    done
```

El **select** permite hacer menús modo texto y es equivalente a la sentencia **for**, solo que itera sobre el valor de la variable indicada en el select. Otro aspecto importante es la lectura de argumentos, como por ejemplo:

```
    #!/bin/bash
    if [ -z "$1" ]; then
    echo uso: $0 directorio
    exit
    fi
    A=$1
    B="/home/$USER"
    C=backup-home-$(date +%d%m%Y).tgz
    tar -cZf $A$B $C
```

Si el número de argumentos es 0, escribe el nombre del comando ($0) con un mensaje de uso. El resto es similar a lo que hemos visto anteriormente excepto por el primer argumento ($1). Mirad con atención el uso de " y la sustitución de variables. El **case** es otra forma de selección, miremos este ejemplo que nos alerta en función del espacio de disco que tenemos disponible:

```
#!/bin/bash
```

```
space=`df -h | awk '{print $5}' | grep % \
| grep -v Use | sort -n | tail -1 | cut -d "%" -f1 -`

   case $space in
   [1-6]*)
   Message="todo ok."
   ;;
   [7-8]*)
   Message="Podría comenzar a borrar algo en $space %"
   ;;
   9[1-8])
   Message="Uff. Mejor un nuevo disco. Partición $space % a tope."
   ;;
   99)
   Message="Pánico! No tiene espacio en $space %!"
   ;;
   *)
   Message="NO tienen nada..."
   ;;
   esac

   echo $Message
```

Además de prestar atención a la sintaxis del **case**, podemos mirar cómo escribir un comando en dos líneas con "\", el filtrado secuencia con varios greps y | y un comando interesante (filtro) como el ***awk*** (el mejor comando de todos los tiempos).

En muchas ocasiones, puede querer solicitar al usuario alguna información, y existen varias maneras para hacer esto:

```
   #!/bin/bash
   echo Por favor, introduzca su nombre
   read nombre
   echo "Hola $nombre!"
```

Como variante, se pueden obtener múltiples valores con read:

```
   #!/bin/bash
   echo Por favor, introduzca su nombre y primer apellido
   read NO AP
   echo "Hola $AP, $NO!"
```

Si hacemos *echo 10 + 10* y esperabais ver 20 quedaréis desilusionados, la forma de evaluación directa será con **echo $((10+10))** o también **echo $[10+10]** y si necesitáis operaciones como fracciones u otras podéis utilizar **bc**, ya que lo anterior solo sirve para números enteros. Por ejemplo, *echo $[3/4]* dará 0 pero **echo 3/4|bc -l** sí que funcionará. También recordemos el uso del **let**, por ejemplo, **let a=75*2** asignará 150 a la variable a. Un ejemplo más:

```
RS@debian:~$ echo $date
20042011
RS@debian:~$ echo $((date++))
20042011
RS@debian:~$ echo $date
20042012
```

Es decir, como operadores podemos utilizar:

- VAR++ VAR-- variable post-incremento y post-decremento.
- ++VAR –VAR ídem anterior pero antes.
- + - operadores unarios.
- ! ~ negación lógica y negación en *strings*.
- ** exponente.
- / + - % operaciones.
- << >> desplazamiento izquierda y derecha.
- <= >= < > comparación.
- == != igualdad y diferencia.
- & ^ | operaciones and, or exclusivo y or.
- && || operaciones and y or lógicos.
- expr ? expr : expr evaluación condicional.
- = *= /= %= += -= <<= >>= &= ^= |= operaciones implícitas.
- , separador entre expresiones.

Para capturar la salida de un comando, es decir, el resultado de la ejecución, podemos utilizar el nombre del comando entre comilla hacia la izquierda (` `).

```
#!/bin/bash
A=`ls`
for b in $A ;
do
file $b
done
```

Para la comparación tenemos las siguientes opciones:

- *s1 = s2* verdadero si s1 coincide con s2.
- *s1 != s2* verdadero si s1 no coincide con s2.

- *s1 < s2* verdadero si s1 es alfabéticamente anterior a s2.
- *s1 > s2* verdadero si s1 es alfabéticamente posterior a s2.
- *-n s1* s1 no es nulo (contiene uno o más caracteres).
- *-z s1* s1 es nulo.

Por ejemplo, se debe ir con cuidado porque si S1 o S2 están vacíos, nos dará un error de interpretación (*parse*):

```
#!/bin/bash
S1='cadena'
S2='Cadena'
if [ $S1!=$S2 ];
then
echo "S1('$S1') no es igual a S2('$S2')"
fi
if [ $S1=$S1 ];
then
echo "S1('$S1') es igual a S1('$S1')"
fi
```

En cuanto a los operadores aritméticos y relacionales, podemos utilizar:

a) Operadores aritméticos:

- + (adición).
- - (sustracción).
- * (producto).
- / (división).
- % (módulo).

b) Operadores relacionales:

- -lt (<).
- -gt (>).
- -le (<=).
- -ge (>=).
- -eq (==).
- -ne (!=).

Hagamos un ejemplo de *script* que nos permitirá renombrar ficheros S1 con una serie de parámetros (prefijo o sufijos) de acuerdo a los argumentos. El modo será p [prefijo] ficheros... o si no, s [sufijo] ficheros.. o también r [patrón-antiguo] [patrón-nuevo] ficheros.. (y como siempre decimos, los scripts son mejorables):

```
#!/bin/bash -x
# renom: renombra múltiples ficheros de acuerdo con ciertas reglas
# Basado en original de F. Hudson Enero - 2000
# comprueba si deseo renombrar con prefijo y dejo solo los nombres de
# archivos
if [ $1 = p ]; then
   prefijo=$2 ; shift ; shift

   # si no hay entradas de archivos termino.
   if [ "$1" = '' ]; then
      echo "no se especificaron ficheros"
      exit 0
   fi
   # Interación para renombrar
   for fichero in $*
      do
      mv ${fichero} $prefijo$fichero
      done
   exit 0
fi
# comprueba si deseo renombrar con prefijo y dejo solo los nombres de
# archivos
if [ $1 = s ]; then
   sufijo=$2 ; shift ; shift
   if [ "$1" = '' ]; then
      echo "no se especificaron ficheros"
      exit 0
   fi
   for fichero in $*
      do
      mv ${fichero} $fichero$sufijo
      done
   exit 0
fi
# comprueba si es una sustitución de patrones
if [ $1 = r ]; then
shift
# se ha incluido esto como medida de seguridad
   if [ $# -lt 3 ] ; then
      echo "uso: renom r [expresión] [sustituto] ficheros... "
      exit 0
   fi
   VIEJO=$1 ; NUEVO=$2 ; shift ; shift
   for fichero in $*
      do
      nuevo=`echo ${fichero} | sed s/${VIEJO}/${NUEVO}/g`
      mv ${fichero} $nuevo
```

```
      done
   exit 0
fi
# si no le muestro la ayuda
echo "uso:"
echo " renom p [prefijo] ficheros.."
echo " renom s [sufijo] ficheros.."
echo " renom r [patrón-antiguo] [patrón-nuevo] ficheros.."
exit 0
```

Especial atención se debe prestar a las "", '', `` {}etc., por ejemplo:

```
RS@debian:~$ date=20042010
RS@debian:~$ echo $date
20042010
RS@debian:~$ echo \$date
$date
RS@debian:~$ echo '$date'
$date
RS@debian:~$ echo "$date"
20042010
RS@debian:~$ echo "`date`"
Sat Apr 17 14:35:03 EDT 2010
RS@debian:~$ echo "lo que se me ocurre es: \"Estoy cansado\""
lo que se me ocurre es: "Estoy cansado"
RS@debian:~$ echo "\"
>
(espera más entradas: hacer Crtl-C)
remo@debian:~$ echo "\\"
\
RS@debian:~$ echo grande{cit,much,poc,ningun}o
grandecito grandemucho grandepoco grandeninguno
```

Una combinación de entrada de teclado y operaciones/expresiones para calcular un año bisiesto:

```
#!/bin/bash
clear;
echo "Entre el año a verificar (4 digits), y después [ENTER]:"
read year
if (( ("$year" % 400) == "0" )) || (( ("$year" % 4 == "0") \
&& ("$year" % 100 !="0") )); then
   echo "$year es bisiesto."
else
   echo "$year este año NO es bisiesto."
```

```
fi
```

Un comando interesante combinado de **read** con delimitador en una expresión (está puesto todo en una línea y por eso la sintaxis está separada por ;). El resultado será "interesante" (nos mostrará todos los string dados por el **find** en un línea y sin /:

```
find "$PWD" -name "m*" | while read -d "/" file; do echo $file; done
```

Un aspecto interesante para trabajar con *string* (ver manual de bash: man bash) es ${VAR:*OFFSET*:*LENGTH*}.

```
RS@debian:~$ export str="unstring muy largo"
RS@debian:~$ echo $str
unstring muy largo
RS@debian:~$ echo ${str:4}
ring muy largo
RS@debian:~$ echo $str
unstring muy largo
RS@debian:~$ echo ${str:9:3}
muy
RS@debian:~$ echo ${str%largo}
unstring muy
```

La sentencia **trap** nos permitirá capturar una señal (por ejemplo, de teclado) dentro de un script.

```
#!/bin/bash
# Para finalizar hacer desde otro terminal kill -9 pid

trap "echo ' Ja!'" SIGINT SIGTERM
echo "El pid es $$"

while : # esto es lo mismo que "while true".
      do
      sleep 10 # El script no hace nada.
done
```

2.6. Filtros: Grep

El grep es un filtro de patrones muy útil y versátil, a continuación podemos ver algunos ejemplos:

```
```

```
RS@debian:~$ grep root /etc/passwd   busca patron root
root:x:0:0:root:/root:/bin/bash

RS@debian:~$ grep -n root /etc/passwd   además numera las líneas
1:root:x:0:0:root:/root:/bin/bash

RS@debian:~$ grep -v bash /etc/passwd | grep -v nologin

   lista los que no tienen el patrón y bash ni el patrón nologin
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
...

RS@debian:~$ grep -c false /etc/passwd   cuenta los que tienen false
7
RS@debian:~$ grep -i ps ~/.bash* | grep -v history
   busca patrón ps en todos los archivos que comienzan por .bash en el directorio home
   excluyendo los que el archivo contenga history
/home/RS/.bashrc:[ -z "$PS1" ] &amp;amp;&amp;amp; return
/home/RS/.bashrc:export HISTCONTROL=$HISTCONTROL${HISTCONTROL+,}ignoredups
/home/RS/.bashrc:# ... or force ignoredups and ignorespace
...

RS@debian:~$ grep ^root /etc/passwd   busca a inicio de línea
root:x:0:0:root:/root:/bin/bash

RS@debian:~$ grep false$ /etc/passwd   busca a final de línea
Debian-exim:x:101:105::/var/spool/exim4:/bin/false
statd:x:102:65534::/var/lib/nfs:/bin/false

RS@debian:~$ grep -w / /etc/fstab   busca /
/dev/hda1 / ext3 errors=remount-ro 0 1

RS@debian:~$ grep [yf] /etc/group   busca y o Y
sys:x:3:
tty:x:5:
....

RS@debian:~$ grep '\<c...h\>' /usr/share/dict/words   busca comenzando por c y terminando
                                        por h con un máximo de tres.
catch
cinch
cinch's
....
```

```
RS@debian:~$ grep '\<c.*h\>' /usr/share/dict/words   busca comenzando por c y terminando
                                          por h todas.
caddish
calabash
...
```

2.7. Filtros: Awk

Awk, o también la implementación de GNU gawk, es un comando que acepta un lenguaje de programación diseñado para procesar datos basados en texto, ya sean ficheros o flujos de datos, y ha sido la inspiración de Larry Wall para escribir Perl. Su sintaxis más común es **awk** '**programa**' **archivos** ... y donde programa puede ser: **patrón {acción} patrón {acción}...**, awk lee la entrada de archivos un renglón a la vez. Cada renglón se compara con cada patrón en orden; para cada patrón que concuerde con el renglón se efectúa la acción correspondiente. Un ejemplo pregunta por el primer campo si es *root* de cada línea del /etc/passwd y la imprime considerando como separador de campos el ":" con -F:, en el segundo ejemplo reemplaza el primer campo por *root* e imprime el resultado cambiado.

```
RS@debian:~$ awk -F: '$1=="root" {print}' /etc/passwd
root:x:0:0:root:/root:/bin/bash


RS@debian:~$ awk -F: '$1="root" {print}' /etc/passwd
root x 0 0 root /root /bin/bash
root x 1 1 daemon /usr/sbin /bin/sh
root x 2 2 bin /bin /bin/sh
root x 3 3 sys /dev /bin/sh
root x 4 65534 sync /bin /bin/sync
root x 5 60 games /usr/games /bin/sh
root x 6 12 man /var/cache/man /bin/sh
```

El awk divide automáticamente la entrada de líneas en campos, es decir, cadena de caracteres que no sean blancos separados por blancos o tabuladores, por ejemplo, el **who** tiene 5 campos y **awk** nos permitirá filtrar cada uno de estos campos.

```
RS@debian:~$ who
RS tty7 2010-04-17 05:49 (:0)
RS pts/0 2010-04-17 05:50 (:0.0)
RS pts/1 2010-04-17 16:46 (:0.0)
remo@debian:~$ who | awk '{print $4}'
05:49
05:50
```

```
    16:46
```

El awk llama a estos campos $1 $2 ... $NF donde NF es una variable igual al número de campos (en este caso NF=5). Por ejemplo:

```
ls -al | awk '{print NR, $0}   Agrega números de entradas (la variable NR cuenta el
                                    número de líneas, $0 es la línea entera.
awk '{printf "%4d %s\n", NR, $0}'  significa un número decimal (NR) un string ($0)
                                    y un new line
awk -F: '$2 == "" ' /etc/passwd   Dará los usuarios que en el archivo de passwd no
                                    tengan puesto la contraseña
```

El patrón puede escribirse de varias formas:

```
$2 == "" si el segundo campo es vacío
$2 ~ /^$/ si el segundo campo coincide con la cadena vacía
$2 !~ /./ si el segundo campo no concuerda con ningún carácter (! es negación)
length($2) == si la longitud del segundo campo es 0, length es una función interna del awk
~ indica coincidencia con una expresión
!~ significa los contrario (no coincidencia)
NF % 2 != 0 muestra el renglón solo si hay un número par de campos
awk 'lenght ($0) 32 {print "Línea", NR, "larga", substr($0,1,30)} ' /etc/passwd
   evalúa las líneas de /etc/passwd y genera un substring de esta
```

Existen dos patrones especiales BEGIN y END. Las acciones BEGIN se realizan antes que el primer renglón se haya leído; puede usarse para inicializar las variables, imprimir encabezados o posicionar separadores de un campo asignándoselos a la variable FS por ejemplo:

```
awk 'BEGIN {FS = ":"} $2 == "" ' /etc/passwd   igual que el ejemplo de antes
awk 'END { print NR } ' /etc/passwd   imprime el número de líneas procesadas al
            final de la lectura del último renglón.
```

El awk permite operaciones numéricas en forma de columnas, por ejemplo, para sumar todos los números de la primera columna:

```
{ s=s + 1} END { print s}   y para la suma y el promedio END { print s, s/NR}
```

Las variables se inicializan a cero al declararse, y se declaran al utilizarse, por lo que resulta muy simple (los operadores son los mismos que en C):

```
{ s+=1} END {print s} Por ejemplo, para contar renglones, palabras y caracteres:
```

```
{ nc += length($0) +1 nw +=NF } END {print NR, nw, nc}
```

Existen variables predefinidas en el awk:

- FILENAME nombre del archivo actual.
- FS carácter delimitador del campo.
- NF número de campos del registro de entrada.
- NR número del registro del entrada.
- OFMT formato de salida para números (%g default).
- OFS cadena separadora de campo en la salida (blanco por default).
- ORS cadena separadora de registro de salida (*new line* por default).
- RS cadena separador de registro de entrada (*new line* por default).

Operadores (ídem C):

- = += -= *= /= %= || && ! >> >>= << <<= == != ~ !~
- + - * / %
- ++ --

Funciones predefinidas en awk: cos(), exp(), getline(), index(), int(), length(), log(), sin(), split(), sprintf(), substr().

Además soporta dentro de acción sentencias de control con la sintaxis similar a C:

```
if (condición)
proposición1
else
proposición2

for (exp1;condición;exp2)

while (condición) {
proposición
expr2
}
continue   sigue, evalúa la condición nuevamente
break   rompe la condición
next   lee la siguiente línea de entrada
exit   salta a END
```

También el awk maneja arreglos, por ejemplo, para hacer un head de */etc/passwd*:

```
awk '{ line[NR] = $0} \
END { for (i=NR; i>2; i--) print line[i]} ' /etc/passwd
```

Otro ejemplo:

```
awk 'BEGIN { print "Usuario UID Shell\n------- --- -----" } $3 >= 500 { print $1, $3,
 $7 | "sort -r"}' FS=":" /etc/passwd
Usuario UID Shell
------- --- -----
RS 1001 /bin/bash
nobody 65534 /bin/sh
debian 1000 /bin/bash
RS@debian:~$ ls -al | awk '
BEGIN { print "File\t\t\tOwner" }
{ print $8, "\t\t\t", \
$3}
END { print "done"}
'
File Owner
. RS
.. root
.bash_history RS
.bash_logout RS
.bashrc RS
.config RS
...
```

2.8. Ejemplos complementarios

1) Un ejemplo completo para borrar los logs (borra */var/log/wtmp* y se queda con 50 líneas –o las que pase el usuario en línea de comando– de */var/log/messages*).

```
   #!/bin/bash
   #Variables
   LOG_DIR=/var/log
   ROOT_UID=0 # solo lo pueden ejecutar el root
   LINES=50 # Líneas por defecto.
   E_XCD=86 # NO me puedo cambiar de directorio
   E_NOTROOT=87 # Salida de No-root error.

   if [ "$UID" -ne "$ROOT_UID" ] # Soy root?
      then
      echo "Debe ser root. Lo siento."
```

```
   exit $E_NOTROOT
   fi

if [ -n "$1" ] # Número de líneas a preservar?
   then
   lines=$1
   else
   lines=$LINES # valor por defecto.
   fi

cd $LOG_DIR

if [ `pwd` != "$LOG_DIR" ]
# también puede ser if [ "$PWD" != "$LOG_DIR" ]
   then
   echo "No puedo ir a $LOG_DIR."
   exit $E_XCD
   fi #

# Otra forma de hacerlo sería :
# cd /var/log || {
# echo "No puedo !" >&2
# exit $E_XCD;
# }

tail -n $lines messages > mesg.temp # Salvo en temporal
mv mesg.temp messages # Muevo.

cat /dev/null > wtmp #Borro wtmp.
echo "Logs Borrados."
exit 0
# Un cero indica que todo Ok.
```

2) Utilización del expr.

```
#!/bin/bash
echo "Ariméticos"
echo
a=`expr 5 + 3`
echo "5 + 3 = $a"
a=`expr $a + 1`
echo
echo "a + 1 = $a"
echo "(incremento)"
a=`expr 5 % 3`
# módulo
```

```
echo
echo "5 mod 3 = $a"

# Lógicos
# 1 si true, 0 si false,
#+ opuesto a normal Bash convention.

echo "Lógicos"
x=24
y=25
b=`expr $x = $y` # por igual.
echo "b = $b" # 0 ( $x -ne $y )
a=3
b=`expr $a \> 10`
echo 'b=`expr $a \> 10`, por lo cual...'
echo "If a > 10, b = 0 (false)"
echo "b = $b" # 0 ( 3 ! -gt 10 )
b=`expr $a \< 10`
echo "If a < 10, b = 1 (true)"
echo "b = $b" # 1 ( 3 -lt 10 )
echo
# Cuidado con los operadores de escape \.
b=`expr $a \<= 3`
echo "If a <= 3, b = 1 (true)"
echo "b = $b" # 1 ( 3 -le 3 )
# Se utiliza un operador "\>=" operator (greater than or equal to).

echo "String"
echo
a=1234zipper43231
echo "String base \"$a\"."
b=`expr length $a`
echo "Long. de \"$a\" es $b."
# index: posición del primer caracter que satisface en el string
#
b=`expr index $a 23`
echo "Posición numérica del \"2\" en \"$a\" es \"$b\"."
# substr: extract substring, starting position &amp;amp; length specified
b=`expr substr $a 2 6`
echo "Substring de \"$a\", comenzando en 2,\
y de 6 chars es \"$b\"."
# Using Regular Expressions ...
b=`expr match "$a" '[0-9]*'` # contador numérico.
echo Número de digitos de \"$a\" es $b.
b=`expr match "$a" '\([0-9]*\)'` # cuidado los caracteres de escape
echo "Los dígitos de \"$a\" son \"$b\"."
```

```
    exit 0
```

Actividades

1. Cread un script interactivo que calcule el número de días entre dos fechas introducidas por el usuario en el siguiente formato día/mes/año. También deberéis calcular el número de horas si el usuario introduce hora de inicio y hora de final en el siguiente formato hh:mm. Si introduce solo una hora, se calculará hasta las 23:59 del día dado como final del período a calcular.

2. Cread un script que se haga una copia (recursiva) de los archivos en */etc*, de modo que un administrador del sistema pueda modificar los archivos de inicio sin temor.

3. Escribid un script llamado *homebackup* que automatice el **tar** con las opciones correctas y en el directorio */var/backups* para hacer una copia de seguridad del directorio principal del usuario con las siguientes condiciones:
a. Haced un test del número de argumentos. El script debe ejecutarse sin argumentos y si hay, debe imprimir un mensaje de uso.
b. Determinad si el directorio de copias de seguridad tiene suficiente espacio libre para almacenar la copia de seguridad.
c. Preguntad al usuario si desea una copia completa (todo) o una copia incremental (solo aquellos archivos que hayan cambiado). Si el usuario no tiene una copia de seguridad completa, se deberá hacer, y en caso de una copia de seguridad incremental, sólo hacerlo si la copia de seguridad completa no tiene más de una semana.
d. Comprimid la copia de seguridad utilizando cualquier herramienta de compresión.
e. Informad al usuario que se hará en cada momento, ya que esto puede tardar algún tiempo y así se evitará que el usuario se ponga nervioso si la salida no aparece en la pantalla.
f. Imprimid un mensaje informando al usuario sobre el tamaño de la copia de seguridad sin comprimir y comprimida, el número de archivos guardados/actualizados y el número de directorios guardados/actualizados.

4. Escribid un script que ejecute un simple navegador web (en modo texto), utilizando **wget** y **links -dump** para mostrar las páginas HTML en un terminal.

El usuario tiene 3 opciones: introducir una dirección URL, entrar **b** para retroceder (back) y **q** para salir. Las 10 últimas URL introducidas por el usuario se almacenan en una matriz, desde donde el usuario puede restaurar la URL utilizando la funcionalidad **b** (back).

Bibliografía

Es recomendable complementar la lectura en la que se ha basado parte de esta documentación y sobre todo para aspectos avanzados, ya que aquí solo se ha visto un breve resumen de lo esencial:

Barnett, Bruce. "AWK". © General Electric Company.

"Big Admin: Script Examples".

Cooper, Mendel. "Advanced Bash-Scripting Guide".

Garrels, Machtelt. "Bash Guide for Beginners".

"Man bash".

Mike G. "Programación en BASH - COMO de introducción".

Robbins, Arnold. "UNIX in a Nutshell" (3ª ed.).

"The GNU awk programming language".

Migración y coexistencia con sistemas no Linux

Josep Jorba Esteve

PID_00167542

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

Una vez realizada una primera aproximación a los sistemas GNU/Linux, el siguiente paso es integrarlos en el entorno de trabajo como sistemas de producción. Según el sistema actual que esté en uso, podemos plantear, o bien una migración total a sistemas GNU/Linux, o bien una coexistencia mediante servicios compatibles.

La migración al entorno GNU/Linux puede hacerse de forma progresiva, sustituyendo servicios parcialmente, o bien sustituyendo todo por los equivalentes GNU/Linux del antiguo sistema.

En los entornos distribuidos actuales, el paradigma más presente es el de cliente/servidor. Cualquier tarea en el sistema global está gestionada por uno o más servidores dedicados, accediendo las aplicaciones o directamente los usuarios a los servicios prestados.

Respecto al ambiente de trabajo, ya sea desde el caso más simple, como el usuario individual, o bien a uno complejo, como un entorno empresarial, en cada entorno se necesitará un conjunto de servicios que tendremos que seleccionar, adaptando luego las máquinas clientes y servidores, para que puedan acceder a éstos o proporcionar sus servicios.

Los servicios pueden englobar muchos aspectos diferentes. Suelen estar presentes varios tipos, ya sea para compartir recursos o información. Son habituales servidores de archivos, de impresión, de web, de nombres, correo, etc.

El administrador, normalmente, seleccionará un conjunto de servicios que deberán estar presentes en el ambiente de trabajo, dependiendo de las necesidades de los usuarios finales, y/o de la organización, y deberá configurar el soporte adecuado a la infraestructura, en forma de servidores que soporten la carga de trabajo esperada (tanto carga interna, como carga del exterior si la hay).

1. Sistemas informáticos: ambientes

En el proceso de algunas instalaciones de distribuciones de GNU/Linux, podemos encontrarnos a menudo que nos preguntan por el tipo de ambiente, o por las tareas a las que va a estar dedicado nuestro sistema. Esto permite muchas veces escoger un subconjunto de software que se nos instalará por defecto, por ser el más adecuado a la función prevista. Es habitual que nos pregunten si el sistema se destinará a:

a) Estación de trabajo (*workstation*): este tipo de sistema incorpora algunas aplicaciones particulares que serán las más usadas. El sistema, básicamente, se dedica a la ejecución de estas aplicaciones y a un pequeño conjunto de servicios de red.

b) Servidor: se integran la mayoría de servicios de red o, en todo caso, alguno particular, el cual va a ser el servicio principal del sistema.

c) Estación dedicada a cálculo: aplicaciones intensivas en cálculo, *renders*, aplicaciones científicas, gráficos CAD, etc.

d) Estación gráfica: escritorio con aplicaciones que necesitan de interacción con el usuario en forma gráfica.

Usualmente, podemos componer nuestro sistema GNU/Linux a partir de una o más de estas posibilidades.

Más en general, si tuviésemos que separar los ambientes de trabajo en que se puede utilizar un sistema GNU/Linux, podríamos identificar tres tipos principales de ambiente: estación de trabajo (*workstation*), servidor y escritorio (*desktop*).

Funciones

Los sistemas GNU/Linux pueden dedicarse a funciones de servidor, estación de trabajo o escritorio.

También se podría incluir otro tipo de sistemas, los que llamaríamos genéricamente como dispositivos empotrados (*embebbed*), o bien sistemas móviles de pequeñas dimensiones, por ejemplo un PDA, un teléfono móvil inteligente (*smartphone*), una videoconsola portátil, etc. GNU/Linux ofrece, asimismo, soporte para estos dispositivos, con *kernels* reducidos y personalizados para ello.

Ejemplo

Destacamos, por ejemplo, un trabajo inicial realizado por la firma Sharp en sus modelos Zaurus, un PDA con Linux de altas prestaciones (varios modelos estuvieron disponibles en las décadas de los noventa y la primera del 2000). O

también otras iniciativas Linux de tipo empotrado como los TPV (terminales punto de venta). O videoconsolas como la GP2X. También es especialmente destacable la evolución en el mercado de los *smartphones* de la plataforma Android de Google Inc., basada en máquina virtual Java sobre un *kernel* Linux modificado, así como otras plataformas móviles, como Nokia Maemo y Intel Moblin.

Nota

Características e historia de Plataforma Android en:

http://en.wikipedia.org/wiki/Android_(operating_system)

http://www.android.com

Respecto a los tres principales ambientes, veamos cómo se desarrolla cada uno de estos sistemas informáticos en un entorno GNU/Linux:

1) Un sistema de tipo ***workstation*** suele ser una máquina de alto rendimiento, utilizada para una tarea específica en lugar de un conjunto general de tareas. La *workstation*, clásicamente, estaba compuesta de una máquina de altas prestaciones con hardware específico adecuado a la tarea que había que desarrollar. Solía tratarse de una máquina Sun Sparc, IBM Risc o Silicon Graphics (entre otras) con sus variantes de UNIX propietarios. Estas máquinas de alto coste se orientaban a un segmento claro de aplicaciones, ya fuese el diseño gráfico 3D (caso Silicon o Sun) o bases de datos (IBM o Sun). Hoy en día, muchos de los actuales PC tienen un rendimiento comparable a estos sistemas (aunque no igual), y la frontera entre uno de estos sistemas y un PC no está ya tan clara, gracias a la existencia de GNU/Linux como alternativa a las versiones de UNIX propietarios. Cabe destacar que muchos de estos fabricantes (Sun, IBM, Silicon) están ofreciendo GNU/Linux como opción de operativo para sus sistemas actuales de gama alta, normalmente mediante convenios con distribuidores Linux comerciales, como Red Hat o Novell SUSE.

2) Un sistema de tipo **servidor** tiene un objetivo concreto, que es ofrecer servicios a otras máquinas de la red: ofrece características o una funcionalidad clara al resto de máquinas. En sistemas informáticos relativamente pequeños (por ejemplo, menores de 5-10 máquinas), no suele haber un sistema exclusivo de servidor (aunque acostumbran a ser comunes los de archivos e impresión), y suele estar compartido con otras funcionalidades, por ejemplo también como máquina de tipo escritorio. En sistemas medianos (unas pocas decenas de máquinas), suele haber una o más máquinas dedicadas a actuar de servidor, ya sea la máquina exclusiva que centra todos los servicios (correo, web, etc.) o un par de máquinas dedicadas a repartirse los servicios principales.

En sistemas grandes (un centenar o más de máquinas, incluso miles), por la capacidad de carga puede ser necesario que exista un buen grupo de servidores, cada uno de ellos dedicado normalmente a algún servicio en exclusiva, o incluso se puede dedicar un conjunto de máquinas exclusivamente a un servicio. Es más, si estos servicios se proporcionan –hacia dentro o hacia fuera

de la organización– mediante acceso por clientes directos o abierto a Internet, dependiendo de la capacidad de carga que tengamos que soportar, tendremos que recurrir a soluciones de tipo SMP (máquinas con varios procesadores interconectados en la misma máquina) o de tipo *clusters* de alta disponibilidad (agrupación de máquinas que se distribuyen la carga de un determinado servicio).

Los servicios que podemos necesitar de forma interna (o externa), podrían englobarse (entre otras) dentro de estas categorías de servicios:

a) Aplicaciones: el servidor dispone de ejecución de aplicaciones y como clientes sólo observamos la ejecución de éstas e interactuamos con ellas. Puede, por ejemplo, englobar servicios de terminales y ejecución de aplicaciones en web.

b) Ficheros: se nos proporciona un espacio común y accesible desde cualquier punto de la red donde almacenar/recuperar nuestros ficheros.

c) Base de datos: se centralizan datos que se van a consultar o producir por parte de las aplicaciones del sistema en red (o bien de otros servicios).

d) Impresión: se dispone de conjuntos de impresoras, donde se gestionan sus colas y los trabajos que se les envíen desde cualquier punto de la red.

e) Correo electrónico: se ofrecen servicios para recibir, enviar o reenviar correos procedentes o destinados tanto al interior como al exterior.

f) Web: servidor (o servidores) propios de la organización, de utilización interna o externa para los clientes.

g) Información de red: en organizaciones grandes es imprescindible poder localizar los servicios ofrecidos, recursos compartidos o los mismos usuarios. Se necesitan servicios que permitan esta localización y consulta de propiedades de cada tipo de objeto.

h) Servicios de nombres: se necesitan servicios que permitan nombrar y traducir los distintos nombres por los que se conoce un mismo recurso.

i) Servicios de acceso remoto: en caso de no disponer de acceso directo, debemos disponer de métodos alternativos que nos permitan interaccionar desde el exterior, que nos dejen acceder al sistema que queramos.

j) Servicios de generación de nombres: en el nombrado de máquinas, por ejemplo, puede darse una situación muy variable de número o que aquellas no sean siempre las mismas. Debemos proporcionar métodos para identificarlas claramente.

k) Servicios de acceso a Internet: en muchas organizaciones no tiene por qué haber accesos directos, sino accesos por medio de pasarelas (*gateways*) o por intermediarios (*proxys*).

l) Servicios de filtrado: medidas de seguridad para filtrar información incorrecta o que afecte a nuestra seguridad, o proteger/limitar el acceso a los recursos.

3) Un sistema de tipo ***desktop*** sería simplemente una máquina que se utiliza para las tareas informáticas rutinarias, de todos los días (por ejemplo, el PC que tenemos en casa o en la oficina).

Ejemplo:

Podríamos definir las siguientes tareas como comunes (se incluyen algunos de los programas GNU/Linux más utilizados):

- **Tareas ofimáticas**. El software clásico de una *suite* ofimática: procesador de texto, hoja de cálculo, presentaciones, alguna pequeña base de datos, etc. Podemos encontrar *suites* como OpenOffice (gratuita), StarOffice (de pago, producida por Sun), KOffice (de KDE), o varios programas como Gnumeric, AbiWord, que formarían una *suite* para Gnome (denominada GnomeOffice).

Nota

Páginas web de *suites* ofimáticas:
http://openoffice.org
http://www.koffice.org/
http://live.gnome.org/GnomeOffice

- **Navegación web**. Navegadores como Mozilla Firefox, Konqueror, Epiphany, etc.

- **Soporte hardware** (dispositivos USB, de almacenamiento...). En GNU/Linux soportados por los controladores adecuados, normalmente proporcionados en el *kernel*, o bien por fabricantes. También hay herramientas de análisis de hardware nuevo, como *kudzu/hal* (Fedora/Red Hat) o *discover* (Debian). Media y entretenimiento (gráficos, procesamiento imágenes, fotografía digital, juegos y más). En GNU/Linux hay una cantidad enorme de estas aplicaciones, de calidad muy profesional: Gimp (retoque fotográfico), Sodipodi, Xine, Mplayer, gphoto, etc.

- **Conectividad** (acceso al escritorio de forma remota, acceso a otros sistemas). Este aspecto en GNU/Linux hay una cantidad enorme de herramientas, ya sea las propias TCP/IP –como ftp, telnet, ssh, web, etc.– como el propio X Window, que tiene capacidades de escritorio remoto hacia cualquier máquina UNIX, rdesktop (para conectarse a escritorios Windows), o VNC (que permite conectarse a UNIX, Windows, Mac, etc.).

2. Servicios en GNU/Linux

GNU/Linux dispone de servidores adaptados para cualquier ambiente de trabajo.

Las categorías de los servicios que hemos apuntado tienen equivalentes en servicios que podemos proporcionar desde nuestros sistemas GNU/Linux al resto de máquinas de la red (y de los que también podremos actuar como cliente):

a) Aplicaciones: GNU/Linux puede ofrecer servicios de terminales remotos, ya sea por conexión directa mediante interfaces serie de terminales "tontos", que sirvan para visualizar o interactuar con las aplicaciones. Otra posibilidad es la conexión remota de modo textual, desde otra máquina, por medio de servicios TCP/IP como los *rlogin*, telnet, o de forma segura con *ssh*. GNU/Linux proporciona servidores para todos estos protocolos. En el caso de ejecutar aplicaciones gráficas, disponemos de soluciones mediante X Window de forma remota. Cualquier cliente UNIX, Linux o Windows (u otros) que dispongan de un cliente X Window puede visualizar la ejecución del entorno y sus aplicaciones. Asimismo, hay otras soluciones como VNC para el mismo problema. En cuanto a las aplicaciones vía web, GNU/Linux dispone del servidor Apache, y cualquiera de los múltiples sistemas de ejecución web están disponibles, ya sean Servlets/JSP (con Tomcat), Perl, PHP, xml, *webservices*, etc., así como servidores de aplicaciones web como Oracle/BEA WebLogic, IBM Websphere, JBoss (libre), que también se ejecutan sobre plataformas GNU/Linux.

b) Ficheros: pueden servirse ficheros de múltiples maneras, desde proporcionar acceso a ellos por ftp como servirlos de forma transparente a otras máquinas UNIX y GNU/Linux con NFS, o bien actuar de cliente o servidor hacia máquinas Windows mediante Samba/CIFS.

c) Base de datos: soporta una gran cantidad de bases de datos cliente/servidor de tipo relacional, como MySQL y PostgreSQL, y varias comerciales, como Oracle o IBM DB2, entre otras.

d) Impresión: puede servir impresoras locales o remotas, tanto a sistemas UNIX con protocolos TCP/IP como a Windows mediante Samba/CIFS.

e) Correo electrónico: ofrece tanto servicios para que los clientes obtengan correo en sus máquinas (servidores POP3 o IMAP), como agentes MTA (*mail transfer agent*) para recuperar y retransmitir correo, como el servidor Sendmail (el estándar UNIX) u otros como Exim o Postfix y, en el caso de envíos externos, el servicio de SMTP.

f) Web: está disponible el servidor http Apache, ya sea en sus versiones 1.3.x o las nuevas 2.0.x/2.2.x. Además, podemos integrar servidores de aplicaciones web, como Tomcat para servir servlets/JSP o servidores de aplicaciones como JBoss.

g) Información de red: servicios como NIS, NIS+ o LDAP nos permiten centralizar la información de las máquinas, usuarios y recursos varios de nuestra red, facilitando la administración y los servicios a los usuarios, de manera que éstos no dependan de su situación en la red. Si nuestra organización tiene cierta estructura interna, estos servicios nos permiten modelarla (mediante estructuras jerárquicas), dejando acceso a los recursos a quien los necesita.

h) Servicios de nombres: servicios como DNS para los nombres de las máquinas y su traducción desde IP o a IP, por medio de, por ejemplo, el servidor Bind (el DNS estándar UNIX).

i) Servicios de acceso remoto: ya sea para ejecutar aplicaciones o para obtener información remota de las máquinas. Los servidores podrían ser los que hemos comentado para aplicaciones: X Window, VNC, etc., y también los que permiten ejecutar algunos comandos remotos sin interactividad, como *rexec*, *rsh*, *ssh*, etc.

j) Servicios de generación de nombres: servicios como DHCP permiten redes TCP/IP, una generación dinámica (o estática) de las direcciones IP que se disponen en función de las máquinas que las necesiten.

k) Servicios de acceso a Internet: en determinadas situaciones puede tenerse un único punto de salida a Internet (o varios). Estos puntos suelen actuar como *proxy*, ya que tienen el acceso y lo redirigen a los posibles accesos a Internet por parte de los clientes. También suelen actuar de caché de contenidos. En GNU/Linux podemos disponer, por ejemplo, del Squid. Dentro de esta categoría, también podría entrar la actuación de un sistema GNU/Linux de pasarela (*gateway*) o de *router*, ya sea para dirigir paquetes hacia otras redes o para buscar rutas de reenvío alternativas. También en el caso de pequeñas instalaciones como las domesticas, podríamos incluir el acceso a Internet mediante módem por los servicios PPP.

l) Servicios de filtrado: una de las medidas de seguridad más utilizadas actualmente es la implantación de cortafuegos (o *firewalls*). Consiste en técnicas de filtrado de los paquetes entrantes o salientes, de los diferentes protocolos

que estemos usando, para poner barreras a los no deseados. En GNU/Linux disponemos de mecanismos como *ipchains* (obsoleto) e *iptables* (más moderno, también denominado Netfilter) para implementar los cortafuegos.

3. Tipologías de uso

GNU/Linux ofrece, como sistema, características válidas para el uso desde el usuario personal hasta el usuario de una infraestructura de media o gran escala.

Desde la perspectiva de los usuarios de los sistemas GNU/Linux, podríamos diferenciar usuarios individuales, de media escala y de organización amplia:

a) El usuario individual o usuario doméstico: este tipo de usuario dispone de una o varias máquinas en su hogar, que serán compartidas o no. En general, en este ambiente, GNU/Linux se usaría para desarrollar un sistema de escritorio, con lo cual será importante la parte gráfica: el escritorio de GNU/Linux. También está adquiriendo importancia, por el crecimiento de los contenidos digitales (foto, video, música), la utilización de GNU/Linux en un ambiente doméstico, como servidor de medios digitales, y/o servidor de archivos, en forma o bien de servidores creados a medida, o de productos comerciales tipo NAS (*network attached storage*) caseros que proporcionan servidores Samba, NFS, FTP, impresión, *backup* y *streaming* de video/audio.

En cuanto a la parte de escritorio tenemos dos opciones principales, en forma de los entornos **Gnome** y **KDE**. Los dos entornos constituyen opciones perfectamente válidas. Ambos disponen de servicios de visualización y ejecución de las aplicaciones, así como de un amplio conjunto de aplicaciones propias básicas que nos permiten desarrollar todo tipo de tareas rutinarias. Los dos entornos ofrecen un escritorio visual con diferentes menús, barras de utilidad e iconos, así como navegadores de ficheros propios y aplicaciones de utilidad variadas. Cada entorno puede ejecutar sus aplicaciones propias y las disponibles en el otro entorno, aunque cada aplicación tiene mejor ejecución en su entorno propio, por tener un aspecto visual más acorde al entorno para el que se diseñó.

En cuanto a las aplicaciones para el usuario personal, incluiríamos las típicas del sistema de escritorio. En el caso de que el usuario disponga de una red en su casa, por ejemplo, un pequeño conjunto de ordenadores con una red de tipo Ethernet, también podrían ser interesantes servicios para compartir ficheros e impresoras entre las máquinas. Podrían ser necesarios servicios como NFS, si hay otras máquinas Linux, o bien Samba, si hay máquinas con Windows (o Mac). Los sistemas NAS caseros intentan integrar todos estos servicios junto con un conjunto de almacenamiento en disco (RAID), accesible por red mediante diferentes protocolos (ftp, http, nfs, samba...).

En el caso de tener una conexión a Internet por algún proveedor de acceso (ISP), según la forma de conexión utilizada, necesitaríamos controlar los dispositivos y los protocolos correspondientes:

- **Conexión por módem**: los módems telefónicos suelen utilizar el protocolo PPP de conexión con el proveedor. Tendríamos que habilitar este protocolo y configurar las cuentas que tengamos habilitadas en el proveedor. Un problema importante con Linux es el tema de los winModems. Estos módems (con excepciones) no están soportados, ya que no son un módem real, sino una simplificación hardware más un software de *driver*, y la mayoría funcionan únicamente con Windows, por lo que hay que evitarlos (si no están soportados) y comprar módem "reales" (completos).

- **Conexión mediante un módem ADSL**: el funcionamiento sería parecido, se podría utilizar el protocolo PPP u otro denominado EoPPP. Esto puede depender del fabricante del módem y del tipo, Ethernet o USB.

- **Conexión por ADSL con *router***: la configuración es muy simple, debido a que en esta situación sólo hay que configurar la tarjeta de red Ethernet y/o la tarjeta *wireless* en nuestro sistema para conectar al *router* ADSL (éste suele proporcionar servicios de IP dinámica para las máquinas cliente).

Una vez la interfaz a Internet está conectada y configurada, el último punto es incluir el tipo de servicios que necesitaremos. Si sólo queremos actuar como clientes en Internet, bastará con utilizar las herramientas cliente de los diferentes protocolos, ya sea ftp, telnet, el navegador web, el lector de correo o *news*, etc. Si además queremos ofrecer servicios hacia el exterior –por ejemplo, publicar una web (servidor web) o permitir nuestro acceso externo a la máquina (servicios de ssh, telnet, ftp, X Window, VNC, etc.), en este caso, servidor– cabe recordar que esto sólo será posible, en principio, si nuestro proveedor nos ofrece direcciones IP fijas (estáticas) para nuestra máquina. De otro modo, nuestra dirección IP cambiaría a cada conexión y la posibilidad de proporcionar un servicio se volvería muy difícil o imposible (hay posibles soluciones a este problema, como los servicios, gratuitos, ofrecidos por la empresa DynDNS).

Otro servicio interesante sería compartir el acceso a Internet entre las máquinas de que dispongamos a partir de un sistema GNU/Linux que esté conectado al exterior (actuando de *proxy* o *gateway* para el resto de máquinas). De hecho, varios *routers* ADSL son en realidad servidores de este tipo, y muchos fabricantes utilizan un GNU/Linux empotrado en el *firmware* de los *routers*. Aun así, si disponemos de una única máquina con acceso a Internet (módem telefónico o módem ADSL), podemos establecer esta máquina con conexión compartida para el resto de nuestra red casera.

b) Usuario de media escala: es un usuario de una organización de media escala, ya sea una pequeña empresa o un grupo de usuarios. Estos usuarios dispondrán de conectividad en red local (por ejemplo, una LAN) con algunas máquinas e impresoras conectadas. Y tendrán acceso directo a Internet, bien por medio de algún *proxy* (punto o máquina destinada a la conexión externa), o bien a través de unas pocas máquinas conectadas físicamente a Internet. En general, en este ambiente, el trabajo suele ser en parte local y en parte compartido (ya sea recursos como las impresoras o aplicaciones comunes). Normalmente, necesitaremos sistemas de escritorio, por ejemplo, en una oficina podemos utilizar las aplicaciones ofimáticas junto con clientes Internet, y quizás también sistemas de tipo *workstation*. Por ejemplo, en trabajos de ingeniería o científicos pueden utilizarse aplicaciones de CAD, de procesamiento de imágenes, aplicaciones de cálculo matemático intensivo, etc., y seguramente habrá algunas máquinas más potentes destinadas a estas tareas.

Cabe comentar que este ambiente está un poco en transición hacia el hecho de ser el común del usuario individual, ya que debido a las conexiones de banda ancha disponibles en los hogares cada vez se amplían más las características del usuario doméstico y se acercan a este segundo ambiente. De hecho, a veces el usuario de media escala suele denominarse como SOHO (Small Office, Home Office).

En este ambiente de uso, necesitaremos servicios de compartición de recursos como ficheros, impresoras, posiblemente aplicaciones, etc. Por lo tanto, en un sistema GNU/Linux serán adecuados los servicios de NFS, servicios de impresión y Samba (si hay máquinas Windows con las que compartir ficheros o impresoras). También es posible que tengamos necesidad de entornos de bases de datos, algún servidor interno de web con aplicaciones compartidas, etc.

c) Usuario de organización amplia: este tipo de usuario es una evolución del anterior, y se diferencia de él en el tamaño de la organización y en los recursos de los que puede disponer, que podrían llegar a ser muy altos, de modo que se necesitarían algunos recursos de sistemas de directorio de red de tipo NIS, NIS+ o LDAP para poder manejar la gran cantidad de información de la organización y reflejar su estructura, así como, seguramente, disponer de grandes infraestructuras de servicios hacia los clientes externos, por lo general en forma de sitios web con aplicaciones distintas.

En este tipo de organizaciones se presentan niveles de heterogeneidad elevados, tanto en el hardware como en el software de los sistemas, y podríamos encontrar muchas arquitecturas y diferentes sistemas operativos, por lo que la tarea principal consiste en facilitar la compatibilidad de los datos vía bases de datos y formatos de documentos estándar y facilitar la interconexión mediante protocolos, clientes y servidores estándar (con elementos TCP/IP).

4. Migrar o coexistir

A continuación, vamos a plantear otro aspecto importante en el proceso de adopción de los sistemas GNU/Linux. Supongamos que o bien somos principiantes en el manejo de este sistema o, por el contrario, que somos experimentados y queremos adoptar uno o varios sistemas GNU/Linux como usuarios individuales, para el trabajo en nuestra pequeña organización, o nos estamos planteando sustituir la infraestructura completa (o parcial) de nuestra gran empresa u organización.

Realizar esta migración no es algo trivial; hay que evaluar las opciones mediante un estudio en el que se analicen tanto los costes como las prestaciones que esperamos obtener. Además, puede realizarse total o parcialmente, con cierto grado de coexistencia con los antiguos sistemas.

Nos encontramos ante un proyecto de migración, total o parcial, de nuestros sistemas informáticos hacia GNU/Linux, y como administradores somos responsables de este proceso.

Como en todo proyecto, habrá que estudiar el modo de responder a cuestiones como: ¿es rentable el cambio, en prestaciones, en coste?, ¿con qué objetivo lo hacemos?, ¿qué requerimientos queremos o debemos cumplir?, ¿podemos hacer o es necesaria una migración completa?, ¿tiene que haber coexistencia con otros sistemas?, ¿habrá que formar de nuevo a los usuarios?, ¿podremos utilizar el mismo hardware o necesitaremos uno nuevo?, ¿habrá costes añadidos importantes? o, simplemente, ¿saldrá bien? Esta y muchas preguntas más son las que tendremos que intentar responder. En el caso empresarial, la respuesta pasaría por la definición de un proyecto/plan de migración, con sus objetivos, análisis de requerimientos, proceso de implantación, estudios económicos, planes de formación de usuarios, calendarios de fases de desarrollo e implementación, etc. No entraremos en esto, pero nos plantearemos algunas de las cuestiones de forma sencilla. Y en el apartado 5 examinaremos unos pequeños casos prácticos de cuestiones a examinar en el proceso de migración.

Además, en el momento en que empecemos la migración a los sistemas GNU/Linux es cuando comenzaremos a apreciar las ventajas que aportará a nuestra organización:

a) Costes: reducción de los costes, en licencias software del sistema y de las aplicaciones. GNU/Linux tiene un coste 0 en cuanto a las licencias, si se obtiene desde la Red (por ejemplo, en forma de imágenes de los CD de la distribu-

ción), o un coste despreciable teniendo en cuenta que la mejor comparación para sistemas equivalentes en prestaciones serían sistemas Windows Server, con costes que se sitúan en rangos de varios miles de euros por licencia, sin incluir gran parte del software extra que proporciona una distribución GNU/Linux típica.

Pero cuidado, no hay que desestimar los costes de mantenimiento y formación. Si nuestra organización sólo está formada por usuarios y administradores Windows, podemos tener costes altos en nueva formación, personal, y quizás mantenimiento. Por eso, grandes empresas desean depender de algún distribuidor comercial de GNU/Linux para que les implante y mantenga el sistema, como por ejemplo las versiones empresariales que ofrecen Red Hat, SuSe y otros. Estas versiones GNU/Linux también tienen costes de licencia altos (comparables a Windows), pero, por el contrario, están ya adaptadas a estructuras empresariales y traen software propio para gestionar la infraestructura informática de las empresas. Otro aspecto importante, que resumiría esta estimación de costes, es el concepto de TCO (*total cost of ownership*), como evaluación global de los costes asociados que nos encontraremos al emprender un desarrollo tecnológico. No sólo hay que evaluar los costes de licencias y maquinaria, sino también los costes de soporte y formación de las personas y productos implicados, que pueden ser tan importantes o más que los de la solución implementada.

b) Soporte: GNU/Linux tiene el soporte de mantenimiento mayor que haya tenido un sistema operativo, y gratis en su mayor parte. A pesar de ello, algunas empresas no lo adoptan por ciertos temores, objetando que no hay soporte del producto, y se dedican a comprar distribuciones comerciales que les ofrecen contratos de soporte y mantenimiento. GNU/Linux tiene una comunidad de soporte mundial bien establecida, por medio de diferentes organizaciones que proporcionan documentación libre (los famosos Howto's), foros de usuarios especializados, comunidades de usuarios de prácticamente cualquier región o país del mundo, etc. Cualquier duda o problema con el que nos encontremos puede buscarse (por ejemplo, por alguno de los buscadores en Internet), y podemos tener respuestas en minutos. Cuando no, si hemos encontrado un *bug*, error, o situación no probada, podemos informar de ella en varios lugares (foros, sitios de desarrollo, sitios de *bugs* de distribuciones, etc.) y obtener soluciones en horas o a lo sumo en algunos días. Siempre que aparezca una duda o algún problema, hay que intentar primero algunos procedimientos (así se desarrolla el aprendizaje de administración), y si no obtenemos solución en un tiempo prudencial, consultar a la comunidad GNU/Linux por si a algún otro usuario (o grupo de ellos) le ha ocurrido el mismo problema y ha obtenido solución. Si no, siempre podemos informar del problema (sitios de *bugs* de las distribuciones), para que nos planteen algunas soluciones temporales, o se comiencen a elaborar revisiones de los paquetes software afectados, si se considera que es un problema importante.

Algunas fuentes de soporte GNU/Linux

Linux Howto's: http://www.tldp.org/

HowtoForge: http://howtoforge.org/

LinuxQuestions: http://www.linuxquestions.org

Linux Forum: http://www.linuxforums.org/forum/

Soporte Debian:

http://www.debianhelp.org/

http://forums.debian.net/

http://www.debian-administration.org/

Soporte Fedora:

http://fedoraforum.org/

http://fedorasolved.org/

http://fedoraunity.org/

4.1. Identificar requerimientos de servicios

Si tenemos unos sistemas ya funcionando, deberemos tener implantados algunos servicios de los cuales serán clientes los usuarios, o servicios que ayuden a la infraestructura del soporte informático. Los servicios entrarán dentro de alguna de las categorías vistas anteriormente, con las opciones GNU/Linux que comentamos.

Los sistemas GNU/Linux no son "nuevos" en absoluto, y derivan (como vimos en la introducción) de una historia de más de cuarenta años de uso y desarrollo de los sistemas UNIX. Gracias a ello, una de las primeras cosas que veremos es que no nos falta soporte para ningún tipo de servicio que queramos. Si acaso, habrá diferencias en la forma de hacer las cosas. Además, muchos de los servicios que se utilizan en los sistemas informáticos fueron pensados, investigados, desarrollados e implementados en su día para UNIX, y posteriormente adaptados a otros sistemas (como Windows, con más o menos acierto).

Muchas de las empresas que disponen de UNIX propietarios participan en GNU/Linux y ofrecen algunos de sus desarrollos a la comunidad.

Cualquier servicio disponible en cada momento podrá ser adaptado en los sistemas GNU/Linux con servicios equivalentes (cuando no iguales).

Ejemplo

Un caso famoso es el de los servidores Samba. Windows ofrece lo que él denomina "compartir archivos e impresoras en red" mediante unos protocolos propios denominados genéricamente SMB (*server message block*) (con apoyo de red en los protocolos NetBios y NetBEUI). También es de común uso el nombre CIFS (*common Internet file system*), que es como se denominó al protocolo en una segunda revisión (que seguía incluyendo a SMB como protocolo base). Estos protocolos permiten compartir carpetas de archivos (o discos) y de impresoras en una red de máquinas Windows (en una configuración de *workgroup,* o trabajo en grupo, o en dominios Windows). En UNIX esta idea ya era antigua, cuando apareció en Windows, y se disponía de servicios como NFS de compartición de archivos o la gestión remota de impresoras, bajo protocolos TCP/IP.

Uno de los problemas de sustituir los servicios Windows de compartición basados en NetBios/NetBeui (y últimamente con NetBios sobre TCP/IP), era cómo dar soporte a estos protocolos, ya que, si queríamos conservar las máquinas clientes con Windows, no podíamos utilizar los servicios UNIX (aunque distintos fabricantes han implementado clientes UNIX para Windows, suelen ser una opción o bien cara o bien no completa). Para este problema, Samba se desarrolló como un servidor para UNIX que soportaba los protocolos Windows y podía sustituir a una máquina cliente/*server* Windows de forma transparente; los usuarios clientes con Windows no tenían por qué notar absolutamente nada. Es más, el resultado fue que en la mayor parte de los casos el rendimiento era comparable, cuando no era mejor que en la máquina original con los servicios Windows.

Actualmente Samba evoluciona constantemente para mantener la compatibilidad de los servicios Windows de compartición de impresoras y archivos, debido a los cambios generales a que Microsoft somete los protocolos SMB/CIFS (base que Samba implementa) en cada nueva versión de Windows; en particular, la evolución desde los esquemas de trabajo en grupo en sus versiones cliente del operativo, a los esquemas centralizados en servidor (o en grupos de ellos), con servicios particulares de autentificación de usuarios (NTLM, NTLMv2, Kerberos) y almacenamiento centralizado de la gestión del sistema como Active Directory. También se varía la configuración de servidores de dominios en las diferentes versiones de Windows *server* existentes (ya sean con servidores controladores primarios, *backup* o Active Directory).

Actualmente, en los procesos de migración con Samba tendremos que observar qué configuraciones de clientes/servidores Windows (y las versiones de éste) existen en el sistema informático, así como qué mecanismos de autentificación de los usuarios y/o gestión de la información se utilizan (grupos, dominios, sistema de autentificación). Además, necesitaremos conocer la estructuración del sistema informático en dominios (y sus servidores controladores, miembros o servidores aislados), para poder realizar un mapeado completo y correcto hacia soluciones basadas sobre Samba, y en servicios complementarios de autentificación de usuarios (*winbind*, *kerberos*, *nss_ldap*) y gestión de directorios (como por ejemplo openLDAP).

4.2. Proceso de migración

En el proceso de migración, hay que tener en cuenta qué se quiere migrar, y si quiere realizarse de forma completa o parcial, coexistiendo con otros servicios o equipos o con un sistema operativo diferente.

En estos ambientes, como en las grandes organizaciones, en los que encontramos un gran número de sistemas heterogéneos, habrá que tener en cuenta que seguramente no se migrarán todos, en especial los sistemas de tipo *workstation* dedicados a la ejecución de alguna aplicación básica para una tarea. Puede que no exista la aplicación equivalente o simplemente podemos desear quedarnos con esos sistemas por razones de coste o para rentabilizar la inversión realizada.

Podemos migrar varios elementos, ya sean los servicios que ofrecemos, las máquinas que los sirven o los clientes que acceden a ellos. Los elementos que se migren pueden ser variados.

En la migración, pasaremos por la sustitución de un servicio por otro equivalente, normalmente con el menor impacto posible si no queremos sustituir también a los clientes.

En caso de clientes Windows, podemos usar el servidor Samba para sustituir los servicios de archivos e impresoras que proporcionaban las máquinas Windows. Si se trata de otros servicios, podremos sustituirlos por los equivalentes GNU/Linux. En el caso de sustituir sólo algún servicio, normalmente se inhabilitará el servicio en la máquina que lo ofrecía y se habilitará en el sistema nuevo. Pueden ser necesarios cambios en los clientes (por ejemplo, direcciones de la nueva máquina o parámetros relacionados con el servicio).

Si la función la cumplía por entero una máquina servidora, hay que analizar si la máquina estaba dedicada a uno o más servicios y si todos podrán ser sustituidos. En tal caso, sólo hay que reemplazar la máquina antigua por la nueva (o mantener la antigua) con los servicios bajo GNU/Linux y, en todo caso, modificar algún parámetro en los clientes si fuese necesario. Normalmente, antes de efectuar el cambio, es conveniente testear la máquina por separado con algunos clientes para asegurarse de que cumple su función correctamente y sustituir las máquinas en algún periodo de inactividad del sistema.

En cualquier caso, seguramente habrá que hacer *backups* de los datos anteriores al nuevo sistema, por ejemplo, el sistema de ficheros o las aplicaciones disponibles en el servidor original. Otro de los puntos previos a tener en cuenta es la portabilidad de los datos; un problema que a menudo presenta difícil solución si en la organización se utilizaban formatos de datos o aplicaciones dependientes de una plataforma.

Algunos casos prácticos de problemas con que se encuentran algunas empresas hoy en día son:

- **1) Aplicaciones en web con ASP o ASP.net**: algunas de estas aplicaciones son sólo realizables en plataformas web con Windows y el servidor web IIS de Microsoft. Habría que evitarlas, si en algún momento pensamos hacer una migración de plataformas, y no queremos reescribirlas o pagar a una empresa para que lo haga. En plataformas GNU/Linux está disponible el servidor web Apache (el más utilizado en Internet), que también se puede utilizar con Windows. Este servidor soporta ASP en Perl (en Windows se suele utilizar visual basic, C# y Javascript generalmente) y existen soluciones de terceros para migrar los ASP o más o menos convertirlos. Pero si nuestra empresa dependiese de esto, sería muy costoso en tiempo y dinero. Una solución práctica habría sido realizar los desarrollos web en Java (que sí que es portable entre plataformas) u otras soluciones como PHP. En este punto cabe destacar el proyecto Mono (patrocinado por Novell) para la portabilidad de parte del entorno .NET de Microsoft a GNU/Linux, en particular gran parte de las API de .NET, el lenguaje C#, y la especificación ASP.NET. Permiten una migración flexible de aplicaciones .NET basadas en API .NET que estén soportadas por la plataforma Mono. Por otra parte, cabe señalar el proyecto DotGnu de la FSF, como alternativa GPL a Mono.

- **2) Bases de datos**: utilizar, por ejemplo, un SQL *server* de Microsoft nos hace totalmente dependientes de su plataforma Windows. Además, si utilizamos soluciones propietarias en un entorno concreto para aplicaciones de la base de datos, serán de difícil transferencia. Otras bases de datos, como Oracle y DB2 (de IBM) son más portables por disponer de versión en las diferentes plataformas (GNU/Linux incluida), o por utilizar lenguajes de programación más portables. También se podría trabajar con sistemas de bases de datos PostgreSQL o MySQL (también tiene versión para Windows) disponibles en GNU/Linux, y que permiten una transición más fácil. Asimismo, si se combina con el desarrollo web tenemos muchas facilidades. En este sentido, hoy en día se utilizan sistemas como aplicaciones web con Java, ya sea *servlets*, *applets* o EJB, o bien soluciones como las famosas LAMP combinación de GNU/Linux, Apache, Mysql y Php.

4.2.1. Workstation

En estas migraciones el mayor problema parte de las aplicaciones, ya que son las que dan su razón de ser a la estación de trabajo, ya sean programas de CAD, de animación, de ingeniería o científicos. Aquí será importante que podamos sustituirlas por aplicaciones iguales o, como mínimo, compatibles con las mismas características o con la funcionalidad esperada, o con portabilidad de formatos de datos o ficheros. Normalmente, la mayor parte de estas aplicaciones ya provienen de un mundo UNIX, puesto que la mayoría de estas *workstations* estaban pensadas como máquinas UNIX. Por ello, quizás baste una recompilación o una adaptación mínima al nuevo sistema GNU/Linux, si disponemos del código fuente (como suele pasar en muchas aplicaciones científicas). Si se trata de aplicaciones comerciales, los fabricantes (de software de ingeniería y científico) comienzan a adaptarlas a GNU/Linux, aunque en estos casos las aplicaciones suelen ser muy caras (pueden ir perfectamente de miles a cente-

nares de miles de euros). También para algunos softwares hay soluciones libres compatibles en mayor o menor medida. Hay que examinar qué soluciones libres tenemos y si se adaptan mínimamente a nuestras necesidades.

4.2.2. Máquinas clientes de escritorio

Las máquinas de escritorio continúan siendo un quebradero de cabeza en el mundo GNU/Linux, ya que ofrecen bastantes problemas adicionales. En los servidores, las máquinas se destinan a funcionalidades claras, que en general no requieren interfaces gráficas complejas (muchas veces con comunicación textual es suficiente), el hardware es de propósito específico y de altas prestaciones, se compra para unas funcionalidades concretas y las aplicaciones suelen ser los propios servidores incluidos en el sistema operativo o algunos de terceros. Además, estas máquinas suelen estar gestionadas por personal de tipo administrador que tiene amplios conocimientos de lo que gestiona. Por contra, en el caso del escritorio, nos encontramos con un factor problemático (en sí mismo, y aún más para los administradores): los usuarios finales del sistema. Los usuarios de escritorio esperan disponer de potentes interfaces gráficas, más o menos intuitivas, y de aplicaciones que permitan desarrollar sus tareas rutinarias, normalmente ofimáticas. Este tipo de usuario (con excepciones) no tiene por qué tener unos conocimientos informáticos elevados. En general, sus conocimientos son de ofimática y suele usar un reducido número de aplicaciones con mayor o menor dominio de éstas. Aquí GNU/Linux tiene un problema claro, ya que UNIX como tal nunca fue pensado como un sistema puramente de escritorio, y sólo fue adaptado a posteriori por sistemas gráficos como X Window y los diferentes entornos de escritorios, como los actuales de GNU/Linux: Gnome y KDE. Además, el usuario final suele estar acostumbrado a sistemas Windows (que copan casi un 95% del mercado de escritorio), aunque la tendencia también está cambiando, con más plataformas disponibles en escritorio como MacOS X (de hecho, un UNIX con interfaz de escritorio propia, aunque puede usar también X Window).

> El ambiente de escritorio es una batalla todavía por librar para los sistemas GNU/Linux; tienen que vencer la desconfianza de los usuarios a cambiar de sistema y saber dar a conocer que ofrecen alternativas de sencillez y aplicaciones, que solucionan las tareas de los usuarios.

En el caso del escritorio, GNU/Linux tiene que superar unos cuantos obstáculos. Uno de los más críticos es que no viene preinstalado en las máquinas (esta tendencia está cambiando gracias a algunos fabricantes y al emergente mercado de los portátiles tipo *notebook*), lo que obliga al usuario a tener conocimientos para poder instalarlo.

Otros problemas podrían ser:

- **Desconfianza del usuario**: una pregunta que se puede plantear un usuario es ¿por qué debo cambiar de sistema?, o ¿me ofrecerá lo mismo el nuevo entorno? Una de las razones básicas para hacer el cambio sería el software de calidad y su precio, del que una buena parte es libre. En este punto, afecta el tema de las copias de software ilegales. Parece ser que los usuarios consideran que su software es gratis, cuando en realidad están en una situación ilegal. El software GNU/Linux ofrece gran calidad a bajo coste (o gratis en muchos casos) y ofrece múltiples alternativas para una misma tarea.

- **Sencillez**: a veces, el usuario se muestra perdido si el sistema no le ofrece algunas referencias que lo hagan parecido a lo que ya conoce, como el comportamiento de la interfaz, o que las herramientas sean parecidas en funcionalidad (de hecho, lo que se denomina la usabilidad del sistema). Espera que, en general, no necesite mucho tiempo extra para aprender y manejar el nuevo sistema. GNU/Linux aún presenta algunos problemas en las instalaciones más o menos automáticas, para las que, aunque mejoran día a día, todavía es necesario un cierto grado de conocimiento. En este punto, cabe destacar la facilidad de instalación en diferentes ambientes ofrecida por distribuciones recientes orientadas a escritorio como Ubuntu, que han permitido que su instalación sea muy sencilla y comparable o incluso mejor que las otras plataformas. Otro problema habitual radica en el soporte del hardware del PC, al que, a pesar de que está mejorando mucho, los fabricantes actualmente no le prestan la atención adecuada (en parte por la cuota de mercado de usuario personal). Hasta que no haya una clara intención en este aspecto, no podremos tener el mismo soporte que en otros sistemas propietarios (como en Windows). Aunque hay que destacar el trabajo de la comunidad del *kernel* de Linux para dar el soporte adecuado a nuevas tecnologías, en algunos casos ayudando al fabricante, o preparando soporte primario (si no está soportado por el fabricante) o alternativo al ofrecido por el fabricante.

- **Transparencia**: los entornos GNU/Linux tienen muchos mecanismos complejos, como los *daemons*, servicios, ficheros ASCII difíciles de configurar, etc. De cara a un usuario final, sería necesario poder ocultar todas estas complejidades, mediante programas gráficos, asistentes de configuración, etc. Es uno de los caminos que han tomado algunas distribuciones como Red Hat, Mandriva, Ubuntu o Novell SuSe.

- **Soporte de aplicaciones conocidas**: un usuario ofimático típico tendrá el problema de la portabilidad de sus datos, o del tratamiento de los formatos de éstos. ¿Qué hace con los datos que tenía hasta el momento? Esta cuestión se está mejorando día a día, gracias a las *suites* ofimáticas que comienzan a tener las funcionalidades necesarias para un usuario de escritorio. Por ejemplo, si nos planteamos una migración desde un uso de un paquete Office de Windows, podemos encontrar *suites* como OpenOffice (software libre y gratuito), que puede leer (y crear) los formatos (con algunas restricciones) de ficheros Office (de diferentes versiones). La compatibilidad de formatos no es que sea difícil, cuando éstos son abiertos, pero en el caso Windows Microsoft continúa manteniendo una política de formatos cerrados (con algunas modificaciones en los últimos tiempos), y hay que hacer un trabajo considerable para poder utilizar estos formatos, mediante reingeniería inversa, un proceso bastante costoso. Además, en la era de Internet, donde la información se supone que se mueve libremente, los formatos cerrados sin documentar son más un obstáculo que otra cosa. Lo mejor es utilizar formatos abiertos editables como RTF (aunque este caso también tiene algún problema, por las múltiples versiones que existen de él), o bien formatos basados en XML (OpenOffice genera sus documentos propios en formatos basados en XML), o PDF para la documentación de lectura. También hay que destacar los esfuerzos realizados recientemente por la comunidad OpenOffice para la creación del estándar OpenDocument (usado por la *suite* a partir de las versiones 2.x y adoptado por otras *suites* ofimáticas), que han permitido disponer de un formato libre como estándar ISO para la creación de documentos. Este hecho ha obligado a Microsoft, a abrir (parcialmente) su formato en las versiones a partir de Office 2007, incorporando los formatos documentados OpenXML (y a incluir soporte para los formatos OpenDocument), ya que muchas administraciones públicas obligan en sus contratos de sistemas de información a que los formatos de datos sean estándares abiertos.

- **Alternativas válidas**: el software que se deja de usar tiene que tener alternativas que cumplan el trabajo anterior en el otro sistema. En la mayoría de aplicaciones existen una o varias alternativas con funcionalidades parecidas, cuando no superiores. Pueden encontrarse por Internet diferentes listas de equivalencias (más o menos completas) de aplicaciones Windows con sus correspondientes GNU/Linux (hemos comentado algunas anteriormente en este módulo).

- **Soporte de ejecución de otras aplicaciones de diferentes sistemas**: en algunas condiciones es posible ejecutar aplicaciones de otros sistemas UNIX (de la misma arquitectura, por ejemplo Intel x86), o bien de msdos o Windows, mediante paquetes de compatibilidad o algún tipo de emuladores. GNU/Linux dispone de herramientas de escritorio remoto para otras plataformas, y de soporte de aplicaciones Windows con paquetes como Wine, que permiten ejecutar abundante cantidad de software sin problemas. Otra posibilidad, para software que deba mantenerse, es la creación de má-

Nota

Para ejemplos de aplicaciones equivalentes:

http://www.linuxalt.com/

http://wiki.linuxquestions.org/wiki/Linux_software_equivalent_to_Windows_software

http://www.linuxrsp.ru/win-lin-soft/table-eng.html

quinas virtuales que ejecuten la plataforma con el software que no hemos podido migrar. Sistemas como VirtualBox, Vmware u otros nos darán soporte en este caso.

La mayor parte de estos problemas que aparecen en las migraciones de escritorio están superándose poco a poco y nos permitirán, en el futuro, disfrutar de una mayor cuota de usuarios GNU/Linux en el escritorio y, a medida que aumenten, disponer de mejores aplicaciones, ya que las empresas de software implementarán versiones para GNU/Linux.

En el caso empresarial, esto puede superarse con una migración suave, primero de las etapas de servidores y *workstations*, para después pasar por un proceso de formación amplia de los usuarios en los nuevos sistemas y aplicaciones y, finalmente, integrarlos en su escritorio.

Un proceso que va a ayudar en gran medida es la introducción del software de código abierto en diferentes frentes, por ejemplo en las fases educativas y en las administraciones públicas. En este sentido, son pioneros: la comunidad autónoma de Extremadura, con su distribución GNU/Linux llamada Linex, el GuadaLinex de la Junta de Andalucía, o bien recientes medidas para implantar este software en la educación primaria, o las iniciativas de algunas universidades de llevar a cabo cursos y materias con estos sistemas.

5. Taller de migración: análisis de casos de estudio

En este taller, vamos a intentar aplicar lo estudiado en el presente módulo para analizar unos procesos de migración sencillos y algún detalle de las técnicas necesarias (en el caso de técnicas de red y/o servidores específicos, las veremos junto con el resto de contenidos de los materiales de administración y administración avanzada).

Nos plantearemos los siguientes casos de estudio:

- Migración individual de un usuario de escritorio Windows a un sistema GNU/Linux.
- Migración de una pequeña organización que dispone de sistemas Windows y algunos UNIX.
- Migración de un servidor Windows aislado (*standalone*) a un servidor Samba en GNU/Linux.

5.1. Migración individual de un usuario de escritorio Windows a un sistema GNU/Linux

Un usuario se plantea la migración a GNU/Linux [Ray02b]. Primero, se pasará por un periodo de convivencia, de modo que el usuario dispondrá de los dos sistemas, y dedicará cada uno de ellos a una serie de tareas: continuará desarrollando tareas en Windows mientras se familiarizará con el nuevo sistema y encontrará software equivalente, o software nuevo que le permita hacer otras tareas que antes no podía realizar.

La migración es, para un usuario personal, quizás uno de los procesos más complejos; hay que ofrecerle alternativas a lo que ya utiliza comúnmente, de forma que la adaptación no entrañe muchas complejidades extra y pueda adaptarse progresivamente con facilidad al nuevo sistema.

Podemos hacer una instalación dual [Ban01] [Sko03b] del sistema original (un Windows, por ejemplo), junto con el sistema GNU/Linux.

Nota

Linux Hardware Howto:
http://www.tldp.org/HOWTO/HardwareHOWTO/index.html

Un primer paso en una determinada configuración de máquina consistirá en verificar que nuestro hardware sea compatible con Linux [Pri02], ya sea por medio de alguna lista de compatibilidad de hardware o verificándolo con el fabricante, por si fuera necesario adquirir nuevos componentes o configurar

de alguna forma particular el existente. Si desconocemos nuestro hardware, podemos verificarlo o bien en Windows con el "administrador de dispositivos" (en el panel de control), o con algún software de reconocimiento de hardware. Por otra parte, un método recomendable es la utilización de distribuciones GNU/Linux de tipo LiveCD (Knoppix, Ubuntu, Fedora Live), que nos permitirán sin instalación física comprobar el funcionamiento de GNU/Linux en nuestro hardware, ya que el único requisito es la posibilidad de arranque del sistema desde CD/DVD (en algunos casos, se necesitará modificar la configuración BIOS para permitir este arranque). Existen LiveCDs como Knoppix [Knp] con gran soporte hardware para las comprobaciones, así como los de la mayoría de las distribuciones GNU/Linux que suelen ofrecer LiveCDs para la comprobación inicial de funcionamiento (en algunos casos, por ejemplo Ubuntu [Ubu], la instalación completa puede realizarse desde el mismo LiveCD). En todo caso, cabe señalar que la prueba con un LiveCD concreto no nos impide que puedan surgir dificultades en la instalación final, debido o bien a que el LiveCD no es de la misma distribución GNU/Linux que la que finalmente instalaremos, o porque las versiones del sistema y/o aplicaciones no son las mismas.

En cuanto a la instalación física en disco, necesitaremos disponer o bien de espacio libre en disco no particionado, o bien, si estamos en particiones de tipo FAT32/NTFS, podemos liberar espacio con programas que permitan el reajuste del tamaño de las particiones, que permiten recortar la partición existente (un previo *backup* de los datos es evidentemente recomendable). En la actualidad, la mayor parte de las distribuciones soportan diferentes esquemas de particionado del disco y el recorte de particiones, aunque dependiendo de la distribución pueden surgir problemas (se recomienda un *backup* previo siempre al proceso de instalación). En caso de no disponer del espacio suficiente, o de tener particiones con sistemas de ficheros que presenten problemas (por ejemplo, NTFS en algunas distribuciones), habrá que plantearse comprar un nuevo disco duro complementario, que dedicaremos totalmente o en parte a GNU/Linux. Si la máquina dispone de arranque desde USB, una posibilidad es disponer de un disco duro externo USB y realizar la instalación allí. También es muy usada la virtualización mediante máquinas virtuales tipo VirtualBox o Vmware, que nos permite instalar una distribución sobre un PC virtual, aunque en este caso no tendremos una experiencia de instalación con nuestro hardware real.

Una vez completada la revisión del hardware, tendremos que decidir la distribución del sistema GNU/Linux que usaremos (una posibilidad apuntada antes es escoger un LiveCD que nos haya satisfecho, y pasar a la instalación de la distribución). Si el usuario es poco experimentado en GNU/Linux, o tiene conocimientos básicos de informática, es mejor decidirse por alguna de las distribuciones más "amigables" de cara al usuario, como Fedora, Mandriva, Novell SuSe o similares. Cabe destacar las facilidades de Ubuntu en este punto. Si tenemos más conocimientos o estamos tentados a experimentar, podemos probar una distribución Debian. En el caso de las comerciales, la distribución, en

la mayoría de veces, con un hardware compatible (en algunos casos como Red Hat, Ubuntu Server y Novell SuSe, versiones empresariales, las distribuidoras certifican el hardware que soportan), se instala perfectamente sin problemas, y se realizan configuraciones básicas que permiten ya utilizar el operativo. En el proceso tendremos que instalar el software, que vendrá definido por unos conjuntos de software orientados: a servidores, a aplicaciones concretas, o a aplicaciones de escritorio como las ofimáticas, aplicaciones de desarrollo (si nos interesa la programación), etc.

Una vez instalado el sistema, se plantea el tema de la compartición de datos [Gon00] [Kat01]. ¿Cómo compartimos datos entre los dos sistemas? o ¿hay posibilidad de compartir algunas aplicaciones? Para esto hay varias soluciones:

a) Método por "intermediario": consiste en compartir los datos, por ejemplo, mediante disquetes (opción un poco ya obsoleta desde la aparición de dispositivos usb removibles). En el caso de los disquetes, lo mejor son las utilidades denominadas *mtools*, que permiten acceder a disquetes con formato msdos de forma transparente, y ofrecen múltiples comandos de línea que funcionan de forma muy parecida a msdos o Windows. Estos comandos se llaman exactamente como los comandos msdos originales, sólo que con una "m" delante, por ejemplo: *mcd*, *mcopy*, *mdir*, *mdel*, *mformat*, *mtype*, etc.

b) Método directo: consiste en usar directamente los sistemas de ficheros de Windows (ya sean los discos duros internos de la máquina, o unidades removibles externas tipo flash o discos duros mediante usb). Como veremos en la unidad de administración local, GNU/Linux puede leer y escribir una gran cantidad de sistemas de ficheros, entre ellos el FAT, FAT32, y NTFS (sólo lectura en algunos casos de distribuciones antiguas, aunque la mayoría de distribuciones ya incorporan, o lo ofrecen como opción el soporte NTFS por medio del *driver* ntfs-3g [Nt3] que permite la escritura). Se tiene que pasar por un proceso denominado "de montaje", que permite incorporar el sistema de ficheros de Windows a un punto del árbol de archivos de Linux. Por ejemplo, podríamos montar nuestro disco Windows en /mnt/Windows y acceder desde este punto a sus carpetas y archivos, permitiendo escrituras y lecturas. Con los ficheros de texto ASCII, hay que tener en cuenta las conversiones, ya que UNIX y Windows los tratan de modo diferente: en UNIX, el final de línea tiene un sólo carácter, el avance de línea, ASCII 10, mientras que en Windows hay dos, un retorno y un avance de línea, caracteres ASCII 13 y 10 (como detalle curioso en MAC es el ASCII 13). Con lo cual, suele ser habitual que, al leer un fichero ASCII dos/windows, éste contenga caracteres "raros" al final de línea. Hay editores como *emacs* que los tratan de forma transparente y, en todo caso, hay utilidades GNU/Linux que permiten convertirlos de uno a otro formato (con utilidades como *duconv*, *recode*, *dos2UNIX*, *UNIX2dos*).

c) Uso de aplicaciones: existen algunas alternativas para poder ejecutar las aplicaciones (no todas) de msdos y Windows. Para GNU/Linux hay emuladores de msdos como Dosemu o DosBox, y para Windows existe el software de

ejecución Wine. Éste puede ejecutar un gran numero de aplicaciones de Windows (por ejemplo, permite ejecutar algunas versiones de Office e Internet Explorer), pero se continúa mejorando constantemente. Si la ejecución de aplicaciones Windows es imprescindible, nos puede ayudar algún software comercial, que da soporte extra a Wine. Existen, por ejemplo, Win4Lin, CrossOver y en algún caso con soporte especial para juegos como Cedega. Otra posible solución es el uso de las máquinas virtuales; un ejemplo de software de amplio uso es Vmware o VirtualBox, que crea como máquina virtual un PC completo, simulado por software, en el cual se puede instalar un gran número diferente de sistemas operativos. Vmware y VirtualBox está disponible en versiones para Windows y para GNU/Linux, lo que permite tener un GNU/Linux instalado con un Windows ejecutándose virtualmente sobre él, o un Windows con GNU/Linux en virtual. Existen también otras soluciones de máquina virtual libres como QEmu, KVM, Bochs. En otro segmento, las máquinas virtuales, o genéricamente la virtualización, es usada orientada a la creación de servidores virtuales, con soluciones como Vmware *server*, o los proyectos abiertos Xen, OpenVZ, Vserver, en los que es posible hacer coexistir varias máquinas virtuales corriendo sobre un operativo (mediante modificaciones en el *kernel* que soporten esta virtualización), o incluso sobre el hardware directamente (sin operativo completo, solo una pequeña capa software de virtualización de recursos).

Aparte de compartir la información (aplicaciones y/o datos), pueden buscarse aplicaciones GNU/Linux que sustituyan a las originales Windows a medida que el usuario vaya aprendiendo a utilizarlas, y observe que cumplen las funcionalidades esperadas.

Un caso típico seria la *suite* ofimática en que el usuario puede trasladar el uso a OpenOffice, que tiene un alto grado de compatibilidad con los ficheros de Office y un funcionamiento bastante semejante, o bien KOffice (para el escritorio KDE), o GNumeric y AbiWord (para Gnome). En el caso de procesamiento de imágenes tomamos Gimp, con funcionalidades semejantes a Adobe Photoshop. Y multitud de reproductores multimedia: Xine, Mplayer (o también una versión del RealPlayer). En Internet, se pueden encontrar listas de equivalencias de programas entre Windows y GNU/Linux.

Nota

Listas de equivalencias:
http://www.linuxrsp.ru/win-lin-soft/table-eng.html
http://www.linuxeq.com/

5.2. Migración de una pequeña organización que dispone de sistemas Windows y algunos UNIX

Consideremos ahora una organización que tenga máquinas Windows y algunas máquinas UNIX dedicadas a servicios, o a *workstations* y unos usuarios un poco "anárquicos". Por ejemplo, estudiemos la siguiente situación: la organización tiene una pequeña red local de máquinas Windows repartidas por los usuarios, como máquinas de igual a igual en un grupo de trabajo Windows (no hay dominios Windows *server*).

El grupo es variopinto: tenemos máquinas con Windows 98, XP, Vista, 7, pero personalizadas por cada usuario con el software que necesita para su trabajo diario: ya sea Office, navegador, lector de correo o entornos de desarrollo para los programadores de diferentes lenguajes (por ejemplo C, C++, Java).

Se dispone de algunos recursos hardware extras, como varias impresoras conectadas a la red local (aceptan trabajos TCP/IP), y permiten utilizarse desde cualquier punto de la organización. Y existe una máquina compartida, con algunos recursos especiales, como *scanner*, grabadora de CD y directorios compartidos por red, donde los usuarios pueden dejar sus directorios con sus ficheros para procesos de *backup* o para recuperar, por ejemplo, imágenes escaneadas.

También disponemos de varias *workstations*, en este caso Sun Microsystem Sparc, que ejecutan Solaris (UNIX, comercial de Sun). Estas estaciones están dedicadas al desarrollo y a algunas aplicaciones científicas y gráficas. Estas máquinas disponen de servicios de NFS para compartir archivos y NIS+ para manejar la información de los usuarios que se conectan a ellas y que puedan hacerlo desde cualquiera de ellas de forma transparente. Algunas de las máquinas incluyen servicios específicos; hay una destinada al servidor web de la organización y otra destinada al servidor de correo.

Se plantea la posibilidad de realizar una migración a GNU/Linux por intereses de desarrollo de software y por el interés particular de algunos usuarios de disponer de este sistema.

Además, se aprovechará la migración para intentar solucionar algunos problemas: de seguridad (algunos sistemas antiguos Windows no son la mejor forma de compartir archivos) y de uso de la impresora (se quiere restringir porque el gasto en papel y el coste asociado es alto) a unas cuotas más razonables. Por otra parte, se ofrece cierta libertad a los usuarios; no se les obligará a cambiar de sistema, aunque se les hará la sugerencia. Y aprovecharemos la ocasión para comprar hardware nuevo que complemente al existente, por ejemplo, si las estaciones de trabajo están faltas de espacio de disco, lo cual supone limitaciones de espacio para el correo y las cuentas de usuario.

La migración en una organización (aunque sea pequeña) plantea muchas dificultades: tendremos diferentes ambientes de trabajo, hardware y software heterogéneo, y más de una vez, reticencias de los usuarios al cambio.

Después de toda esta pequeña descripción de nuestra organización (en otros casos más complejos, podría llenar varias páginas o ser un documento entero de análisis de la situación presente y propuestas futuras), nos comenzamos a plantear posibilidades para solucionar todo esto:

1) ¿Qué hacemos con las *workstations* actuales? El coste en mantenimiento y licencias de software es elevado. Tenemos que cubrir el mantenimiento de fallos en las estaciones, hardware caro (en este caso, discos SCSI) y ampliaciones de memoria también caras. El coste del sistema operativo y sus actualizaciones también es caro. En este caso, se nos ofrecen dos posibilidades (dependiendo del presupuesto de que dispongamos para el cambio):

- Podemos reducir costes convirtiendo las máquinas a sistemas GNU/Linux. Estos sistemas son de arquitectura Sparc y existen distribuciones que soportan esta arquitectura. Podríamos sustituir los servicios por sus equivalentes GNU/Linux. La sustitución sería prácticamente directa, puesto que ya usamos un sistema UNIX.

- Otra posibilidad sería la eliminación del hardware propietario de Sun y convertir las estaciones en PC potentes con GNU/Linux. Esto simplifica su mantenimiento posterior, aunque tiene un alto coste inicial.

2) ¿Y con el software de las *workstation*? Si las aplicaciones son de desarrollo propio, puede ser suficiente volver a compilarlas o la adaptación simple al nuevo entorno. Si son comerciales, tendremos que ver si la empresa puede proporcionarlas en entornos GNU/Linux, o si podemos encontrar reemplazos con funcionalidad parecida. En el caso de los desarrolladores, sus entornos de lenguajes C, C++ y Java pueden portarse fácilmente. En caso de C y C++, se puede utilizar el compilador GNU *gcc* y existen multitud de IDE para el desarrollo (Eclipse, KDevelop, Anjuta...). En el caso de Java, se puede utilizar el *openJDK* como las implementaciones propias de Sun en GNU/Linux y entornos varios de código abierto (Eclipse de IBM o Netbeans).

3) ¿Y los usuarios? A aquellos que estén interesados en GNU/Linux les podemos instalar equipos duales con Windows y GNU/Linux para que comiencen a probar el sistema y, si quieren, pasar finalmente a un único sistema GNU/Linux. Podemos encontrar dos tipos de usuarios: los puramente ofimáticos necesitarán la *suite*, navegador y correo; esto se les puede ofrecer con un escritorio GNU/Linux como Gnome o KDE y software como OpenOffice, navegador Mozilla/Firefox, y correo Mozilla Thunderbird (o cualquier otro Kmail, Evolution...). La equivalencia es más o menos directa, depende de las ganas que los usuarios tengan de probar y usar el nuevo software. Para los desarrolladores, el cambio puede ser más directo, ya que se les ofrecen muchos más entornos y herramientas flexibles. Podrían pasarse completamente a sistemas GNU/Linux o trabajar directamente con las *workstations*.

4) ¿Y las impresoras? Puede establecerse alguna estación de trabajo como servidor de impresión (ya sea por colas TCP/IP o por servidor Samba) y controlar las impresiones mediante cuotas.

5) ¿La máquina compartida? El hardware compartido puede dejarse en la misma máquina o se puede controlar desde un sistema GNU/Linux. En cuanto al espacio de disco compartido, puede moverse a un servidor Samba que sustituya al actual.

6) ¿Ampliamos el espacio del disco? Dependerá del presupuesto. Podemos mejorar el control mediante un sistema de cuotas que reparta el espacio de una forma equitativa y ponga límites a la saturación.

5.3. Migración de un servidor Windows a un servidor Samba en GNU/Linux

En este caso, describimos el proceso básico necesario para efectuar una posible migración de un servidor Windows que comparte carpetas e impresora a un servidor Samba en un sistema GNU/Linux.

Nota

Esta migración suele conllevar un proceso bastante más extenso que las bases expuestas. Consultad la bibliografía para ver los pasos completos del mismo.

Supongamos una máquina que pertenece a un grupo de trabajo GRUPO, que comparte una impresora llamada PRINTER y que tiene una carpeta compartida DATOS que no es más que el disco D de la máquina. Varios clientes Windows acceden a la carpeta para lectura/escritura, dentro de una red local con direcciones IP 192.168.1.x, donde x será 1 para nuestro servidor Windows, y los clientes tienen otros valores (las redes 192.168.x.x se utilizan a menudo como direcciones para montar redes privadas internas).

En nuestro proceso vamos a construir un servidor Samba, que es el que nos permitirá la ejecución en GNU/Linux del protocolo SMB/CIFS (*server message block / common Internet file system*). Este protocolo permite la interacción del sistema de archivos y de las impresoras por medio de redes en múltiples sistemas operativos. Podemos montar carpetas pertenecientes a Windows en las máquinas GNU/Linux, o bien parte de los archivos de GNU/Linux en Windows, y lo mismo con las impresoras de uno u otro. El servidor está compuesto de dos *daemons* (procesos de sistema) llamados *smbd* y *nmbd* (dependiendo de la versión de Samba también existe un tercero denominado *winbind* que se utiliza en entornos de dominios Windows para unirse o proporcionar servicios de Active Directory).

Gracias al software como Samba, la migración desde entornos Windows es muy flexible y rápida e incluso con mejoras de prestaciones o en rendimiento.

El proceso *smbd* gestiona las peticiones de los clientes hacia los archivos o impresoras compartidos. El *nmbd* gestiona el sistema de nombres de las máquinas y los recursos bajo el protocolo NetBIOS (creado por IBM). Este protocolo es independiente de la red que se usa (actualmente, Microsoft utiliza generalmente en NT/2000/XP/Vista/7 Netbios sobre TCP/IP). El *nmbd* también proporciona servicios WINS, que es el servicio de asignación de nombres, que se ejecuta sobre Windows NT/Server si tenemos una colección de máquinas. Es una especie de combinación de DNS y DHCP para entornos Windows. El proceso es un poco complejo; en resumen, cuando una máquina Windows arranca, o bien tiene una dirección IP estática, o bien dinámica por medio de un servidor DHCP, y además puede que tenga un nombre NetBIOS (el nombre que el usuario asigna a la máquina: en identificación de red). El cliente WINS contacta con el servidor para informar de cuál es su IP. Si una máquina de red pregunta posteriormente por el nombre NetBios, se contacta con el servidor WINS para obtener su dirección IP y se establecen las comunicaciones. El *nmbd* ejecuta este proceso sobre GNU/Linux.

Como cualquier otro servicio de red, no se debería ejecutar sin considerar qué riesgos puede suponer su activación y cómo podemos minimizarlos. Respecto a Samba, hay que tener presentes los temas de seguridad, puesto que estamos abriendo parte de nuestros archivos y las impresoras locales o de la Red. Tendremos que verificar bien las restricciones de comunicación que ponemos para no dar acceso a usuarios o máquinas no deseadas. En este ejemplo básico, no vamos a comentar estos temas. En un caso real, tendríamos que examinar las opciones de seguridad y restringir el acceso sólo a quien realmente deseemos.

En el proceso de migración, primero tendremos que configurar el sistema GNU/Linux para el soporte de Samba. Se necesita el soporte en el *kernel* de los *filesystems* Samba (*smbfs*), que ya viene activado. Hay que añadir que, actualmente, hay un soporte adicional en el *kernel* a través del modulo (cifs), el cual, a partir de la versión del *kernel* 2.6.20, se considera el método por defecto, quedando smbfs en segundo termino. El módulo cifs aporta soporte para nuevas prestaciones relacionadas con el protocolo CIFS (como extensión de SMB). Estos módulos nos permiten, mediante los nombres de sistemas de ficheros "smbfs" y "cifs", realizar operaciones de montaje de sistemas de ficheros Windows en el árbol de directorios de Windows (*mount -t smbfs* o *mount -t cifs*). Aparte de que el soporte *kernel* se decanta hacia el módulo cifs, hay algunas características que pueden necesitar soporte smbfs, con lo cual suele ser habitual disponer de los dos módulos activados en el *kernel*. También hay que destacar la cuestión de la configuración: mientras smbfs basa su operación en la configuración Samba (como veremos en el fichero smb.conf), al módulo cifs se le proporciona la configuración en las operaciones (por ejemplo, en el proceso de montaje mediante *mount*).

En el caso del uso del servidor Samba, además del soporte *kernel*, necesitaremos instalar los paquetes software asociados: habrá que examinar qué paquetes relacionados con Samba hay en la distribución e instalar los que tengan que ver

con el funcionamiento de servidor. Y también, si se quiere, los relacionados con Samba como cliente, en el caso de que deseemos ser clientes de máquinas Windows o testear, desde nuestro GNU/Linux, los recursos compartidos de las máquinas Windows. En una distribución Debian, estos paquetes son: *samba, samba-common, smbclient, smbfs*. También puede ser interesante instalar *swat*, que es una herramienta gráfica basada en web para la administración de los servicios Samba. Para nuestro servidor GNU/Linux de Samba [Woo00], del ejemplo propuesto, tendremos que transferir los contenidos del anterior disco D (donde teníamos nuestro sistema de ficheros compartido) de la máquina original a la nueva máquina y colocar su contenido en algún *path*, por ejemplo, */home/DATOS*, ya sea por copia de *backup*, transferencia ftp, o usando Samba como cliente para transferir los archivos.

En cuanto a la utilización de GNU/Linux como cliente Samba, es bastante sencilla. Mediante el uso de comandos cliente para un uso ocasional de un sistema de ficheros:

1) Montamos un directorio compartido Windows (sea *host* el nombre del servidor Windows), en un punto de montaje predefinido (ya existente):

```
smbmount //host/carpeta /mnt/windows
```

2) Colocamos el acceso a la "carpeta" Windows de la máquina *host* en nuestro directorio local, accediendo en el árbol de directorios a:

```
/mnt/windows
```

3) A continuación, cuando ya no esté en uso, podemos desmontar el recurso con:

```
smbumount /mnt/windows
```

Si no conocemos los recursos compartidos, podemos obtener una lista con:

```
smbclient -L host
```

Y también podemos utilizar *smbclient //host/carpeta*, que es un programa parecido a un cliente ftp.

En caso de querer hacer los sistemas de ficheros disponibles permanentemente, o proporcionar determinadas configuraciones particulares, podemos estudiar el uso de *mount* directamente (las utilidades smbxxxx lo utilizan), ya sea con los sistemas de ficheros (soportados en el *kernel*) smbfs o cifs, teniendo en cuenta los parámetros (autentificación de usuarios/grupos Windows u otros

Nota

Es importante consultar siempre las páginas ***man***, o 'manuales', que acompañen a las órdenes y aplicaciones Samba, o a los ficheros de configuración.

parámetros de servicio) que deberemos aportar dependiendo del caso, y de la configuración Samba preexistente [Ste07], e integrar los puntos de montaje en el arranque del sistema (configuración de */etc/fstab*, entre otros).

En el caso del servidor Samba, una vez tengamos instalado todo el software Samba, tendremos que configurar el servidor por medio de su fichero de configuración. Según la versión (o la distribución), este fichero puede estar en */etc/smb.conf* o bien en */etc/samba/smb.conf*. Las opciones aquí mostradas pertenecen a un Samba 3.x.x instalado sobre Debian. Otras versiones pueden tener algunas modificaciones menores.

Durante la instalación de los paquetes de software, es habitual que se nos pregunten algunos datos sobre su configuración. En el caso de Samba, se nos pregunta por el grupo de trabajo al que se va a servir, y deberemos colocar el mismo nombre del grupo que en Windows. También se nos pregunta si deseamos contraseñas encriptadas, recomendables por seguridad (antes en los Windows 9x se enviaban en texto en bruto, en lo que constituye un claro ejemplo de escasa seguridad y de alta vulnerabilidad del sistema).

A continuación, pasamos a ver el proceso de configuración del fichero *smb.conf*. Este fichero tiene tres apartados principales:

1) *Global* (características básicas de funcionamiento para todos los servicios).

2) *Browser, Name Resolution, Domain Options* (controla lo que otras máquinas ven de nuestros recursos y cómo actuamos, si de cliente o *server* de los recursos compartidos por *samba/cifs*).

3) *Share* (controla qué compartimos).

También podemos examinar el fichero de configuración de Samba desde estas secciones, que son apartados del fichero separados por indicaciones como *[nombre_seccion].* Así aparecen las secciones [global] con las configuraciones correspondientes a los apartados 1 (*Global*) y 2 (*Browser, Name Resolution, Domain Options*) mencionados. En el apartado 3 (*Share*), podemos encontrar diferentes secciones:

a) *[homes]*: una sección especial que nos indica no un servicio, sino una serie de ellos destinados a que cada usuario con directorio personal en Linux pueda ser accesible en la Red mediante Samba, previa autentificación de la máquina cliente, que nos pedirá el correspondiente usuario y contraseña. Un método sencillo de configurar estos servicios en entornos locales de tamaño pequeño/medio es utilizar los mismos usuarios y *passwords* tanto en entorno Windows como Samba sobre Linux. Como veremos más adelante en los materiales de administración avanzada, en entornos mayores, hay determinados métodos que nos permiten tener sincronizados los usuarios entre los dos entornos, o sustituir completamente mediante un servidor Samba Linux los servicios de

directorio tipo ADS (*active directory service*), que nos permitirán controlar todos los usuarios y servicios de la red homogénea de los dos entornos desde el servidor Samba.

b) *[printers]*: otra sección especial, que nos permite proporcionar servicios de impresión desde Samba para todas las impresoras definidas en el sistema. Es una solución rápida, que puede alternativamente hacerse por cada impresora, si solamente deseamos proporcionar algunas concretas o limitar su uso.

c) Por último, aparecerán diferentes secciones dedicadas a cada servicio que demos de alta y configuremos *[resource]* donde definiremos qué recurso proporcionamos y bajo qué características individuales.

Esta configuración podría quedar así:

```
[global]
   workgroup = GRUPO
   server string = Samba Server Version %v
   log file = /var/log/samba/log.%m
   max log size = 50
   cups options = raw

[homes]
   comment = Home Directories
   read only = No
   browseable = No
   browsable = No

[printers]
   comment = All Printers
   path = /var/spool/samba
   printable = Yes
   browseable = Yes

[media]
   path = /mnt/media
   valid users = user1,user2
   read only = No
```

En este ejemplo rápido de configuración se nos proporcionan unos parámetros globales. Se desactivan los *homes* y en cambio se hacen disponibles todas las impresoras en *printers* y un recurso de disco disponible denominado *media*, que es accesible por dos usuarios para lectura y escritura.

Veamos una configuración básica, paso a paso, en el fichero de configuración Samba. En el manual (extenso) de este fichero de configuración Samba pueden verse las opciones disponibles (*man smb.conf*). Editaremos el fichero con algún

editor e iremos viendo algunas de las líneas del fichero (los caracteres '#' o ';' a principio de línea son comentarios, si la línea contiene ';' es un comentario; para habilitar una línea, si es alguna línea opcional de configuración, deberemos editarla y quitar el ';'):

```
workgroup = GRUPO
```

Aquí está indicado el grupo de trabajo Windows del cual las máquinas Windows clientes serán miembros.

```
server string = %h server (Samba %v)
```

Podemos colocar una descripción textual de nuestro servidor. La *h* y la *v* que aparecen son variables de Samba, que hacen referencia al nombre del *host* y a la versión de Samba. Por seguridad, es mejor quitar la *v*, ya que con ello se informa al exterior de qué versión de Samba tenemos. Si hay *bugs* de seguridad conocidos, esto puede aprovecharse.

```
hosts allow = 192.168.1
```

Esta línea puede estar presente o no, y la podemos incluir para habilitar qué *hosts* serán servidos; en este caso, todos los del rango 192.168.1.x.

```
printcap name = /etc/printcap
```

El fichero *printcap* es donde GNU/Linux guarda la definición de las impresoras, y es aquí donde Samba buscará la información acerca de éstas.

```
guest account = nobody
```

Esta es la cuenta de "invitado". Podemos crear una cuenta diferente, o solamente habilitar el acceso a Samba a los usuarios dados de alta en el sistema GNU/Linux (o usuarios que definamos nuevos para sólo Samba).

```
log file = /var/log/samba/log.%m
```

Esta línea nos dice dónde se van a guardar los ficheros del registro de sesión (*logs*) de Samba. Se guarda uno por cada cliente (variable *m* es el nombre del cliente conectado).

```
encrypt passwords = true
```

Es conveniente, por seguridad, usar encriptación de contraseñas (*passwords*) si tenemos máquinas clientes con Windows 98, NT o superiores. Estas contraseñas se guardan en un fichero */etc/samba/smbpasswd*, que se genera para los usuarios de la instalación de Samba. Las contraseñas se pueden cambiar con el

comando *smbpasswd*. También hay una opción llamada *UNIX passwdord sync*, que permite que el cambio sea simultáneo a las dos contraseñas (usuario Samba y usuario Linux).

A continuación, describiremos la sección "Share Definitions":

```
[homes]
```

Estas líneas permiten dar acceso a las cuentas de los usuarios desde las máquinas Windows. Si no lo queremos, añadimos unos ';' al inicio de estas líneas, y las máquinas, al conectarse, verán el nombre *comment*. En principio, la escritura está deshabilitada, y para habilitarla sólo hay que poner "yes" en la opción *writable*.

Cualquier compartición de un directorio concreto (en Samba se suele denominar *partición* a un grupo de datos compartidos), se hará como los ejemplos que aparecen (ved, por ejemplo, la definición de compartir el CD-ROM en las líneas que empiezan por *[cdrom]*). En *path* se coloca la ruta de acceso.

Nota

Ved *man smb.conf*

En nuestro caso, por ejemplo, pondríamos un nombre DATOS a la partición en la ruta */home/DATOS*, donde habíamos copiado el disco D de la máquina original Windows y el *path* donde se puede encontrar, además de un alto grupo de opciones que puede modificar el usuario.

También hay una definición [*profiles*] que permite controlar los perfiles (*profiles*) de los usuarios Windows, o sea, el directorio donde se guarda su configuración de escritorio Windows, el menú de inicio, etc.

El método es parecido para las impresoras: se hace una partición con el nombre de la impresora (el mismo que se haya dado en GNU/Linux) y en el *path* se coloca la dirección de la cola asociada a la impresora (en GNU/Linux la encontramos en: */var/spool/samba/PRINTER*). Utilizaremos la opción *printable = yes* si queremos que se envíen trabajos con Samba. También se puede determinar qué usuarios pueden acceder (*valid users*).

Una vez hechos estos cambios, sólo tenemos que guardarlos y reiniciar Samba para que lea la nueva configuración. En Debian:

```
/etc/init.d/samba restart
```

Ahora, nuestro directorio compartido y la impresora por Samba estarán disponibles, de manera que sirvan a los usuarios sin que éstos noten diferencia alguna respecto a las conexiones anteriores con el servidor Windows.

Antes de utilizar la configuración Samba anterior, también podemos comprobarla con el comando *testparam*, que verificará que la configuración Samba sea correcta, y además nos señalará los errores de configuración cometidos. Con

respecto al reinicio de Samba, dependiendo de la distribución deberemos reiniciar los diferentes servicios, que en las últimas versiones son los daemons/servicios *smb*, *nmb*, y *winbind* (por ejemplo, en Fedora será necesario reiniciar estos tres servicios).

Actividades

1. En la descripción de servicios GNU/Linux, ¿se encuentra a faltar alguna funcionalidad?, ¿qué otro tipo de servicios añadiríais?

2. En el segundo caso de estudio del taller (el de la organización), ¿cómo cambiaríais la infraestructura informática si dispusierais de un presupuesto de coste cero, un presupuesto medio, o un presupuesto alto? Presentad algunas soluciones diferentes a las expuestas.

3. Tecnologías de virtualización como Vmware o VirtualBox son máquinas virtuales por software que permiten instalar operativos sobre un PC virtual. Se puede conseguir una demo en http://www.vmware.com o http://www.virtualbox.org. Probad (en el caso de que dispongáis de una licencia Windows) a instalarla sobre Windows, y después un GNU/Linux sobre el PC virtual (o al revés). ¿Qué ventajas nos aporta este sistema de compartir los operativos? ¿Qué problemas ocasiona?

4. Si disponéis de dos máquinas para instalar un servidor Samba, podéis probar la instalación o configuración del servidor en configuraciones de cliente Samba UNIX-servidor Windows, o cliente Windows-servidor Samba en GNU/Linux. Lo podéis probar también en una sola máquina, utilizando la misma máquina como servidor y cliente Samba.

Bibliografía

[Ban] Banerjee, T. "Linux Installation Strategies HOWTO". *The Linux Documentation Project*.

The Dot Gnu Project. http://www.gnu.org/software/dotgnu/

[Ste07] French, S. "Linux CIFS Client Guide".http://us1.samba.org/samba/ftp/cifs-cvs/linux-cifs-client-guide.pdf

[Gon] Gonzato, G. "From DOS/Windows to Linux HOWTO". *The Linux Documentation Project*.

[Knp] Distribución Knoppix. http://knoppix.org

[Ubn] Distribución Ubuntu. http://www.ubuntu.com

[LPD] LPD. *The Linux Documentation Project*. http://www.tldp.org. Proporciona los Howto's de los diferentes aspectos de un sistema GNU/Linux y un conjunto de manuales más elaborados.

[Mon] Monit. http://www.tildeslash.com/monit/

[Mor 03] Morill, D. (2003). *Configuración de sistemas Linux*. Anaya Multimedia.

Buena referencia de configuración de sistemas Linux, con algunos casos de estudio en diferentes entornos; comenta diferentes distribuciones Debian y Red Hat.

[Nt3] NTFS-3g Project: NTFS-3G Read/Write Driver. http://www.ntfs-3g.org/

[Pri] Pritchard, S. "Linux Hardware HOWTO". *The Linux Documentation Project*.

[Rayb] Raymond, E. S. "The Linux Installation HOWTO". *The Linux Documentation Project*.

[Sam] Samba Project. http://samba.org

[Sama] Samba HOWTO and Reference Guide (Chapter Domain Control).

<http://samba.org/samba/docs/man/Samba-HOWTO-Collection/samba-pdc.html>

[Samb] Samba Guide (Chapter Adding Domain member Servers and Clients).

<http://samba.org/samba/docs/man/Samba-Guide/unixclients.html>

[Skob] Skoric, M. "Linux+WindowsNT mini-HOWTO". *The Linux Documentation Project*.

[Smb] Entrada "Server Message Block" en la Wikipedia. http://en.wikipedia.org/wiki/Server_Message_Block

[Sun 02] Sundaram, R. (2002). "The dosemu HOWTO". *The Linux Documentation Project*.

[War] Ward, I. "Debian and Windows Shared Printing mini-HOWTO". *The Linux Documentation Project*.

Wine Project. http://www.winehq.com/

[Woo] Wood, D. "SMB HOWTO". *The Linux Documentation Project*.

Administración local

Josep Jorba Esteve

PID_00167543

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

Una de las primeras tareas con la que tendrá que enfrentarse el administrador será la gestión de los recursos locales presentes en la máquina. En el presente módulo veremos algunas de estas tareas de administración básicas, y algunos de los aspectos de personalización y rendimiento de los recursos.

Antes de comenzar con los aspectos más prácticos de la administración, revisaremos algunas de las herramientas básicas de que dispondrá el administrador (algunas como los *shell scripts* ya las hemos revisado previamente).

Posteriormente, analizaremos el proceso de arranque de un sistema GNU/Linux, que nos hará comprender la estructura inicial del sistema y su relación con los servicios que éste proporciona.

A continuación, aprenderemos cómo obtener una visión general del estado actual del sistema por medio de los diferentes procedimientos y comandos de que disponemos para evaluar las partes del sistema. De este modo, podremos tomar decisiones de administración si detectamos algún fallo o deficiencia de rendimiento, o la falta de algún recurso.

Nota

La administración local engloba muchas tareas variadas, que quizás sean las más utilizadas por el administrador en su trabajo diario.

Uno de los principales puntos de la administración es la gestión de usuarios, ya que cualquier configuración de la máquina estará destinada a que pueda ser utilizada por éstos. Veremos cómo definir nuevos usuarios al sistema y controlar su nivel de acceso a los recursos.

En cuanto a los periféricos del sistema, como discos e impresoras, disponemos de diferentes posibilidades de gestión, ya sea vía diferentes servidores (caso impresión) o diferentes sistemas de archivos que podemos tratar, así como algunas técnicas de optimización del rendimiento de los discos.

También examinaremos el problema de la actualización del sistema, así como la nueva incorporación de software de aplicación y cómo hacerlo disponible a los usuarios. Asimismo, analizaremos la problemática de ejecutar trabajos temporizados en el sistema.

En el taller final examinaremos la evaluación de estado de una máquina, siguiendo los puntos vistos en este módulo, y llevaremos a cabo algunas de las tareas de administración básicas descritas. En el desarrollo de la unidad comentaremos algunos comandos y, posteriormente, en el taller, veremos algunos de ellos con más detalle en lo que respecta a su funcionamiento y opciones.

1. Herramientas básicas para el administrador

El administrador de sistemas GNU/Linux tiene que enfrentarse, diariamente, a una gran cantidad de tareas. En general, en la filosofía UNIX no suele haber una única herramienta para cada tarea o una sola manera de hacer las cosas. Lo común es que los sistemas UNIX proporcionen una gran cantidad de herramientas más o menos simples para afrontar las diferentes tareas.

Será la combinación de las herramientas básicas, cada una con una tarea muy definida, la que nos dará la posibilidad de solucionar un problema o tarea de administración.

En este apartado veremos diferentes grupos de herramientas, identificaremos algunas de sus funciones básicas y veremos varios ejemplos de sus usos. Comenzaremos por examinar algunos estándares del mundo GNU/Linux, que nos permitirán hallar algunas de las características básicas que esperamos de cualquier distribución de GNU/Linux. Estos estándares, como el LSB (o Linux Standard Base) [Linc] y el FHS (Filesystem Hierarchy Standard) [Key], nos hablan de herramientas que esperamos encontrar disponibles, de una estructura común para el sistema de ficheros, así como de distintas normas que tienen que cumplirse para que una distribución sea considerada un sistema GNU/Linux y mantenga reglas comunes para la compatibilidad entre estos estándares.

En la automatización de tareas de administración suelen utilizarse comandos agrupados en *shell scripts* (también llamados guiones de comandos), mediante lenguaje interpretados por el *shell* (intérprete de comandos) del sistema. En la programación de estos *shell scripts* se nos permite unir los comandos del sistema con estructuras de control de flujo, y así disponer de un entorno de prototipo rápido de herramientas para la automatización de tareas.

Otro esquema habitual es la utilización de herramientas de compilación y depuración de lenguajes de alto nivel (como por ejemplo C). En general, serán utilizadas por el administrador para generar nuevos desarrollos de aplicaciones o herramientas, o para incorporar al sistema aplicaciones que vengan como código fuente y tengan que adaptarse y compilarse.

También analizaremos el uso de algunas herramientas gráficas con respecto a las habituales de la línea de comandos. Estas herramientas suelen facilitar las tareas al administrador, pero su uso es limitado, ya que dependen fuertemente de la distribución de GNU/Linux, o incluso de cada versión. Aun así, hay algunas herramientas útiles que son compartidas entre distribuciones.

Por último, analizaremos un grupo de herramientas imprescindibles para mantener el sistema actualizado, las herramientas de gestión de paquetes. El software servido en la distribución GNU/Linux, o incorporado posteriormente, se suele ofrecer en unidades denominadas paquetes, que incluyen los archivos de un determinado software, más pasos necesarios para la preparación de la instalación, la configuración posterior o, si es el caso, la actualización o desinstalación de un determinado software. Y cada distribución suele aportar software de gestión para mantener las listas de paquetes instalados, o por instalar, así como el control de las versiones existentes, o posibilidades diversas de actualización por medio de diferentes fuentes origen.

1.1. Herramientas gráficas y líneas de comandos

Existe un gran número de herramientas, de las que examinamos una pequeña porción en este y en los siguientes módulos, que como herramientas de administración son proporcionadas por terceros de forma independiente a la distribución o por el mismo distribuidor del sistema GNU/Linux.

Estas herramientas pueden cubrir más o menos aspectos de la administración de una tarea concreta, y presentarse con múltiples interfaces diferentes: ya sean herramientas de línea de comandos con múltiples opciones y/o ficheros de configuración asociados, o herramientas textuales con algún tipo de menús elaborados, o bien herramientas gráficas con interfaces más adecuadas para el manejo de información, o asistentes que automaticen las tareas, o bien interfaces web de administración.

Funcionalidad

Las herramientas gráficas de administración no suelen ofrecer una funcionalidad completa, y es interesante conocer cuáles son los efectos de sus acciones.

Todo esto nos ofrece un gran número de posibilidades de cara a la administración, pero siempre tenemos que valorar su facilidad de uso junto con las prestaciones y los conocimientos que posea el administrador que se dedica a estas tareas.

Las tareas habituales del administrador GNU/Linux pueden pasar por trabajar con diferentes distribuciones (por ejemplo, las que comentaremos, Fedora [Fed] o Debian [Debb], o cualquier otra), o incluso por trabajar con variantes comerciales de otros UNIX. Esto conlleva que tengamos que establecer una cierta manera de trabajar que nos permita realizar las tareas de la misma forma en los diferentes sistemas.

Por esta razón, en los diferentes apartados intentaremos destacar todos aquellos aspectos más comunes, y las técnicas de administración serán realizadas en su mayor parte a bajo nivel, mediante una línea de comandos y/o con edición de ficheros de configuración asociados.

Cualquiera de las distribuciones de GNU/Linux suele aportar herramientas del tipo línea de comandos, textual o, en particular, gráficas, que complementan las anteriores y simplifican, en mayor o menor medida, la administración de las tareas [Sm]. Pero hay que tener en cuenta varias puntualizaciones:

a) Estas herramientas son una interfaz más o menos elaborada de las herramientas básicas de línea de comandos y los correspondientes ficheros de configuración.

b) Normalmente no ofrecen todas las prestaciones o configuraciones que pueden realizarse a bajo nivel.

c) Los errores pueden no gestionarse bien, o simplemente proporcionar mensajes tipo "la tarea no se ha podido realizar".

d) El uso de estas herramientas oculta, a veces completamente, el funcionamiento interno del servicio o tarea. Comprender bien el funcionamiento interno es un conocimiento básico para el administrador, y más si tiene que desarrollar tareas de corrección de errores o optimización de servicios.

e) Estas herramientas son útiles en la mejora de la producción. Una vez que el administrador tiene los conocimientos adecuados, puede gestionar con ellas de forma más eficaz las tareas rutinarias y automatizarlas.

f) O también el caso contrario, la tarea puede ser tan compleja, o necesitar tantos parámetros, o generar tantos datos, que se vuelve imposible controlarla de forma manual. En estos casos, las herramientas de alto nivel pueden ser muy útiles y volver practicables algunas tareas que de otra manera son difíciles de controlar. Por ejemplo, dentro de esta categoría entrarían las herramientas de visualización, monitorización y resumen de actividades o servicios complejos.

g) En la automatización de tareas, estas herramientas (de más alto nivel) pueden no ser las más adecuadas: pueden no haber sido pensadas para los pasos que hay que realizar, o bien hacerlo de una forma no eficaz. Un caso concreto puede ser la creación de usuarios, una herramienta visual puede ser muy atrayente, por la forma de introducir los datos, pero ¿qué sucede cuando en lugar de introducir uno o pocos usuarios queremos introducir una lista de decenas o centenares de éstos? La herramienta, si no está preparada, se vuelve totalmente ineficiente.

h) Por último, los administradores suelen querer personalizar sus tareas utilizando las herramientas que consideran más cómodas y fáciles de adaptar. En este aspecto, suele ser habitual la utilización de las herramientas básicas de bajo nivel y la utilización de *shell scripts* para combinarlas de modo que formen una tarea.

Tenemos que saber valorar estas herramientas extra según la valía que tengan para nuestras tareas.

Podemos dar a estas herramientas un uso casual (o cotidiano), si tenemos los conocimientos suficientes para tratar los errores que puedan producirse, o bien con el objetivo de facilitar algún proceso para el que haya sido pensada la herramienta, pero siempre controlando las tareas que implementamos y el conocimiento técnico subyacente.

1.2. Documentos de estándares

Los estándares, ya sean genéricos del mundo UNIX o particulares de GNU/Linux, nos permiten seguir unos criterios básicos, por los que nos guiamos en el momento de aprender o realizar una tarea, y que nos proporcionan información básica para comenzar nuestro trabajo.

En GNU/Linux podemos encontrarnos con estándares como el FHS (Filesystem Hierarchy Standard) [Linb], que nos explica qué podemos encontrarnos (o dónde buscarlo) en la estructura del sistema de ficheros de nuestro sistema. O el LSB (Linux Standard Base), que nos comenta diferentes componentes que solemos encontrar en los sistemas [Linc].

El estándar FHS

El Filesystem Hierchachy Standard es una herramienta básica para el conocimiento de una distribución, que nos permite conocer la estructura y funcionalidad del sistema de archivos principal del sistema.

Ved FHS en http://www.pathname.com/fhs

En el estándar **FHS** (Filesystem Hierchachy Standard) se describe la estructura en árbol del sistema de ficheros principal (/), donde se especifica la estructura de los directorios y los principales ficheros que contendrán. Este estándar es usado en mayor o menor medida también para los UNIX comerciales, en los cuales al principio hubo muchas diferencias que hicieron que cada fabricante cambiara la estructura a su gusto. El estándar pensado en origen para GNU/Linux se hizo para normalizar esta situación y evitar cambios drásticos. Aun así, el estándar es seguido con diferentes grados; la mayoría de distribuciones siguen en un alto porcentaje el FHS, realizando cambios menores o aportando ficheros o directorios que no existían en el estándar.

Un esquema básico de directorios podría ser:

- ***/bin***: utilidades de base del sistema, normalmente programas empleados por los usuarios, ya sean desde los comandos básicos del sistema (como */bin/ls*, listar directorio), pasando por los shells (*/bin/bash*), etc.

- ***/boot***: archivos necesarios durante el arranque del sistema, por ejemplo la imagen del kernel Linux, en */boot/vmlinuz*.

- ***/dev***: aquí encontramos ficheros especiales que representan los dispositivos posibles en el sistema. El acceso a los periféricos en sistemas UNIX se

hace como si fueran ficheros. Podemos encontrar ficheros como */dev/console, /dev/modem, /dev/mouse, /dev/cdrom, /dev/floppy*... que suelen ser enlaces a dispositivos más específicos del tipo de controlador o interfaz que utilizan los dispositivos, como */dev/mouse*, enlazado a */dev/psaux*, representado un ratón de tipo PS2, o */dev/cdrom* a */dev/hdc*, un CD-ROM que es un dispositivo del segundo conector IDE y máster. Aquí encontramos los dispositivos IDE como */dev/hdx*, los scsi */dev/sdx*... con x variando según el número de dispositivo. Hay que mencionar que las últimas distribuciones, respecto a los dispositivos de disco, soportan en el *kernel* emulación de dispositivos IDE como si fueran scsi; por eso es corriente ver hoy en día todos los discos como dispositivos de tipo */dev/sdx*. Aquí cabe comentar que, en su inicio, este directorio era estático, con los ficheros predefinidos, y/o configurados en determinados momentos. Actualmente, se utilizan técnicas tecnologías dinámicas (como *hotplug*, y principalmente *udev*), que permiten detectar dispositivos y crear los archivos */dev* dinámicamente, bien al inicio del sistema o durante la ejecución, con la inserción de dispositivos removibles.

- ***/etc***: ficheros de configuración. La mayoría de tareas de administración necesitarán examinar o modificar los ficheros contenidos en este directorio. Por ejemplo, */etc/passwd* contiene parte de la información de las cuentas de los usuarios del sistema.

- ***/home***: contiene las cuentas de los usuarios, es decir, los directorios personales de cada usuario.

- ***/lib***: las bibliotecas del sistema, compartidas por los programas de usuario, ya sean estáticas (extensión *.a*) o dinámicas (extensión *.so*). Por ejemplo, la biblioteca C estándar, en ficheros *libc.so* o *libc.a*. También en particular suelen encontrarse los módulos dinámicos del kernel Linux, en */lib/modules*.

- ***/mnt***: punto para montar (comando *mount*) sistemas de ficheros de forma temporal, como */mnt/cdrom*, para montar un disco en el lector de CD-ROM momentáneamente.

- ***/media***: para punto de montaje habitual de dispositivos extraibles.

- ***/opt***: suele colocarse el software añadido al sistema posterior a la instalación. Otra instalación válida es en */usr/local*.

- ***/sbin***: utilidades de base del sistema. Suelen ser comandos reservados al administrador (*root*). Por ejemplo, */sbin/fsck* para verificar el estado de los sistemas de ficheros.

- ***/tmp***: ficheros temporales de las aplicaciones o del propio sistema. Aunque son para la ejecución temporal, entre dos ejecuciones la aplicación/servicio no puede asumir que va a encontrar los anteriores ficheros.

- ***/usr***: diferentes elementos instalados en el sistema. Algún software de sistema más completo se instala aquí, además de complementos multimedia (iconos, imágenes, sonidos, por ejemplo en */usr/share*) y la documentación del sistema (*/usr/share/doc*). También en */usr/local* se suele utilizar para instalar documentación o elementos complementarios.

- ***/var***: ficheros de registro de sesión o de estado (ficheros de tipo log) y/o errores del propio sistema y de diferentes servicios, tanto locales como de red. Por ejemplo, ficheros de sesión en */var/log*, contenido de los mails en */var/spool/mail* o trabajos de impresión en */var/spool/lpd*.

Estos son algunos de los directorios definidos en el FHS para el sistema raíz; luego se especifican algunas subdivisiones, como el contenido de los */usr* y */var* y los ficheros de datos y/o ejecutables típicos que se esperan encontrar como mínimo en los directorios (ved las referencias a los documentos FHS).

Con respecto a las distribuciones, Fedora/Red Hat sigue el estándar FHS muy de cerca. Sólo presenta algunos cambios en los archivos presentes en */usr* y */var*. En */etc* suele existir un directorio por componente configurable, y hay algún directorio especial, como */etc/sysconfig*, donde se encuentra gran parte de la configuración de varios servicios básicos del sistema. En */opt*, */usr/local* no suele existir software instalado, a no ser que el usuario lo instale. Debian, por su parte, sigue el estándar, aunque añade algunos directorios de configuración especiales en */etc*.

Otro estándar en proceso es el LSB (Linux Standard Base) [Linc]. La idea de éste es definir unos niveles de compatibilidad entre las aplicaciones, bibliotecas y utilidades, de manera que sea posible la portabilidad de las aplicaciones entre distribuciones sin demasiados problemas. Además del estándar, proporcionan conjuntos de prueba (tests) para verificar el nivel de compatibilidad. LSB en sí mismo es un recopilatorio de varios estándares aplicados a GNU/Linux.

Nota

Estándares de especificaciones: http://www.linuxfoundation.org/en/Specifications
LSB: http://www.linuxfoundation.org/collaborate/workgroups/lsb

1.3. Documentación del sistema en línea

Uno de los aspectos más importantes para nuestras tareas de administración será disponer de la documentación correcta para nuestro sistema y el software instalado. Hay muchas fuentes de información, entre las que destacaremos las siguientes:

a) man es la ayuda por excelencia. Nos permite consultar el manual de GNU/Linux, que está agrupado en varias secciones, correspondientes a comandos administración, formatos de ficheros, comandos de usuario, llamadas de lenguaje C, etc. Normalmente, para obtener la ayuda asociada, tendremos suficiente con:

```
man comando
```

Cada página describiría el comando junto con sus opciones y aportaría algunos ejemplos de utilización. A veces, puede haber más de una entrada en el manual. Por ejemplo, es posible que haya una llamada C con igual nombre que un comando. En este caso, hay que especificar qué sección quiere mirarse:

```
man n comando
```

siendo n el número de sección (por ejemplo, 2 para las llamadas a sistema, o 3 para las rutinas de la librería C).

Existen también unas cuantas herramientas de exploración de los manuales, por ejemplo *xman* y *tkman*, que mediante interfaz gráfica facilitan el examen de las diferentes secciones, así como índices de los comandos (también los entornos KDE/Gnome permiten en sus ayudas acceder a las páginas man). Otro comando interesante es *apropos*, palabra que nos permitirá localizar páginas man que hablen de un tema determinado (asociado con la palabra buscada).

b) info es otro sistema de ayuda habitual. Es un programa desarrollado por GNU para la documentación de muchas de sus herramientas. Se trata de una herramienta textual en la que los capítulos y las páginas se pueden recorrer por medio de un sistema de navegación simple (basado en teclado).

c) Documentación de las aplicaciones: además de ciertas páginas man, es habitual incluir en las aplicaciones documentación extra, ya sea en forma de manuales o tutoriales, o simples guías de usuario. Estos componentes de documentación se instalan en el directorio */usr/share/doc* (o */usr/doc*, dependiendo de la distribución), donde se crea un directorio por paquete de aplicación (por lo general, la aplicación puede disponer de paquete de documentación por separado).

d) Sistemas propios de las distribuciones. Red Hat suele venir con unos CD de manuales de consulta que son instalables en el sistema y tienen formatos HTML o PDF. Fedora dispone de un proyecto de documentación en su web. Debian trae los manuales como un paquete de software más y suelen instalarse en */usr/doc*. Por otra parte, dispone de herramientas que clasifican la documentación presente en el sistema y la organizan por menús para su visualización, como *dwww* o *dhelp*, que presentan interfaces web para examinar la documentación del sistema.

e) Por último, **los escritorios X**, como Gnome y KDE, también traen sistemas de documentación propios con su documentación y manuales, así como información para desarrolladores, ya sea en forma de ayudas gráficas en sus aplicaciones, o en aplicaciones propias que recopilan las ayudas (por ejemplo, *devhelp* en Gnome).

1.4. Herramientas de gestión de paquetes

En cualquier distribución, los paquetes son el elemento básico para tratar las tareas de instalación de nuevo software, actualización del existente o eliminación del no utilizado.

> Básicamente, un paquete es un conjunto de ficheros que forman una aplicación o una unión de varias aplicaciones relacionadas, formando un único fichero (denominado paquete), con un formato propio y comprimido, que es el que se distribuye, ya sea vía CD, disquete o mediante acceso a servicios de ftp o web.

El uso de paquetes facilita añadir o quitar software, al considerarlo una unidad y no tener que trabajar con los ficheros individuales.

En el contenido de la distribución (sus CD) los paquetes suelen estar agrupados por categorías como: **a)** Base: paquetes indispensables para el funcionamiento del sistema (útiles, programas de inicio, bibliotecas de sistema). **b)** Sistema: útiles de administración, comandos de utilidad. **c)** Desarrollo (*developement*): útiles de programación, como editores, compiladores, depuradores... **d)** Gráficos: controladores e interfaces gráficas, escritorios, gestores de ventanas. **e)** Otras categorías.

Para la instalación de un paquete, será necesario efectuar una serie de pasos:

1) Previo (preinstalación): comprobar que existe el software necesario (y con las versiones correctas) para su funcionamiento (dependencias), ya sean bibliotecas de sistema u otras aplicaciones que sean usadas por el software.

2) Descompresión del contenido del paquete, copiando los ficheros a sus localizaciones definitivas, ya sean absolutas (tendrán una posición fija) o, si se permite, reubicadas a otros directorios.

3) Postinstalación: retocar los ficheros necesarios, configurar posibles parámetros del software, adecuarlo al sistema...

Dependiendo de los tipos de paquetes, estos pasos pueden ser automáticos en su mayoría (así es en el caso de RPM [Bai03] y DEB [Deb02]). También puede que se necesite hacerlos todos manuales (caso TGZ), dependiendo de las herramientas que proporcione la distribución.

Veremos, a continuación, quizás los tres paquetes más clásicos de la mayoría de distribuciones. Cada distribución tiene uno por estándar y soporta alguno de los demás.

1.4.1. Paquetes TGZ

Los paquetes TGZ son quizás los de utilización más antigua. Las primeras distribuciones de GNU/Linux los utilizaban para instalar el software, y todavía los usan varias distribuciones (por ejemplo, Slackware) y algunos UNIX comerciales.

Son una combinación de ficheros unidos por el comando *tar* en un único fichero .tar, que luego ha sido comprimido por la utilidad gzip, suele aparecer con la extensión .tgz o bien .tar.gz. Asimismo, hoy en día es común encontrar los tar.bz2 que utilizan, en lugar de gzip, otra utilidad llamada bzip2 que, en algunos casos, consigue mayor compresión del archivo.

Este tipo de paquete no contiene ningún tipo de información de dependencias, y puede presentar tanto contenido de aplicaciones en formato binario como en código fuente. Podemos considerarlo como una especie de colección de ficheros comprimida.

Nota

Los paquetes TGZ son una herramienta básica a la hora de instalar software no organizado y son muy útiles para realizar procesos de *backup* y restauración de archivos.

En contra de lo que pudiera parecer, este es un formato muy utilizado y, sobre todo, por creadores o distribuidores de software externo a la distribución. Muchos creadores de software que trabajan para plataformas varias, como varios UNIX comerciales y diferentes distribuciones de GNU/Linux, lo prefieren como sistema más sencillo y portable.

Ejemplo

Uno de estos casos es el proyecto GNU, que distribuye su software en este formato (en forma de código fuente), ya que puede utilizarse en cualquier UNIX, ya sea un sistema propietario, una variante BSD o una distribución GNU/Linux.

Si se trata de formato binario, tendremos que tener en cuenta que sea adecuado para nuestro sistema; por ejemplo, suele ser común alguna denominación como la que sigue (en este caso, la versión 1.4 del navegador web Mozilla):

```
mozilla-i686-pc-linux-gnu-1.4-installer.tar.gz
```

donde tenemos el nombre del paquete, como Mozilla, arquitectura a la que ha destinado *i686* (Pentium II o superiores o compatibles). Podría ser *i386*, *i586*, *i686*, *k6* (*amd k6*), *k7* (*amd athlon*), *amd64* u *x86_64* (para AMD64 y algunos intel de 64bits), o *ia64* (intel Itaniums). Otras para arquitecturas de otras máquinas, como sparc, powerpc, mips, hppa, alpha... Después nos indica que es para Linux, en una máquina PC, la versión del software 1.4.

Si fuese en formato fuente, suele aparecer como:

```
mozilla-source-1.4.tar.gz
```

donde se nos indica la palabra *source*. En este caso, no menciona versión de arquitectura de máquina, lo que nos indica que está preparado (en principio) para compilarse en diferentes arquitecturas.

De otro modo, habría diferentes códigos para cada sistema operativo o fuente: *GNU/Linux*, *Solaris*, *Irix*, *bsd*...

El proceso básico con estos paquetes consiste en:

1) Descomprimir el paquete (no suelen utilizar *path* absoluto, con lo que se pueden descomprimir en cualquier directorio):

```
tar -xzvf fichero.tar.gz (o fichero.tgz)
```

Con el comando *tar* ponemos opciones de *x*: extraer ficheros, *z*: descomprimir, *v*: ver pasos proceso, *f*: dar nombre del fichero a tratar.

También se puede hacer por separado (sin la *z* del *tar*):

```
gunzip fichero.tar.gz
```

(nos deja un fichero tar)

```
tar -xvf fichero.tar
```

2) Una vez tenemos descomprimido el *tgz*, tendremos los ficheros que contenía; normalmente, el software debe incluir algún fichero de tipo *Readme* o *Install*, donde nos especificarán las opciones de instalación paso a paso, y también posibles dependencias del software.

En primer lugar, habrá que verificar las dependencias (suelen indicarlas en el fichero con los pasos de instalación), por si disponemos del software adecuado. Si no, deberemos buscarlo e instalarlo.

Si se trata de un paquete binario, la instalación suele ser bastante fácil, ya que o bien directamente ya será ejecutable donde lo hayamos dejado, o traerá algún instalador propio. Otra posibilidad será que tengamos que hacerlo manualmente, con lo que bastará con copiar (*cp -r*, copia recursiva) o mover (comando *mv*) el directorio a la posición deseada.

Otro caso es el formato de código fuente. En éste, antes de instalar el software, tendremos que hacer un paso de compilación. Para eso habrá que leer con cierto detalle las instrucciones que lleve el programa. Pero la mayoría de desarrolladores usan un sistema de GNU llamado *autoconf* (de autoconfiguración), en el que habitualmente se usan los siguientes pasos (si no aparecen errores):

a) *./configure*: se trata de un *script* que configura el código para poder ser compilado en nuestra máquina, y verifica que existan las herramientas adecuadas. La opción *--prefix = directorio* permite especificar dónde se instalará el software. Normalmente, se soportan muchas opciones adicionales de configuración según el software (con *–help* se mostraran las que acepta).

b) *make*: la compilación propiamente dicha.

c) *make install*: la instalación del software a un lugar adecuado, especificado previamente como opción al *configure* o asumida por defecto.

Este es un proceso general, pero depende del software que lo siga o no; hay casos bastante peores, donde todo el proceso se tiene que realizar a mano, retocando ficheros de configuración o el mismo *makefile*, y/o compilando uno a uno los ficheros, pero esto, por suerte, es cada vez menos habitual.

En caso de querer borrar el software instalado, habrá que utilizar el desinstalador si nos lo proporcionan, o si no, borrar directamente el directorio o los ficheros que se instalaron, teniendo cuidado de posibles dependencias.

Los paquetes *tgz* son bastante habituales como mecanismo de *backup* en tareas de administración, por ejemplo, para guardar copias de datos importantes, hacer *backups* de cuentas de usuario, o guardar copias antiguas de datos que no sabemos si volveremos a necesitar. Suele utilizarse el siguiente proceso: supongamos que queremos guardar copia del directorio "dir" *tar -cvf dir.tar dir* (*c:* compactar dir en el fichero dir.tar) *gzip dir.tar* (comprimir) o bien en una sola instrucción como:

```
tar -czvf dir.tgz dir
```

El resultado será un fichero *dir.tgz*. Hay que ser cuidadoso si nos interesa conservar los atributos de los ficheros, y permisos de usuario, así como posiblemente ficheros de enlace (links) que pudieran existir (deberemos examinar las opciones de *tar* para que se ajuste a las opciones de *backup* deseadas).

1.4.2. Fedora/Red Hat: paquetes RPM

El sistema de paquetes RPM [Bai] creado por Red Hat supone un paso adelante, ya que incluye la gestión de dependencias y tareas de configuración del software. Además, el sistema guarda una pequeña base de datos con los paquetes ya instalados, que puede consultarse y se actualiza con las nuevas instalaciones.

Los paquetes RPM, por convención, suelen usar un nombre como:

```
paquete-version-rev.arq.rpm
```

donde *paquete* es el nombre del software, *version* es la numeración de versión del software, *rev* suele ser la revisión del paquete RPM, que indica las veces que se ha construido, y *arq,* la arquitectura a la que va destinado el paquete, ya sea Intel/AMD (i386, i586, i686, x86_64, ia64) u otras como alpha, sparc, ppc... La "arquitectura" *noarch* suele usarse cuando es independiente, por ejemplo, un conjunto de *scripts*, y *src* en el caso de que se trate de paquetes de código fuente. La ejecución típica incluye la ejecución de rpm, con las opciones de la operación a realizar, junto con uno o más nombres de paquetes por procesar juntos.

Ejemplo

El paquete *apache-1.3.19-23.i686.rpm* indicaría que se trata del software Apache (el servidor web), en su versión 1.3.19, revisión del paquete RPM 23, para arquitecturas Pentium II o superiores.

Las operaciones típicas con los paquetes RPM incluyen:

- **Gestión de dependencias**: los paquetes RPM incorporan la idea de gestión de dependencias y de base de datos de los paquetes existentes.

- **Información del paquete**: se consulta sobre el paquete una información determinada, se usa la opción *-q* acompañada del nombre del paquete (con *-p* si se hace sobre un archivo rpm). Si el paquete no ha sido instalado todavía, la opción sería *-q* acompañada de la opción de información que se quiera pedir, y si se quiere preguntar a todos los paquetes instalados a la vez, la opción sería *-qa*. Preguntas a un paquete instalado pueden ser:

Consulta	Opciones RPM	Resultados
Archivos	*rpm -ql*	Lista de los archivos que contiene el paquete (pasado como parámetro)
Información	*rpm -qi*	Descripción del paquete
Requisitos	*rpm -qR*	Requisitos previos, bibliotecas o software

- **Instalación:** simplemente *rpm -i paquete.rpm*, o bien puede hacerse con URL donde encontrar el paquete, para descargarlo desde servidores FTP o web. Sólo hay que utilizar la sintaxis ftp:// o http:// para dar la localización del paquete. La instalación podrá realizarse siempre que se estén

cumpliendo las dependencias del paquete, ya sea software previo o bibliotecas que deberían estar instaladas. En caso de no cumplirlo, se nos listará qué software falta, y el nombre del paquete que lo proporciona. Puede forzarse la instalación (a riesgo de que no funcione) con las opciones *--force* o *--nodeps,* o simplemente no hacer caso de la información de las dependencias. La tarea de instalación (realizada por rpm) de un paquete conlleva diferentes subtareas:

– Verificar las posibles dependencias.

– Examinar por conflictos con otros paquetes previamente instalados.

– Realizar tareas previas a la instalación.

– Decidir qué hacer con los ficheros de configuración asociados al paquete si previamente existían.

– Desempaquetar los ficheros y colocarlos en el sitio correcto.

– Realizar tareas de postinstalación.

– Finalmente, almacenar registro de las tareas realizadas en la base de datos de RPM.

- **Actualización:** equivalente a la instalación, pero comprobando primero que el software ya existe *rpm -U paquete.rpm*. Se encargará de borrar la instalación previa.

- **Verificación:** durante el funcionamiento normal del sistema, muchos de los archivos instalados cambian. En este sentido, RPM permite verificar los archivos para detectar las modificaciones, bien por proceso normal, bien por algún error que podría indicar datos corrompidos. Mediante *rpm -V paquete* verificamos un paquete concreto, y mediante *rpm -Va* los verificamos todos.

- **Eliminación:** borrar el paquete del sistema RPM (*-e* o *--erase*). Si hay dependencias, puede ser necesario eliminar otros primero.

Ejemplo

Para un caso remoto:

```
rpm -i ftp://sitio/directorio/paquete.rpm
```

nos permitiría descargar el paquete desde el sitio ftp o web proporcionado, con su localización de directorios, y proceder en este caso a la instalación del paquete.

Hay que vigilar la procedencia de los paquetes, y sólo utilizar fuentes de paquetes conocidas y fiables, ya sea del propio fabricante de la distribución, o de sitios en los que confiemos. Se nos ofrece junto con los paquetes alguna "firma" digital de éstos para que podamos comprobar su autenticidad; suelen utilizarse las sumas md5 para comprobar que el paquete no se ha alterado, y otros sistemas como GPG (versión Gnu de PGP) para comprobar la autenticidad del emisor del paquete. Hay también en Internet diferentes almacenes de paquetes RPM, donde están disponibles para diferentes distribuciones que usen o permitan el formato RPM.

Nota

Ver el sitio www.rpmfind.net

Para un uso seguro de paquetes, los repositorios (oficiales, y algunos de terceros) firman electrónicamente los paquetes, por ejemplo con el mencionado GPG. Esto nos permite asegurar (si disponemos de las firmas) que los paquetes proceden de la fuente fiable. Cada proveedor (el repositorio) incluye unos ficheros de firma PGP con la clave para su sitio. De los repositorios oficiales ya se encuentran instaladas, de terceros deberemos obtener el fichero de clave, e incluirla en RPM, típicamente:

```
$ rpm --import GPG-KEY-FILE
```

Siendo *GPP-KEY-FILE* el fichero clave GPG o la URL de dicho fichero, este fichero también tendrá suma md5 para comprobar su integridad. Podemos conocer las claves existentes en el sistema con:

```
$ rpm -qa | grep ^gpg-pubkey
```

Podemos observar más detalles a partir de la llave obtenida:

```
$ rpm -qi gpg-key-xxxxx-yyyyy
```

Para un paquete rpm concreto podremos comprobar si dispone de firma y cuál se ha utilizado con:

```
$ rpm --checksig -v <paquete>.rpm
```

Y para verificar que un paquete es correcto en base a las firmas disponibles, puede comprobarse con:

```
$ rpm -K <paquete.rpm>
```

Debemos ser cuidadosos en importar sólo aquellas claves de los sitios en que confiemos. Cuando RPM encuentre paquetes con firma que no poseamos en nuestro sistema, o el paquete no esté firmado, nos avisará, y la acción ya dependerá de nuestra actuación.

En cuanto al soporte RPM en las distribuciones, en Fedora (Red Hat, y también en sus derivadas), RPM es el formato por defecto de paquetes y el que usa ampliamente la distribución para las actualizaciones, y la instalación de software. En Debian se utiliza el formato denominado DEB (como veremos), hay soporte para RPM (existe el comando rpm), pero sólo para consulta o información de paquetes. En el caso de que sea imprescindible instalar un paquete rpm en Debian, se recomienda utilizar la utilidad *alien*, que permite convertir formatos de paquetes, en este caso de RPM a DEB, y proceder a la instalación con el paquete convertido.

Además del sistema base de empaquetado de la distribución, hoy en día cada una suele aportar un sistema de gestión de software intermedio de más alto nivel, que añade una capa superior al sistema base, facilitando las tareas de gestión del software y añadiendo una serie de utilidades para controlar mejor el proceso.

En el caso de Fedora (Red Hat y derivados) se utiliza el sistema YUM, que permite, como herramienta de más alto nivel, la instalación y gestión de paquetes en sistemas rpm, así como la gestión automática de dependencias entre los paquetes. Facilita el acceso a múltiples repositorios diferentes, centraliza su configuración en un fichero (*/etc/yum.conf* habitualmente) y tiene una interfaz de comandos simple.

Nota

Ved http://yum.baseurl.org/

La configuración de yum se basa en:

```
/etc/yum.config | (fichero de opciones)
/etc/yum | (directorio para algunas utilidades asociadas)
/etc/yum.repos.d | (directorio de especificación de repositorios, un fichero para cada uno,
      se incluye información del acceso y localización de las firmas <i>gpg</i>)
```

Para las operaciones típicas de yum, un resumen sería:

Orden	Descripción
yum install <nombre>	Instalar el paquete con el nombre
yum update <nombre>	Actualizar un paquete existente
yum remove <nombre>	Eliminar paquete
yum list <nombre>	Buscar paquete por nombre (sólo nombre)
yum search <nombre>	Buscar más ampliamente
yum provices <file>	Buscar paquetes que proporcionen el fichero
yum update	Actualizar todo el sistema
yum upgrade	Ídem al anterior incluyendo paquetes adicionales

Para finalizar, Fedora tambien ofrece un par de utilidades gráficas para yum, *pup* para controlar las actualizaciones recientes disponibles y *pirut* como paquete de gestión de software (en nuevas versiones de Fedora, se han sustituido por *gpk-update-viewer* y *gpk-application*). También existen algunas otras como *yumex,* con mayor control de la configuración interna de YUM.

1.4.3. Debian: paquetes DEB

Debian tiene herramientas interactivas como *tasksel*, que permiten escoger unos subconjuntos de paquetes agrupados por tipo de tareas: paquetes para X, para desarrollo, para documentación, etc., o como *dselect* que facilita navegar por toda la lista de paquetes disponible (hay miles) y escoger aquellos que queramos instalar o desinstalar. De hecho, estas son sólo un *front-end* del gestor de software nivel intermedio APT (equivalente al yum en Fedora).

En el nivel de línea de comandos dispone de *dpkg,* que es el comando de más bajo nivel (sería el equivalente a rpm), para gestionar directamente los paquetes DEB de software [Deb], típicamente *dpkg -i paquete.deb* para realizar la instalación. Pueden realizarse todo tipo de tareas, de información, instalación, borrado o cambios internos a los paquetes de software.

El nivel intermedio (como el caso de yum en Fedora) lo presentan las herramientas APT (la mayoría son comandos *apt-xxx*). APT permite gestionar los paquetes por medio de una lista de paquetes actuales y disponibles a partir de varias fuentes de software, ya sea desde los propios CD de la instalación, sitios ftp o web (HTTP). Esta gestión se hace de forma transparente, de manera que el sistema es independiente de las fuentes de software.

La configuración del sistema APT se efectúa desde los archivos disponibles en */etc/apt*, donde */etc/apt/sources.list* es la lista de fuentes disponibles. Podría ser, por ejemplo:

```
deb http://http.us.debian.org/debian stable main contrib non-free
debsrc http://http.us.debian.org/debian stable main contrib non-free
deb http://security.debian.org stable/updates main contrib non-free
```

en que hay recopiladas varias de las fuentes "oficiales" para una Debian (*stable* en este caso), desde donde se pueden obtener los paquetes de software, así como las actualizaciones que estén disponibles. Básicamente, se especifica el tipo de fuente (web/ftp en este caso), el sitio, la versión de la distribución (*stable*), y categorías del software que se buscará (libre, o contribuciones de terceros o de licencia no libre o comercial).

Un sistema muy potente

Los paquetes DEB de Debian son quizás el sistema de instalación más potente existente en GNU/Linux. Una de sus prestaciones más destacables es la independencia del sistema de las fuentes de los paquetes (mediante APT).

Los paquetes de software están disponibles para las diferentes versiones de la distribución Debian. Existen paquetes para las versiones *stable*, *testing* y *unstable*. El uso de unos u otros determina el tipo de distribución (previo cambio de las fuentes de repositorios en *sources.list*). Pueden tenerse fuentes de paquetes mezcladas, pero no es muy recomendable, ya que se podrían dar conflictos entre las versiones de las diferentes distribuciones.

Una vez tenemos las fuentes de software configuradas, la principal herramienta para manejarlas en nuestro sistema es *apt-get*, que nos permite instalar, actualizar o borrar desde el paquete individual, hasta actualizar la distribución entera. Existe también un *front-end* a *apt-get*, llamado *aptitude*, cuya interfaz de opciones es prácticamente igual (de hecho, podría calificarse de emulador de *apt-get*, ya que la interfaz es equivalente). Como ventaja, aporta una mejor gestión de dependencias de los paquetes, y algoritmos para solucionar los conflictos de paquetes que pueden aparecer. De hecho, en las últimas versiones de Debian, *aptitude es* la interfaz por defecto en línea de comandos para la gestión de paquetes.

Algunas funciones básicas de *apt-get* son:

1) Instalación de un paquete particular:

```
apt-get install paquete
```

2) Borrado de un paquete:

```
apt-get remove paquete
```

3) Actualización de la lista de paquetes disponibles:

```
apt-get update
```

4) Actualización de la distribución, podríamos efectuar los pasos combinados:

```
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

Mediante este último proceso, podemos mantener nuestra distribución actualizada permanentemente, actualizando los paquetes instalados y verificando las dependencias con los nuevos. Una herramienta útil para construir esta lista es *apt-spy*, que intenta buscar los sitios oficiales más rápidos, o *netselect*, que nos permite probar una lista de sitios. Por otro lado, podemos buscar las fuentes oficiales (configurarlas con *apt-setup*), o bien copiar algún fichero de fuen-

tes disponible. Software adicional (de terceros) puede necesitar añadir otras fuentes más (a */etc/apt/sources.list*), se pueden obtener listas de sitios de fuentes disponibles (por ejemplo, en http://www.apt-get.org).

La actualización del sistema en particular genera una descarga de un gran número de paquetes (en especial en *unstable)*, lo que hace recomendable vaciar la cache, el repositorio local, con los paquetes descargados (se mantienen en */var/cache/apt/archive*) que ya no vayan a ser utilizados, bien con *apt-get clean,* para eliminarlos todos, o bien con *apt-get autoclean*, para eliminar aquellos paquetes no necesarios porque ya hay nuevas versiones y ya no serán necesarios (en principio). Hay que tener en cuenta que no volvamos a necesitar estos paquetes por razones de reinstalación, porque en este caso tendríamos que volver a descargarlos.

El sistema APT también permite lo que se denomina SecureAPT, que es la gestión segura de paquetes mediante verificación de sumas (md5) y la firma de fuentes de paquetes (de tipo GPG). Si durante la descarga no están disponibles las firmas, *apt-get* informa de ello y genera un listado con los paquetes no firmados, pidiendo si se van a dejar instalar o no, dejando la decisión al administrador. Se obtiene la lista de fuentes confiables actuales con:

```
# apt-key list
```

Las claves *gpg* de los sitios oficiales de Debian son distribuidas mediante un paquete. Las instalamos:

```
apt-get install debian-archive-keyring
```

evidentemente considerando que tenemos *sources.list* con los sitios oficiales. Se espera que por defecto (dependiendo de la versión Debian) estas claves ya se instalen, en la instalación inicial del sistema. Para otros sitios no oficiales (que no proporcionen la clave en paquete), pero que consideremos confiables, podemos importar su clave, obteniéndola desde el repositorio (tendremos que consultar dónde tienen la clave disponible, no hay un estándar definido, aunque suele estar en la página web inicial del repositorio). Se utiliza *apt-key add* con el fichero, para añadir la clave, o también:

```
# gpg --import fichero.key
# gpg --export --armor XXXXXXXX | apt-key add -
```

siendo *X* un hexadecimal relacionado con la clave (ved instrucciones del repositorio para comprobar la forma recomendada de importar la clave y los datos necesarios).

Otra funcionalidad importante del sistema APT son las funciones de consulta de información de los paquetes, con la herramienta *apt-cache*, que nos permite interactuar con las listas de paquetes de software Debian.

Ejemplo:

La herramienta *apt-cache* dispone de comandos que nos permiten buscar información sobre los paquetes, como por ejemplo:

1) Buscar paquetes sobre la base de un nombre incompleto:

```
apt-cache search nombre
```

2) Mostrar la descripción del paquete:

```
apt-cache show paquete
```

3) De qué paquetes depende:

```
apt-cache depends paquete
```

Otra herramienta o funcionalidad de *apt* interesante es *apt-show-versions*, que nos especifica qué paquetes pueden ser actualizados (y por qué versiones, ved opción *-u*).

Otras tareas más específicas necesitarán realizarse con la herramienta de más bajo nivel, como *dpkg*. Se puede, por ejemplo, obtener la lista de archivos de un paquete determinado ya instalado:

```
dpkg -L paquete
```

O la lista de paquetes entera con:

```
dpkg -l
```

O buscar de qué paquete proviene un elemento (fichero, por ejemplo):

```
dpkg -S fichero
```

Este en particular funciona para paquetes instalados; *apt-file* permite también buscar para paquetes todavía no instalados.

Por último, cabe mencionar también algunas herramientas gráficas para APT, como *synaptic, gnome*-apt para Gnome (o *gnome-app-install* y *update-manager* para versiones recientes), y *Kpackage* o *Adept* para KDE. O las textuales ya mencionadas, como *aptitude* o *dselect*.

En conclusión, cabe destacar que el sistema de gestión APT (en combinación con el base *dpkg*) es muy flexible y potente a la hora de gestionar las actualizaciones, y es el sistema de gestión de paquetes usado en Debian y sus distribuciones derivadas, como Ubuntu, Kubuntu, Knoppix, Linex, etc.

1.5. Herramientas genéricas de administración

En el campo de la administración, también podríamos considerar algunas herramientas, como las pensadas de forma genérica para la administración. Aunque cabe indicar que para estas herramientas es difícil mantenerse al dia, debido a los actuales planes de versiones de las distribuciones, con evolución muy rápida. Algunas de estas herramientas (aunque en un momento determinado pueden no ser funcionales al completo en una distribución dada) son:

1) Webmin: es una herramienta de administración pensada desde interfaz web. Funciona con una serie de *plugins* que se pueden añadir para cada servicio a administrar y cuenta con formularios donde se especifican los parámetros de configuración de los servicios. Además, ofrece la posibilidad (si se activa) de permitir administración remota desde cualquier máquina con navegador.

Nota

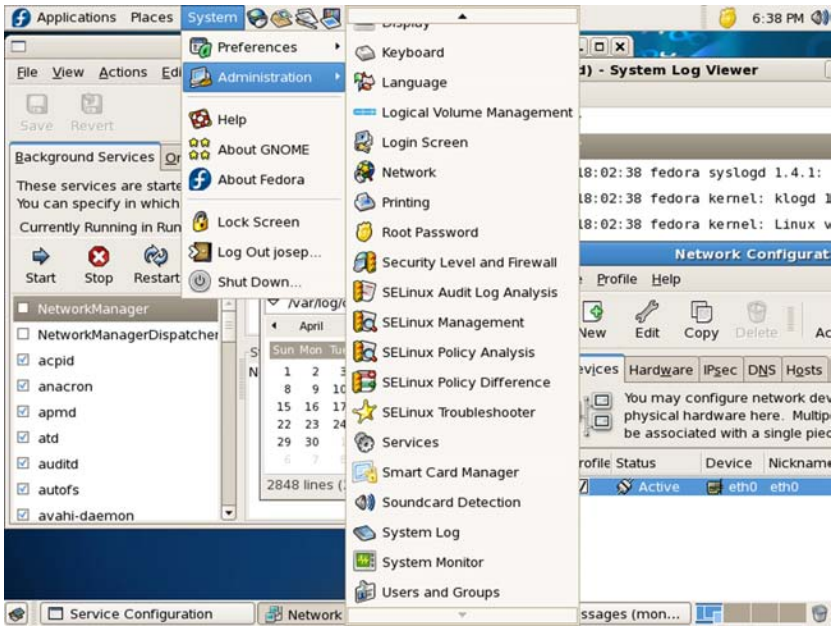
Podéis encontrar esta herramienta en Webmin: www.webmin.com

2) Otras en desarrollo, como **cPanel**, **ISPConfig**...

Por otra parte, en los entornos de escritorio de Gnome y KDE, suelen disponer del concepto de "Panel de control", que permite gestionar tanto el aspecto visual de las interfaces gráficas como tratar algunos parámetros de los dispositivos del sistema.

En cuanto a las herramientas gráficas individuales de administración, la propia distribución de GNU/Linux ofrece algunas directamente (herramientas que acompañan tanto a Gnome como KDE), herramientas dedicadas a gestionar un dispositivo (impresoras, sonido, tarjeta de red, etc.) y otras para la ejecución de tareas concretas (conexión a Internet, configurar arranque de servicios del sistema, configurar X Window, visualizar logs...). Muchas de ellas son simples *frontends* (o carátulas) a las herramientas básicas de sistema, o bien están adaptadas a particularidades de la distribución.

Cabe destacar, en especial en este apartado, la distribución Fedora (Red Hat y derivados), que intenta disponer de distintas utilidades (más o menos minimalistas) para diferentes funciones de administración. Las podemos encontrar en el escritorio (en el menú de administración), o en comandos como *system-config-xxxxx* para diferentes funcionalidades como gestión de pantalla, impresora, red, seguridad, usuarios, paquetes, etc. Podemos ver en la figura algunas de ellas:



Algunas utilidades gráficas de administración en Fedora.

1.6. Otras herramientas

En el espacio limitado de esta unidad no pueden llegar a comentarse todas aquellas herramientas que nos pueden aportar beneficios para la administración. Citaremos algunas de las herramientas que podríamos considerar básicas:

Nota

Consultad las páginas man de los comandos, o una referencia de herramientas como [Stu01].

- **Los múltiples comandos de utilidad UNIX básicos**: *grep*, *awk*, *sed*, *find*, *diff*, *gzip*, *bzip2*, *cut*, *sort*, *df*, *du*, *cat*, *more*, *file*, *which*...

- **Los editores**, imprescindibles para cualquier tarea de edición, cuentan con editores como *Vi*, muy utilizado en tareas de administración por la rapidez de efectuar pequeños cambios en los ficheros. *Vim* es el editor compatible *Vi*, que suele traer GNU/Linux. Permite sintaxis coloreada en varios lenguajes. *Emacs*, editor muy completo, adaptado a diferentes lenguajes de programación (sintaxis y modos de edición), dispone de un entorno muy completo y de una versión *X* denominada *xemacs*. *Joe* es un editor compatible con Wordstar. Y muchos otros...

- **Lenguajes de tipo *script***, útiles para administración, como *Perl*, muy adecuado para tratamiento de expresiones regulares y análisis de ficheros (filtrado, ordenación, etc.); *PHP*, lenguaje muy utilizado en entornos web; *Python*, otro lenguaje que permite hacer prototipos rápidos de aplicaciones...

- **Herramientas de compilación y depuración de lenguajes de alto nivel**: *GNU gcc* (compilador de C y C++ entre otros), *gdb* (depurador), *xxgdb* (interfaz *X* para *gdb*), *ddd* (depurador para varios lenguajes).

2. Distribuciones: particularidades

Intentamos destacar ahora algunas diferencias técnicas menores (que cada vez se reducen más) en las distribuciones (Fedora/Red Hat y Debian) utilizadas [Mor03], que iremos viendo con más detalle a lo largo de las unidades, a medida que vayan apareciendo.

Cambios o particularidades de Fedora/Red Hat:

- **Uso del gestor de arranque *grub*** (una utilidad GNU). A diferencia de pasadas versiones de la mayoría de distribuciones, que suelen usar *lilo*, Fedora utiliza *grub*. GRUB (Grand Unified Bootloader) tiene una configuración en modo texto (normalmente en */boot/grub/grub.conf*) bastante sencilla, y que puede modificarse en el arranque. Es quizás más flexible que *lilo*. Últimamente las distribuciones tienden al uso de *grub*; Debian también lo incluye ya por defecto.

Nota

Es importante conocer los detalles de una distribución, ya que pueden ser básicos para resolver una tarea o acelerar su solución (por ejemplo, si dispone de herramientas propias adicionales especializadas para alguna tarea).

- **Gestión de alternativas**. En el caso de que haya más de un software equivalente presente para una tarea concreta, mediante un directorio (*/etc/alternatives*) se indica cuál es la alternativa que se usa. Este sistema se tomó prestado de Debian, que hace un uso amplio de él en su distribución.

- **Programa de escucha de puertos TCP/IP basado en *xinetd***. En */etc/xinetd.d* podemos encontrar, de forma modular, los ficheros de configuración para algunos de los servicios TCP/IP, junto con el fichero de configuración */etc/xinetd.conf*. En los sistemas UNIX clásicos, el programa utilizado es el *inetd*, que poseía un único fichero de configuración en */etc/inetd.conf*, caso, por ejemplo, de la distribución Debian, que utiliza *inetd*, dejando *xinetd* como opción.

- **Algunos directorios de configuración especiales**: */etc/profile.d*, archivos que se ejecutan cuando un usuario abre un shell; */etc/xinetd.d*, configuración de algunos servicios de red; */etc/sysconfig*, datos de configuración de varios aspectos y servicios del sistema; */etc/cron.*, varios directorios donde se especifican trabajos para hacer periódicamente (mediante *crontab*); */etc/pam.d*, donde *pam* son los denominados módulos de autentificación: en cada uno de cuyos archivos se configuran permisos para el programa o servicio particular; */etc/logrotate.d*, configuración de rotación (cuando hay que limpiar, comprimir, etc.) de algunos de los ficheros de log para diferentes servicios.

- **Dispone de un software llamado *kudzu***, en algunas de las versiones previas de Fedora, y en Read Hat empresarial, que examina el hardware en arranque para detectar posibles cambios de configuración y generar los

dispositivos o configuraciones adecuadas. Aunque se está migrando progresivamente a la API HAL que controla precisamente este tema.

En el caso de Debian:

- **Sistema de empaquetado propio basado en los paquetes DEB**, con herramientas de varios niveles para trabajar con los paquetes como: *dpkg*, *apt-get*, *dselect*, *tasksel*.

- **Debian sigue el FHS**, sobre la estructura de directorios, añadiendo algunos particulares en */etc*, como por ejemplo: */etc/default*, archivos de configuración, y valores por defecto para algunos programas; */etc/network*, datos y guiones de configuración de las interfaces de red; */etc/dpkg* y */etc/apt*, información de la configuración de las herramientas de gestión de paquetes; */etc/alternatives*, enlaces a los programas por defecto, en aquellos en los que hay (o puede haber) varias alternativas disponibles.

- **Sistema de configuración** de muchos paquetes de software por medio de la herramienta *dpkg-reconfigure*. Por ejemplo:

```
dpkg-reconfigure gdm
```

  permite escoger el gestor de entrada para *X*, o:

```
dpkg-reconfigure X-Window-system
```

  nos permite configurar los diferentes elementos de *X*.

- **Utiliza configuración de servicios TCP/IP por *inetd***, configuración en fichero */etc/inetd.conf*. Dispone de una herramienta *update-inetd* para inhabilitar o crear entradas de servicios.

- Algunos directorios de configuración especiales: */etc/cron.*, varios directorios donde se especifican trabajos para hacer periódicamente (mediante *crontab*); */etc/pam.d*, donde *pam* son módulos de autentificación.

3. Niveles de arranque y servicios

Un primer punto importante en el análisis del comportamiento local del sistema es su funcionamiento en los llamados niveles de ejecución (o *runlevels*), que determinan (en el nivel) el modo actual de trabajo del sistema y los servicios que se proporcionan [Wm02].

Un servicio es una funcionalidad proporcionada por la máquina, normalmente basada en *daemonds* (o procesos en segundo plano de ejecución, que controlan peticiones de red, actividad del hardware, u otros programas que provean alguna tarea).

La activación o parada de servicios se realiza mediante la utilización de *scripts*. La mayoría de los servicios estándar, que suelen tener su configuración en el directorio */etc*, suelen controlarse mediante los *scripts* presentes en */etc/init.d/* . En este directorio suelen aparecer *scripts* con nombres similares al servicio donde van destinados, y se suelen aceptar parámetros de activación o parada. Se realiza:

```
/etc/init.d/servicio start
```

Arranque del servicio.

```
/etc/init.d/servicio stop
```

Parada del servicio.

```
/etc/init.d/servicio restart
```

Parada y posterior arranque del servicio.

Cuando un sistema GNU/Linux arranca, primero se carga el *kernel* del sistema, después se inicia el primer proceso, denominado *init*, que es el responsable de ejecutar y activar el resto del sistema, mediante la gestión de los niveles de ejecución (o *runlevels*).

Un nivel de ejecución es sencillamente una configuración de programas y servicios que se ejecutarán orientados a un determinado funcionamiento.

Los niveles típicos suelen ser (puede haber diferencias en el orden, en especial en los niveles 2-5, en la tabla la configuración en Fedora, y la recomendada por el standard LSB):

Runlevel	Función	Descripción
0	Parada	Finaliza servicios y programas activos, así como desmonta *filesystems* activos y para la CPU.
1	Monousuario	Finaliza la mayoría de servicios, permitiendo sólo la entrada del administrador (*root*). Se usa para tareas de mantenimiento y corrección de errores críticos.
2	Multiusuario sin red	No se inician servicios de red, permitiendo sólo entradas locales en el sistema.
3	Multiusuario	Inicia todos los servicios excepto los gráficos asociados a X Window.
4	Multiusuario	No suele usarse, típicamente es igual que el 3.
5	Multiusuario X	Igual que 3, pero con soporte X para la entrada de usuarios (*login* gráfico).
6	Reinicio	Para todos los programas y servicios. Reinicia el sistema.

Por el contrario, cabe señalar que Debian usa un modelo donde prácticamente los niveles 2-5 son equivalentes, realizando exactamente la misma función (aunque podría ocurrir que en alguna versión esto fuera a cambiar para coincidir con el estándar LSB).

Estos niveles suelen estar configurados en los sistemas GNU/Linux (y UNIX) por dos sistemas diferentes, el BSD, o el SystemV (a veces abreviado como sysV). En el caso de Fedora y Debian, se utiliza el sistema SystemV, que es el que mostraremos, pero otros UNIX y alguna distribución GNU/Linux (como Slackware) utilizan el modelo BSD.

En el caso del modelo runlevel de SystemV, cuando el proceso *init* arranca, utiliza un fichero de configuración llamado */etc/inittab* para decidir el modo de ejecución en el que va a entrar. En este fichero se define el runlevel por defecto (*initdefault*) en arranque (por instalación en Fedora el 5, en Debian el 2) y una serie de servicios de terminal por activar para atender la entrada del usuario.

Después, el sistema, según el runlevel escogido, consulta los ficheros contenidos en */etc/rcn.d*, donde *n* es el número asociado al *runlevel* (nivel escogido), en el que se encuentra una lista de servicios por activar o parar en caso de que arranquemos en el *runlevel* o lo abandonemos. Dentro del directorio encontraremos una serie de *scripts* o enlaces a los *scripts* que controlan el servicio.

Cada *script* posee un nombre relacionado con el servicio, una *S* o *K* inicial que indica si es el *script* para iniciar *(S)* o matar *(K)* el servicio, y un número que refleja el orden en que se ejecutarán los servicios.

Una serie de comandos de sistema sirven de ayuda para manejar los niveles de ejecución. Cabe mencionar:

- **Los *scripts***, que ya hemos visto, en */etc/init.d/* nos permiten arrancar, parar o reiniciar servicios individuales.

- ***telinit*** nos deja cambiar de nivel de ejecución; sólo tenemos que indicar el número. Por ejemplo, necesitamos hacer una tarea crítica en *root*; sin usuarios trabajando, podemos hacer un *telinit 1* (también puede usarse *S*) para pasar a *runlevel* monousuario, y después de la tarea un *telinit 3* para volver a multiusuario. También puede utilizarse el comando *init* para la misma tarea, aunque *telinit* aporta algún parámetro extra. Por ejemplo, el reinicio típico de un sistema UNIX se hacía con *sync; sync; sync; init 6*. El comando *sync* fuerza el vaciado de los *buffers* del sistema de archivos, y luego reiniciamos en *runlevel* 6.

- ***shutdown*** permite parar (*'-h'* de *halt*) o reiniciar el sistema (*'-r'* de *reboot*). Puede darse un intervalo de tiempo para hacerse, o bien realizarse inmediatamente. Para estas tareas también existen los comandos *halt* y *reboot*.

- ***wall*** permite enviar mensajes de advertencia a los usuarios del sistema. Concretamente, el administrador puede anunciar que se va a parar la máquina en un determinado momento. Comandos como *shutdown* suelen utilizarlo de forma automática.

- ***pidof*** permite averiguar el PID (*process* ID) asociado a un proceso. Con *ps* obtenemos los listados de procesos, y si queremos eliminar un servicio o proceso mediante *kill*, necesitaremos su PID.

Respecto a todo el modelo de arranque, las distribuciones presentan algún pequeño cambio:

- **Fedora/Red Hat**: el *runlevel* 4 no tiene un uso declarado. Los directorios */etc/rcn.d* existen como enlaces hacia subdirectorios de */etc/rc.d*, donde están centralizados los *scripts* de arranque. Los directorios son, así: */etc/rc.d/rcn.d*; pero como existen los enlaces, es transparente al usuario. El *runlevel* por defecto es el 5 con arranque con *X*.
  Los comandos y ficheros relacionados con el arranque del sistema están en los paquetes de software sysvinit y initscripts.
  Respecto a los cambios de ficheros y guiones en Fedora, cabe destacar: en */etc/sysconfig* podemos encontrar archivos que especifican valores por defecto de la configuración de dispositivos o servicios. El guión */etc/rc.d/rc.sysinit* es invocado una vez cuando el sistema arranca; el guión */etc/rc.d/rc.local* se invoca al final del proceso de carga y sirve para indicar inicializaciones específicas de la máquina que nos interesen sin pasar por todo el sistema siguiente.

El arranque real de los servicios se hace por medio de los guiones almacenados en */etc/rc.d/init.d*. Hay también un enlace desde */etc/init.d*. Además, Fedora proporciona unos *scripts* de utilidad para manejar servicios: */sbin/service* para parar o iniciar un servicio por el nombre y */sbin/chkconfig* para añadir enlaces a los ficheros *S* y *K* necesarios para un servicio, o la obtención de información sobre los servicios.

- **Debian** dispone de comandos de gestión de los *runlevels* como *update-rc.d*, que permite instalar o borrar servicios arrancándolos o parándolos en uno o más runlevels. Otro comando es *invoke-rc.d*, que permite las clásicas acciones de arrancar, parar o reiniciar el servicio.
  El *runlevel* por defecto en Debian es el 2, el X Window System no se gestiona desde */etc/inittab*, sino que existe el gestor (por ejemplo, *gdm* o *kdm*), como si fuera un servicio más del *runlevel* 2.

3.1. Upstart, un nuevo sistema

Upstart es un nuevo sistema de arranque basado en eventos, diseñado para sustituir al proceso comentado generado por el *init* del sistema de arranque *sysvinit*. Uno de los principales problemas del *daemon init* de SystemV es que no fue pensado para hardware moderno, que permite dispositivos removibles, o dispositivos que puedan ser conectados/sustituidos en caliente (*hotplug*).

Este sistema fue diseñado por la empresa Canonical, para incorporarse en Ubuntu 6.10 (2006) y ha sido incorporado posteriormente a las diferentes versiones de Ubuntu, y a Fedora (a partir de la versión 9) y openSUSE (11.3). Debian mantiene el anterior sistema con ciertas modificaciones (*initng*, una alternativa a Upstart) hasta las últimas versiones, pero migrará progresivamente al nuevo sistema (debido a los intentos, por ambas partes, por mantener el máximo nivel de semejanza con Ubuntu y Debian).

Aunque cabe destacar que Upstart (al menos de momento) mantiene una compatibilidad total con los *scripts* iniciales del proceso de *init* (vistos en este apartado, "Niveles de arranque y servicios"), lo que se modifica es el proceso interno que gestiona los servicios de arranque, que pasa a estar controlado por el *daemon* de Upstart.

El demonio *init* de Upstart está basado en eventos que permiten ejecutar programas (o acciones) específicas a partir de cambios en el sistema, que suelen ser típicamente (respecto al *init* SystemV) *scripts* de parada o arranque de servicios. En System V esto solamente se producía por cambio de nivel de ejecución; en Upstart puede provocarlo cualquier evento que suceda dinámicamente en el sistema. Por ejemplo, Upstart suele comunicarse (por eventos) con demo-

nios (como *udev*) que controlan los cambios del hardware de la máquina, por ejemplo si aparece/desaparece un determinado hardware, pueden activarse o desactivarse los servicios de sistema asociados a él.

La gestión de estos eventos está centralizada en */etc/event.d* y, a medida que se vaya migrando (en las distribuciones) hacia upstart, este directorio irá remplazando los contenidos de */etc/init.d* y los directorios asociados a los niveles de ejecución */etc/rc<n>.d*.

En la terminologia de Upstart, un evento es un cambio de estado del sistema que puede ser informado al proceso *init*. Un *job* es un conjunto de instrucciones que *init* lee, típicamente o bien un binario o un *shell script*, junto con el nombre del evento. Así, cuando el evento aparece en el sistema, Upstart lanza el *job* asociado (los *jobs* del sistema podemos encontrarlos definidos en el directorio mencionado */etc/event.d*). Los *jobs* pueden ser de dos tipos, *task* o *service*. Una *task* es un *job* que realiza su tarea y retorna a un estado de espera cuando acaba; un *service* no termina por sí mismo, sino por una decisión basada en un evento o bien una intervención manual de parada.

Así, el proceso general *init* de Upstart es un funcionamiento de máquina de estados; mantiene control del estado de los trabajos (*jobs*) y los eventos que dinámicamente aparecen, y gestiona los cambios de estado de los trabajos en reacción a los eventos aparecidos.

Hay una serie de trabajos (*jobs*) predefinidos con nombres *rcn* que sirven para emular los cambios de *runlevel* de SystemV, para así emular el concepto de nivel de ejecución.

Para examinar el sistema Upstart, podemos usar como herramienta básica el comando *initctl,* que nos permite (entre otras tareas):

- initctl list (listar *jobs* y su estado).
- initctl emit EVENT (emitir manualmente eventos concretos).
- initctl start/stop/status JOB (la acción sobre un *job*).

Para finalizar, podemos observar que Upstart es un sistema bastante nuevo y podrá tener cierta evolución futura, ya sustituyendo completamente al System V inicial o evolucionando a nuevas funcionalidades, pues el sistema basado en máquina de estados es muy completo para integrar diferentes funcionalidades que ahora están repartidas por varios elementos del sistema. Se prevee que podría llegar a reemplazar programación de tareas periódicas de sistema (la usada actualmente, como veremos, por elementos como *cron*, *anacron*, *atd*) y posiblemente gestiones varias de demonios de red (como las realizadas por *inetd* o *xinetd*).

Upstart es un aspecto al que deberemos prestar atención en futuras evoluciones, sin descuidar SystemV, ya que se mantiene cierta compatibilidad al no estar todas las distribuciones migradas a Upstart. Por otra parte, SystemV sigue utilizándose en la mayoría de UNIX propietarios, y versiones empresariales de GNU/Linux.

4. Observar el estado del sistema

Una de las principales tareas del administrador (*root*) en su día a día será verificar el correcto funcionamiento del sistema y vigilar la existencia de posibles errores o de saturación de los recursos de la máquina (memoria, discos, etc.). Pasaremos a detallar, en los siguientes apartados, los métodos básicos para examinar el estado del sistema en un determinado momento y llevar a cabo las acciones necesarias para evitar problemas posteriores.

En el taller final de esta unidad, realizaremos un examen de un sistema ejemplo, para que podáis ver algunas de estas técnicas.

4.1. Arranque del sistema

En el arranque de un sistema GNU/Linux se produce todo un volcado de información interesante. Cuando el sistema arranca, suelen aparecer los datos de detección de las características de la máquina, detección de dispositivos, arranque de servicios de sistema, etc., y se mencionan los problemas aparecidos.

En la mayoría de las distribuciones, esto puede verse en la consola del sistema directamente durante el proceso de arranque. Sin embargo, o la velocidad de los mensajes o algunas modernas distribuciones que los ocultan tras carátulas gráficas pueden impedir seguir los mensajes correctamente, con lo que necesitaremos una serie de herramientas para este proceso.

Básicamente, podemos utilizar:

- **Comando dmesg**: da los mensajes del último arranque del *kernel*.
- **Fichero */var/log/messages***: log general del sistema, que contiene los mensajes generados por el *kernel* y otros *daemons* (puede haber multitud de archivos diferentes de log, normalmente en */var/log*, y dependiendo de la configuración del servicio *syslog*).
- **Comando *uptime***: indica cuánto tiempo hace que el sistema está activo.
- **Sistema */proc***: pseudosistema de ficheros (*procfs*) que utiliza el *kernel* para almacenar la información de procesos y de sistema.
- **Sistema */sys***: pseudosistema de ficheros (*sysfs*) que apareció con la rama 2.6.x del *kernel*, con objetivo de proporcionar una forma más coherente de acceder a la información de los dispositivos y sus controladores (*drivers*).

4.2. Kernel: directorio */proc*

El *kernel*, durante su arranque, pone en funcionamiento un pseudosistema de dicheros llamado */proc*, donde vuelca la información que recopila de la máquina, así como muchos de sus datos internos, durante la ejecución. El directorio */proc* está implementado sobre memoria y no se guarda en disco. Los datos contenidos son tanto de naturaleza estática como dinámica (varían durante la ejecución).

Hay que tener en cuenta que al ser */proc* fuertemente dependiente del *kernel*, propicia que su estructura dependa del *kernel* de que dispone el sistema y la estructura y los ficheros incluidos pueden cambiar.

Una de las características interesantes es que en el directorio */proc* podremos encontrar las imágenes de los procesos en ejecución, junto con la información que el *kernel* maneja acerca de ellos. Cada proceso del sistema se puede encontrar en el directorio */proc/<pidproceso>,* donde hay un directorio con ficheros que representan su estado. Esta información es básica para programas de depuración, o bien para los propios comandos del sistema, como *ps* o *top,* que pueden utilizarla para ver el estado de los procesos. En general, muchas de las utilidades del sistema consultan la información dinámica del sistema desde */proc (*en especial, algunas utilidades proporcionadas en el paquete *procps)*.

Nota

El directorio */proc* es un recurso extraordinario para obtener información de bajo nivel sobre el funcionamiento del sistema. Muchos comandos de sistema se apoyan en él para sus tareas.

Por otra parte, en */proc* podemos encontrar otros ficheros de estado global del sistema. Comentamos de forma breve, algunos ficheros que podremos examinar para obtener información importante:

Fichero	Descripción
/proc/bus	Directorio con información de los buses PCI y USB
/proc/cmdline	Línea de arranque del *kernel*
/proc/cpuinfo	Información de la CPU
/proc/devices	Listado de dispositivos del sistema de caracteres o bloques
/proc/driver	Información de algunos módulos *kernel* de hardware
/proc/filesystems	Sistemas de ficheros habilitados en el *kernel*
/proc/ide	Directorio de información del bus IDE, características de discos
/proc/interrups	Mapa de interrupciones hardware (IRQ) utilizadas
/proc/ioports	Puertos de E/S utilizados
/proc/meminfo	Datos del uso de la memoria
/proc/modules	Módulos del *kernel*
/proc/mounts	Sistemas de archivos montados actualmente

Fichero	Descripción
/proc/net	Directorio con toda la información de red
/proc/scsi	Directorio de dispositivos SCSI, o IDE emulados por SCSI
/proc/sys	Acceso a parámetros del *kernel* configurables dinámicamente
/proc/version	Versión y fecha del *kernel*

A partir de la rama 2.6 del *kernel*, se ha iniciado una transición progresiva de *procfs* (*/proc)* a *sysfs* (*/sys*) con el objetivo de mover toda aquella información que no esté relacionada con procesos, en especial dispositivos y sus controladores (módulos del kernel) hacia el sistema */sys*.

4.3. *Kernel*: */sys*

El sistema Sys se encarga de hacer disponible la información de dispositivos y controladores, información de la cual dispone el *kernel*, al espacio de usuario, de manera que otras API o aplicaciones puedan acceder de una forma flexible a la información de los dispositivos (o sus controladores). Suele ser utilizada por capas como HAL y el servicio *udev* para la monitorización y configuración dinámica de los dispositivos.

Dentro del concepto de *sys* existe una estructura de datos en árbol de los dispositivos y controladores (digamos el modelo conceptual fijo), y después se accede a él por medio del sistema de ficheros *sysfs* (cuya estructura puede cambiar entre versiones).

En cuanto se detecta o aparece en el sistema un objeto añadido, en el árbol del modelo de controladores (controladores, dispositivos incluyendo sus diferentes clases), se crea un directorio en *sysfs*. La relación padre/hijo se refleja con subdirectorios bajo */sys/devices/* (reflejando la capa física y sus identificadores). En el subdirectorio */sys/bus* se colocan enlaces simbólicos, reflejando el modo en el que los dispositivos pertenecen a los diferentes buses físicos del sistema. Y en */sys/class* muestra los dispositivos agrupados de acuerdo a su clase, como por ejemplo red, mientras que */sys/block/* contiene los dispositivos de bloques.

Alguna de la información proporcionada por */sys* puede encontrarse también en */proc*, pero se consideró que éste estaba mezclando diferentes cosas (dispositivos, procesos, datos hardware, parámetros *kernel*) de forma no coherente, y ello fue uno de los motivos para crear */sys*. Se espera que, progresivamente, se migre información de */proc* a */sys* para centralizar la información de los dispositivos.

4.4. Procesos

Los procesos que se encuentren en ejecución en un determinado momento serán, en general, de diferente naturaleza. Podemos encontrar:

- **Procesos de sistema**, ya sean procesos asociados al funcionamiento local de la máquina, *kernel*, o bien procesos (denominados *daemons*) asociados al control de diferentes servicios. Por otro lado pueden ser locales, o de red, debido a que estemos ofreciendo el servicio (actuamos de servidor) o estemos recibiendo los resultados del servicio (actuamos de clientes). La mayoría de estos procesos de sistema aparecerán asociados al usuario *root* (aunque se suelen migrar a pseudousuarios especializados por servicio), aunque no estemos presentes en ese momento como usuarios. Puede haber algunos servicios asociados a otros usuarios de sistema (lp, bin, www, mail, etc.). Estos son pseudousuarios "virtuales", no interactivos, que utiliza el sistema para ejecutar ciertos procesos.

- **Procesos del usuario administrador**: en caso de actuar como *root*, nuestros procesos interactivos o aplicaciones lanzadas también aparecerán como procesos asociados al usuario *root*.

- **Procesos de usuarios del sistema**: asociados a la ejecución de sus aplicaciones, ya sea tareas interactivas en modo texto o en modo gráfico.

Como comandos rápidos y más útiles, podemos utilizar:

- ***ps***: el comando estándar, lista los procesos con sus datos de usuario, tiempo, identificador de proceso y línea de comandos usada. Una de las opciones más utilizada es *ps -ef* (o *-ax*), pero hay muchas opciones disponibles (ved man).

- ***top***: una versión que nos da una lista actualizada a intervalos, monitorizando dinámicamente los cambios. Y nos permite ordenar el listado de procesos por diferentes ítems, como gasto de memoria, de uso CPU, con propósito de obtener un ranking de los procesos que acaparan los recursos. Muy útil para dar indicios en situaciones extremas de saturación de uso de recursos, de la posible fuente de problemas.

- ***kill***: nos permite eliminar procesos del sistema mediante el envío de señales al proceso como, por ejemplo, la de terminación *kill -9 pid_del_proceso* (*9* corresponde a *SIGKILL*), donde indicamos el identificador del proceso. Resulta útil para procesos con comportamiento inestable o programas interactivos que han dejado de responder. Podemos ver una lista de las señales válidas en el sistema con *man 7 signal*.

4.5. Logs del sistema

Tanto el *kernel* como muchos de los *daemons* de servicios, así como diferentes aplicaciones o subsistemas de GNU/Linux, pueden generar mensajes que vayan a parar a ficheros log, ya sea para tener una traza de su funcionamiento, o bien para detectar errores o advertencias de mal funcionamiento o situaciones críticas. Este tipo de logs son imprescindibles, en muchos casos, para las tareas de administración y se suelen emplear bastante tiempo de administración en el procesamiento y análisis de sus contenidos.

La mayor parte de los logs se generan en el directorio */var/log*, aunque algunas aplicaciones pueden modificar este comportamiento. La mayoría de logs del propio sistema sí que se encuentran en este directorio.

Un *daemon* particular del sistema (importante) es el *daemon Syslogd*, que se encarga de recibir los mensajes que se envían por parte del *kernel* y otros *daemons* de servicios y los envía a un fichero log que se encuentra en */var/log/messages*. Este es el fichero por defecto, pero *Syslogd* es también configurable (en el fichero */etc/syslog.conf*), de manera que se pueden generar otros ficheros dependiendo de la fuente, según el *daemon* que envía el mensaje, y así dirigirlo a un log u a otro (clasificando así por fuente), y/o también clasificar los mensajes por importancia (nivel de prioridad): *alarm, warning, error, critical*, etc.

Nota

El *daemon Syslogd* es el servicio más importante de obtención de información dinámica de la máquina. El proceso de análisis de los logs nos ayuda a entender el funcionamiento, los posibles errores y el rendimiento del sistema.

Dependiendo de la distribución, puede estar configurado de diferentes modos por defecto, en */var/log* suele generar (por ejemplo) en Debian ficheros como: *kern.log, mail.err, mail.info*... que son los logs de diferentes servicios. Podemos examinar la configuración para determinar de dónde provienen los mensajes y en qué ficheros los guarda. Una opción que suele ser útil es la posibilidad de enviar los mensajes a una consola virtual de texto (en */etc/syslog.conf* se especifica para el tipo, o tipos, de mensaje una consola de destino, como */dev/tty8* o */dev/xconsole*), de manera que podremos ir viendo los mensajes a medida que se produzcan. Esto suele ser útil para monitorizar la ejecución del sistema sin tener que estar mirando los ficheros de log a cada momento. Una modificación simple de este método podría ser introducir, desde un terminal, la instrucción siguiente (para el log general):

```
tail -f /var/log/messages
```

Esta sentencia nos permite dejar el terminal o ventana de terminal, de manera que irán apareciendo los cambios que se produzcan en el fichero.

Otros comandos relacionados son:

- ***uptime***: tiempo que hace que el sistema está activo. Útil para comprobar que no hay existido algún rearranque del sistema inesperado.
- ***last***: analiza log de entradas/salidas del sistema (*/var/log/wtmp*) de los usuarios, y los arranques del sistema. O *lastlog,* control de la última vez que los usuarios han sido vistos en el sistema (información en */var/log/lastlog*).
- **Varias utilidades para procesamiento combinado de logs**, que emiten resúmenes (o alarmas) de lo sucedido en el sistema, como por ejemplo: *logwatch, logcheck(Debian), log_analysis (Debian)...*

4.6. Memoria

Respecto a la memoria del sistema, deberemos tener en cuenta que disponemos de la memoria física de la propia máquina y de la memoria virtual, que puede ser direccionada por los procesos. Normalmente (a no ser que estemos tratando con servidores empresariales), no dispondremos de cantidades demasiado grandes, de modo que la memoria física será menor que el tamaño de memoria virtual necesario (4GB en sistemas de 32 bits). Esto obligará a utilizar una zona de intercambio (*swap*) sobre disco, para implementar los procesos asociados a memoria virtual.

Esta zona de intercambio (*swap*) puede implementarse como un fichero en el sistema de archivos, pero es más habitual encontrarla como una partición de intercambio (llamada de *swap*), creada durante la instalación del sistema. En el momento de particionar el disco, se declara como de tipo Linux Swap.

Para examinar la información sobre memoria, tenemos varios métodos y comandos útiles:

- **Fichero */etc/fstab***: aparece la partición *swap* (si existiese). Con un comando *fdisk* podemos averiguar su tamaño (o consultar a */proc/swaps*).
- **Comando *ps***: permite conocer qué procesos tenemos, y con las opciones de porcentaje y memoria usada.
- **Comando *top***: es una versión ps dinámica actualizable por periodos de tiempo. Puede clasificar los procesos según la memoria que se usa o el tiempo de CPU.
- **Comando *free***: informa sobre el estado global de la memoria. Da también el tamaño de la memoria virtual.
- **Comando *vmstat***: informa sobre el estado de la memoria virtual, y el uso que se le da.
- **Algunos paquetes** como *dstat* permiten recoger datos de los diferentes parámetros (memoria, *swap* y otros) a intervalos de tiempo (de forma parecida a *top*).

4.7. Discos y *filesystems*

Examinaremos qué discos tenemos disponibles, cómo están organizados y de qué particiones y sistemas de archivos (*filesystems*) disponemos.Cuando dispongamos de una partición y de un determinado *filesystem* accesible, tendremos que realizar un proceso de montaje para integrarla en el sistema, ya sea explícitamente o bien programada en arranque. En el proceso de montaje, se conecta el sistema de archivos asociado a la partición a un punto del árbol de directorios.

Para conocer los discos (o dispositivos de almacenamiento) que tenemos en el sistema, podemos basarnos en la información de arranque del sistema (comando *dmesg* o */var/log/messages*), donde se detectan los presentes, como los */dev/hdx* para los dispositivos IDE o los SCSI con dispositivos */dev/sdx*. Últimamente los discos SATA y antiguos IDE son presentados como scsi, debido a una capa de emulación del *kernel* que trata a los discos como scsi. Otros dispositivos, como discos duros conectados por USB, discos *flash* (los de tipo *pen drive*), unidades removibles, CD-ROM externos, suelen ser dispositivos con algún tipo de emulación scsi, por lo que también se verán como dispositivo de este tipo.

Cualquier dispositivo de almacenamiento presentará una serie de particiones de su espacio. Típicamente, un disco IDE soporta un máximo de cuatro particiones físicas, o más si éstas son lógicas (permiten colocar varias particiones de este tipo sobre una física). Cada partición puede contener tipos de *filesystems* diferentes, ya sean de un mismo operativo o de operativos diferentes.

Para examinar la estructura de un dispositivo conocido o cambiar su estructura particionando el disco, podemos utilizar el comando *fdisk,* o cualquiera de sus variantes más o menos interactivas (*cfdisk, sfdisk).* Por ejemplo, al examinar un disco ejemplo *ide /dev/hda*, nos da la siguiente información:

```
# fdisk -l /dev/hda
Disk /dev/hda: 20.5 GB, 20520493056 bytes 255 heads, 63 sectors/track, 2494 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device  Boot Start End   Blocks Id System
/dev/hda1    *     1     1305   10482381   7 HPFS/NTFS
/dev/hda2    *     1306  2429   9028530    83 Linux
/dev/hda3          2430  2494   522112+    82 Linux swap
```

Disco de 20 GB con tres particiones (se identifican con el número añadido al nombre del dispositivo), donde observamos dos particiones con arranque (columna *Boot* con *) de tipo NTFS y Linux, lo que supone la existencia de un Windows NT/2000/XP/Vista/7 junto con una distribución GNU/Linux, y

la última partición que es usada de *swap* para Linux. Además, tenemos información de la estructura del disco y de los tamaños de cada partición. También se puede obtener información de todas las particiones presentes en el sistema consultado el fichero */proc/partitions*.

De los discos y particiones de que dispongamos, algunos se encontrarán montados en nuestro sistema de ficheros, o estarán preparados para montarse bajo demanda o bien montarse en el momento en que se disponga de medio (en el caso de dispositivos extraíbles).

Esta información la podemos obtener de diferentes maneras (lo veremos con más detalle en el taller final):

- **Fichero */etc/fstab***: indica dispositivos que están preparados para montarse en el arranque o los extraíbles que podrán ser montados. No tienen por qué estar todos los del sistema, sino sólo aquellos que queramos tener en arranque. Los demás podemos montarlos bajo demanda con el comando *mount*, o desmontarlos con *umount*.

- **Comando *mount***: nos informa de los *filesystems* montados en ese momento (ya sean dispositivos reales o *filesystems* virtuales, como */proc*). Podemos obtener esta información también desde el fichero */etc/mtab*.

- **Comando *df -k***: nos informa de los *filesystems* de almacenamiento, y nos permite verificar el espacio usado y disponible. Se trata de un comando básico para controlar el espacio de disco disponible. Respecto a este comando *df -k*, una de nuestras tareas básicas de administración de la máquina es controlar los recursos de ésta, y en este caso el espacio disponible en los *filesystems* utilizados. Estos tamaños hay que monitorizarlos con cierta frecuencia para evitar la caída del sistema; nunca tendría que dejarse un *filesystem* (sobre todo si es el */*) por debajo de un 10-15%, ya que hay muchos procesos (*daemons* de servicios) que están escribiendo información temporal o logs, que pueden consumir gran espacio. Un caso particular lo forman los ficheros *core,* generados por fallos de software y que continen información de depuración de los errores que los han provocado junto con la imagen del proceso, los cuales pueden suponer (dependiendo del proceso) tamaños muy grandes de archivo. Normalmente, habrá que tener algunas precauciones de "limpieza del sistema" si se detectan situaciones de saturación del *filesystem*:

- **Eliminar temporales antiguos.** Los directorios */tmp* y */var/tmp* suelen acumular muchos archivos generados por diferentes usuarios o aplicaciones. Algunos sistemas o distribuciones ya toman medidas de limpieza, como limpiar */tmp* en cada arranque del sistema.

- **Logs:** evitar su crecimiento excesivo, ya que según la configuración del sistema (por ejemplo, de *Syslogd*) la información generada de mensajes pue-

de ser muy grande. Habrá que limpiar periódicamente al llegar a determinados tamaños, y en todo caso, si necesitamos la información para posteriores análisis, podemos realizar *backups* en medios extraíbles. Este proceso puede automatizarse mediante uso de *scripts cron*, o bien por medio de herramientas especializadas como *logrotate*.

- Hay otros puntos del sistema que suelen crecer mucho, como pueden ser:
  - Ficheros *core* de los usuarios: podemos eliminarlos periódicamente o eliminar su generación.
  - El sistema de correo electrónico: almacena todos los correos enviados y recibidos; podemos pedir a los usuarios que hagan limpieza periódica, o bien poner sistemas de cuotas.
  - Las cachés de los navegadores u otras aplicaciones: también suelen tener tamaños grandes, otra limpieza que habrá que hacer periódicamente.
  - Las cuentas de los propios usuarios: pueden tener cuotas para no exceder los tamaños prefijados...

5. Sistema de ficheros

En cada máquina con un sistema GNU/Linux podemos ver sistemas de ficheros de diferentes tipos [Hin]. Para empezar, es habitual encontrarse con los propios sistemas de ficheros Linux creados en distintas particiones de los discos [Koe].

La configuración habitual suele ser de dos particiones:

1) la correspondiente a "/" (*root filesystem*) y

2) la correspondiente al fichero de intercambio o de *swap*.

Con todo, en configuraciones más profesionales suele ser habitual separar particiones con partes "diferenciadas" del sistema. Una técnica habitual es, por ejemplo (veremos más opciones después), crear particiones diferentes para:

```
/    /boot /home      /opt  /tmp  /usr  /var  swap
```

que seguramente se encontrarán montadas desde diferentes orígenes (diferentes discos, o incluso red en algunos casos).

Las particiones se realizan para separar claramente partes estáticas y dinámicas del sistema, para permitir de una forma más fácil, ante problemas de saturación, extender las particiones o aislar más fácilmente partes para la realización de *backups* (por ejemplo, las cuentas de los usuarios en la partición */home*).

El tipo de particiones *swap* es de tipo *Linux swap*, y la correspondiente a */* suele ser de alguno de los sistemas de ficheros estándar, ya sea *ext2* (el tipo por defecto hasta los *kernels* 2.4), *ext3* o el nuevo *ext4*, que son mejoras del *ext2* compatibles pero con *journaling*, lo que permite tener un log de lo que va pasando al sistema de ficheros, para recuperaciones más rápidas en caso de error. También pueden ser habituales otros sistemas de archivos, como Reiser o XFS.

Otra configuración habitual puede ser de tres particiones: */*, *swap*, */home*, donde la */home* se dedicará a las cuentas de los usuarios. Esto permite separar las cuentas de los usuarios del sistema, aislando en dos particiones separadas, y podemos dar el espacio necesario para las cuentas en otra partición.

Otro esquema muy utilizado es el de separar en particiones las partes estáticas del sistema de las dinámicas, por ejemplo, una partición donde se coloca / con la parte estática (*/bin /sbin* y */usr* en algunos casos) que se espera que no va a crecer o lo va a hacer muy poco, y otra o varias con la parte dinámica (*/var /tmp /opt*), suponiendo que */opt*, por ejemplo, es el punto de instalación del software nuevo. Esto permite ajustar mejor el espacio de disco y dejar más espacio para las partes del sistema que lo necesiten.

Respecto a los sistemas de ficheros soportados debemos destacar la gran variedad de ellos; actualmente, podemos encontrar (entre otros):

- **Sistemas asociados a GNU/Linux**, como el estándar *ext2*, *ext3*, evolución del anterior con concepto de *journaling* (soporte de log de operaciones realizadas en el sistema de fichero que puede permitir su recuperación en caso de algún desastre que lo haga inconsistente). O el nuevo *ext4*.

- **Compatibilidad con entornos no GNU/Linux**: *msdos*, *vfat*, *ntfs*, acceso a los diferentes sistemas de *fat16*, *fat32* y *ntfs*. En particular, cabe resaltar que el soporte *kernel*, en el caso del *kernel*, está limitado a lectura. Pero como ya hemos dicho, existen soluciones en espacio de usuario (mediante FUSE, un componente que permite escribir sistemas de ficheros en espacio de usuario), que la permiten, como el ya mencionado *ntfs-3g*. También se disponen de compatibilidad a otros entornos como Mac con *hfs* y *hfsplus*.

- **Sistemas asociados a soportes físicos**, caso de CD/DVD, como los *iso9660* y *udf*.

- **Sistemas usados en diferentes Unix**, que ofrecen generalmente mejor rendimiento (a veces, a costa de mayor consumo de recursos, en CPU por ejemplo), como JFS2 (IBM), XFS (SGI), o ReiserFS.

- **Sistemas de ficheros en red** (más tradicionales): NFS, Samba (*smbfs*, *cifs*), permiten acceder a sistemas de ficheros disponibles en otras máquinas de forma transparente por red.

- **Sistemas distribuidos en red**: como GFS o Coda.

- **Pseudosistemas de ficheros**, como procfs (*/proc*) o sysfs (*/sys*).

En la mayoría (excepto algún caso especial) de estos sistemas de ficheros, GNU/Linux nos permitirá crear particiones de estos tipos, construir el sistema de ficheros del tipo requerido y montarlas como parte integrante del árbol de directorios, ya sea de forma temporal o permanente.

5.1. Puntos de montaje

Aparte del *filesystem* principal / y de sus posibles divisiones en particiones extras (*/usr /var /tmp /home*), cabe tener en cuenta la posibilidad de dejar puntos de montaje preparados para el montaje de otros sistemas de ficheros, ya sea particiones de disco u otros dispositivos de almacenamiento.

En las máquinas en que GNU/Linux comparte la partición con otros sistemas operativos, mediante algún sistema de arranque (*lilo* o *grub*), pueden existir varias particiones asignadas a los diferentes operativos. Muchas veces es interesante compartir datos con estos sistemas, ya sea para leer sus ficheros o modificarlos. A diferencia de otros sistemas (que sólo tienen en cuenta sus propios datos y sistemas de ficheros, y en los cuales en algunas versiones no se soportan algunos de sus propios sistemas de ficheros), GNU/Linux es capaz de tratar, como hemos visto, con una cantidad enorme de sistemas de ficheros de diferentes operativos y poder compartir la información.

Ejemplo

Si en los PC personales hemos instalado GNU/Linux, seguramente encontraremos más de un operativo, por ejemplo, otra versión de GNU/Linux con *ext2* o *3* de sistema de ficheros; podríamos encontrar un antiguo msdos con su sistema de ficheros FAT, un Windows98/ME/XP Home con FAT32 (o *vfat* para Linux) o un Windows NT/2000/XP/Vista/7 con sistemas NTFS (*ntfs* para Linux) y FAT32 (*vfat*) a la vez.

Nuestro sistema GNU/Linux puede leer datos (o sea, ficheros y directorios) de todos estos sistemas de ficheros y escribir en la mayoría de ellos.

En el caso de NTFS, hasta ciertos momentos existieron problemas en la escritura, que estaba en forma experimental en la mayoría de *drivers* de *kernel* aparecidos debido, principalmente, a las diferentes versiones que van apareciendo del sistema de ficheros, ya que existen dos versiones principales llamadas NTFS y NTFS2, y algunas extensiones como los llamados volúmenes dinámicos, o los sistemas de ficheros cifrados. Acceder con según que opción de *drivers* presentaba ciertas incompatibilidades, que podrían causar corrupciones de datos o errores en el sistema de ficheros.

Debido a FUSE, un módulo integrado en el *kernel* (a partir de 2.6.11), se ha permitido un desarrollo más flexible de sistemas de ficheros, directamente en espacio de usuario (de hecho, FUSE actúa como un "puente" entre las peticiones del *kernel* y el acceso que se hace desde el *driver*).

Gracias a las posibilidades de FUSE, se tiene un soporte más o menos completo de NTFS (mientras Microsoft no haga más cambios en la especificación), en especial desde la aparición del *driver* (basado en FUSE) ntfs-3g y la combinación con las utilidades *ntfsprogs*.

Para que se puedan leer o escribir los datos, la partición tiene que estar disponible dentro de nuestro sistema de ficheros raíz (*/*). Por lo tanto, hay que llevar a cabo un proceso de "montaje" del sistema de ficheros en algún punto de nuestro árbol de directorios. Se seguirá el mismo proceso si se trata de un dispositivo de almacenamiento, ya sea disquete o *floppy*.

Dependiendo de la distribución, se usan unos sistemas u otros, o también los podemos crear nosotros. Normalmente, suelen existir o bien como subdirectorios de la raíz, por ejemplo */cdrom /win /floppy*, o bien son subdirectorios dentro de */mnt*, el punto estándar de montaje (aparecen como */mnt/cdrom* / */mnt/floppy*...), o el directorio */media*, que es el preferido últimamente por las distribuciones. Según el estándar FHS, */mnt* se debería usar para montajes temporales de sistemas de archivo, mientras */media* se utilizaría para montar dispositivos removibles.

El proceso de montaje se realiza mediante la orden *mount* con el siguiente formato:

```
mount -t filesystem-type device mount-point
```

El tipo de *filesystem* puede ser: *msdos* (fat), *vfat* (fat32), *ntfs* (ntfs lectura), *iso9660* (para cdrom), *ext2*, *ext3*, *xfs*... (de los disponibles).

El dispositivo es la entrada correspondiente en el directorio */dev* a la localización del dispositivo; los IDE tenían */dev/hdxy*, donde *x* es *a,b,c,* o *d* (1 master, 1 slave, 2 master, 2 slave) *e* y, el número de partición, los SCSI (*/dev/sdx*), donde *x* es *a,b,c,d*... (según el ID SCSI asociado 0,1,2,3,4...).

Vamos a ver algunos casos:

```
mount -t iso9660 /dev/hdc /mnt/cdrom
```

montaría el CD-ROM (si es el IDE que está en el segundo IDE de forma master) en el punto */mnt/cdrom*.

```
mount -t iso9660 /dev/cdrom /mnt/cdrom
```

montaría el CD-ROM; */dev/cdrom* se usa como sinónimo (es un link) del dispositivo donde está conectado.

```
mount -t vfat /dev/fd0H1440 /mnt/floppy
```

montaría el disquete, */dev/fd0H1440*. Sería la disquetera A en alta densidad (1.44 MB). También puede usarse */dev/fd0*.

```
mount -t ntfs /dev/hda2 /mnt/winXP
```

montaría la segunda partición del primer dispositivo IDE (la C:), de tipo NTFS (por ejemplo, un Windows XP).

Si estas particiones son más o menos estables en el sistema (o sea, no cambian frecuentemente) y las queremos utilizar, lo mejor será incluir los montajes para que se hagan en tiempo de ejecución, al iniciar el sistema, mediante la configuración del fichero */etc/fstab*:

```
# /etc/fstab: Información estática del sistema de ficheros
#
#<Sis. ficheros>    <Punto montaje>   <Tipo>     <Opciones>          <volcado>   <pasada>
/dev/hda2           /                 ext3       errors = remountro        0           1
/dev/hdb3           none              swap       sw                        0           0
proc                /proc             proc       defaults                  0           0
/dev/fd0            /floppy           auto       user,noauto               0           0
/dev/cdrom          /cdrom            iso9660    ro,user,noauto            0           0
/dev/sdb1           /mnt/usb          vfat       user,noauto               0           0
```

Por ejemplo, esta configuración incluye algunos de los sistemas estándar, como la raíz en */dev/hda2*, la partición de *swap* que está en *hdb3*, el sistema *proc* (que utiliza el *kernel* para guardar su información). Y el disquete, el CD-ROM, y en este caso un disco USB de tipo *flash* (que se detecta como un dispositivo scsi). En algunos casos, se especifica *auto* como tipo de *filesystem*. Esto permite que se autodetecte el sistema de ficheros. Si se conoce, es mejor indicarlo en la configuración, y por otra parte, el *noauto* en las opciones permite que no sea montado de forma automática siempre, sino bajo petición (o acceso al directorio).

Si tenemos esta información en el fichero, el proceso de montaje se simplifica mucho, ya que se hará o bien en ejecución, en arranque, o bien bajo demanda (para los *noauto*). Y puede hacerse ahora simplemente pidiendo que se monte el punto de montaje o el dispositivo:

```
mount /mnt/cdrom
mount /dev/fd0
```

dado que el sistema ya tiene el resto de la información.

El proceso contrario, el desmontaje, es bastante sencillo, el comando *umount* con punto o dispositivo:

```
umount /mnt/cdrom
umount /dev/fd0
```

En el caso de medios extraíbles, tipo CD-ROM (u otros), puede usarse *eject* para la extracción del soporte físico:

```
eject /dev/cdrom
```

o, en este caso, sólo:

```
eject
```

Los comandos *mount* y *umount* montan o desmontan todos los sistemas disponibles. En el fichero */etc/mtab* se mantiene una lista de los sistemas montados. En un momento concreto se puede consultar o ejecutar *mount* sin parámetros para obtener esta información.

5.2. Permisos

Otro asunto que habrá que controlar, en el caso de los ficheros y directorios, es el de los permisos que queremos establecer en cada uno de ellos; hay que recordar que cada fichero puede disponer de la serie de permisos: *rwxrwxrwx*, donde se corresponden con *rwx* del propietario, *rwx* del grupo al que el usuario pertenece y *rwx* para otros usuarios. En cada uno se puede establecer el permiso de lectura (*r*), escritura (*w*) o ejecución (*x*). En el caso de un directorio, *x* denota el permiso para poder entrar en ese directorio (con el comando *cd*, por ejemplo).

Para modificar los derechos sobre un directorio o fichero, existen los comandos:

- ***chown***: cambiar propietario de los ficheros.
- ***chgrp***: cambiar grupo propietario de los ficheros.
- ***chmod***: cambiar permisos específicos (*rwx*) de los archivos.

Estos comandos también permiten la opción *-R*, que es recursiva si se trata de un directorio.

6. Usuarios y grupos

Los usuarios de un sistema GNU/Linux disponen de una cuenta asociada (definida con algunos de sus datos y preferencias), junto con el espacio en disco para que puedan desarrollar sus archivos y directorios. Este espacio está asignado al usuario, y sólo puede ser usado por éste (a no ser que los permisos especifiquen cosas diferentes).

Dentro de las cuentas asociadas a usuarios, podemos encontrar diferentes tipos:

- **La del administrador**, con identificador *root*, que sólo es (o debería ser) utilizada para las operaciones de administración. El usuario *root* es el que dispone de más permisos y acceso completo a la máquina y a los archivos de configuración. Por lo tanto, también es el que más daño puede causar por errores u omisiones. Es mejor evitar usar la cuenta de *root* como si fuese un usuario más, por lo que se recomienda dejarla sólo para operaciones de administración.

- **Cuentas de usuarios**: las cuentas normales para cualquier usuario de la máquina tienen los permisos restringidos al uso de ficheros de su cuenta, y a algunas otras zonas particulares (por ejemplo, los temporales en */tmp*), así como a utilizar algunos dispositivos para los que se les haya habilitado permisos.

- **Cuentas especiales de los servicios**: *lp*, *news*, *wheel*, *www-data*... cuentas que no son usadas por personas, sino por servicios internos del sistema, que los usa bajo estos nombres de usuario. Algunos de los servicios también son usados bajo el usuario de *root* (aunque por razones de seguridad debería evitarse).

Un usuario normalmente se crea mediante la especificación de un nombre (o identificador de usuario), una palabra de paso (*password*) y un directorio personal asociado (la cuenta).

La información de los usuarios del sistema está incluida en los siguientes archivos:

```
/etc/passwd
/etc/shadow
/etc/group
/etc/gshadow
```

Unas líneas del */etc/passwd* podrían ser:

```
juan:x:1000:1000:Juan Garcia,,,:/home/juan:/bin/bash
root:x:0:0:root:/root:/bin/bash
```

donde se indica (si aparecen :: seguidos, que el campo está vacío):

- *juan*: identificador de usuario en el sistema.
- *x*: palabra de paso del usuario codificada, si hay una "*x*" es que se encuentra en el fichero */etc/shadow*.
- *1000*: código del usuario; lo usa el sistema como código de identidad del usuario.
- *1000*: código del grupo principal al que pertenece; la información del grupo se encuentra en */etc/group*.
- *Juan García*: comentario; suele colocarse el nombre completo del usuario, o algún comentario para identificar el objetivo de la cuenta.
- */home/juan*: directorio personal asociado a su cuenta.
- */bin/bash*: *shell* interactivo que utilizará el usuario al interactuar con el sistema, en modo texto, o por *shell* gráfico. En este caso, el *shell* Bash de GNU, que es el utilizado por defecto. El fichero */etc/passwd* solía contener las palabras de paso de los usuarios de forma encriptada, pero el problema estaba en que cualquier usuario podía ver el fichero, y en su momento se diseñaron *cracks* que intentaban encontrar en forma bruta la palabra de paso, mediante la palabra de paso encriptada como punto de partida (palabra codificada con el sistema *crypt).*

Para evitar esto, hoy en día ya no se colocan las palabras de paso en este archivo, sólo una "*x*" que indica que se encuentran en otro fichero, que es sólo de lectura para el usuario *root*, */etc/shadow,* cuyo contenido podría ser algo parecido a lo siguiente:

```
juan:a1gNcs82ICst8CjVJS7ZFCVnu0N2pBcn/:12208:0:99999:7:::
```

donde se encuentra el identificador del usuario junto con la palabra de paso encriptada. Además, aparecen (como campos separados por ":" con información respecto a la contraseña:

- Días desde el 1 de enero de 1970 en que la palabra de paso se cambió por última vez.

- Días que faltan para que se cambie (0 no hay que cambiarla).
- Días después en que hay que cambiarla (o sea, plazo de cambio).
- Días en que el usuario será avisado antes de que le expire.
- Días una vez expirado, en que se producirá la deshabilitación de la cuenta.
- Días desde el 1 de enero de 1970 en que la cuenta está deshabilitada.
- Y un campo reservado.

Además, las claves de encriptación pueden ser más difíciles, ya que ahora puede utilizarse un sistema denominado md5 (suele aparecer como opción a la hora de instalar el sistema) para proteger las palabras de paso de los usuarios. Veremos más detalles al respecto en la unidad dedicada a la seguridad.

En */etc/group* está la información de los grupos de usuarios:

```
jose:x:1000:
```

donde tenemos:

```
nombre-grupo:contraseña-grupo:identificador-del-grupo:lista-usuarios
```

La lista de usuarios del grupo puede estar presente o no, pues la información ya está en */etc/passwd*; no suele ponerse en */etc/group*. Si se pone, suele aparecer como una lista de usuarios separada por comas. Los grupos también pueden poseer una contraseña asociada (aunque no suele ser tan común), como en el caso de los de usuario, en que también existe un fichero de tipo shadow: */etc/gshadow*.

Otros ficheros interesantes son los del directorio */etc/skel*, donde se hallan los ficheros que se incluyen en cada cuenta de usuario al crearla. Recordad que, como vimos con los *shell* interactivos, podíamos tener unos *scripts* de configuración que se ejecutaban al entrar o salir de la cuenta. En el directorio *skel* se guardan los "esqueletos" que se copian en cada usuario al crearlo. Suele ser responsabilidad del administrador crear unos ficheros adecuados para los usuarios, poniendo los *path* necesarios de ejecución, inicialización de variables de sistema, variables que se necesiten para el software, etc.

A continuación, vamos a ver una serie de comandos útiles para esta administración de usuarios (mencionamos su funcionalidad y en el taller haremos algunas pruebas):

- ***useradd***: añadir un usuario al sistema.

- ***userdel***: borrar un usuario del sistema.

- ***usermod***: modificar un usuario del sistema.

- ***groupadd, groupdel, groupmod***: lo mismo para grupos.

- ***newusers, chpasswd***: pueden ser de utilidad en grandes instalaciones con muchos usuarios, ya que permiten crear varias cuentas desde la información introducida en un fichero (*newusers*) o bien cambiar las contraseñas a un gran número de usuarios (*chpasswd).*

- ***chsh***: cambiar el *shell* de *login* del usuario.

- ***chfn***: cambiar la información del usuario, la presente en el comentario del fichero */etc/passwd*.

- ***passwd***: cambiar la contraseña de un usuario. Puede ejecutarse como usuario, y entonces pide la contraseña antigua y la nueva. En el caso de hacerlo, el *root* tiene que especificar el usuario al que va a cambiar la contraseña (si no, estaría cambiando la suya) y no necesita la contraseña antigua. Es quizás el comando más usado por el *root*, de cara a los usuarios cuando se les olvida la contraseña antigua.

- ***su***: una especie de cambio de identidad. Lo utilizan tanto usuarios como el *root* para cambiar el usuario actual. En el caso del administrador, es bastante utilizado para testear que la cuenta del usuario funcione correctamente. Hay diferentes variantes: *su* (sin parámetros, sirve para pasar a usuario *root*, previa identificación, permitiendo, cuando estamos en una cuenta de usuario, pasar a *root* para hacer alguna tarea). La sentencia *su iduser* cambia el usuario a *iduser*, pero dejando el entorno como está, o sea, en el mismo directorio. El mandato *su - iduser* hace una sustitución total, como si el segundo usuario hubiese entrado en el sistema haciendo un *login*.

Respecto a la administración de usuarios y grupos, lo que hemos comentado aquí hace referencia a la administración local de una sola máquina. En sistemas con múltiples máquinas que comparten los usuarios suele utilizarse otro sistema de gestión de la información de los usuarios. Estos sistemas, denominados genéricamente sistemas de información de red, como NIS, NIS+ o LDAP, utilizan bases de datos para almacenar la información de los usuarios y grupos, de manera que se utilizan máquinas servidoras, donde se almacena la base de datos, y otras máquinas clientes, donde se consulta esta información. Esto permite tener una sola copia de los datos de los usuarios (o varias sincronizadas), y que éstos puedan entrar en cualquier máquina disponible del conjunto administrado con estos sistemas. Además, estos sistemas incorporan conceptos adicionales de jerarquías, y/o dominios/zonas de máquinas y recursos, que

permiten representar adecuadamente los recursos y su uso en organizaciones con diferentes estructuras de organización interna de su personal y sus secciones internas.

Podemos comprobar si estamos en un entorno de tipo NIS si en las líneas *passwd* y *group* del archivo de configuración */etc/nsswitch.conf* aparece *files* en primer término, si estamos trabajando con los ficheros locales, o bien *nis* o *nisplus* según el sistema con que estemos trabajando. En general, para el usuario simple no supone ninguna modificación, ya que la gestión de las máquinas le es transparente, y más si se combina con ficheros compartidos por NFS que permite disponer de su cuenta sin importar con qué máquina trabaje. La mayor parte de los comandos anteriores pueden seguir usándose sin problema bajo NIS o NIS+; son equivalentes a excepción del cambio de contraseña, que en lugar de *passwd* se suele hacer con *yppasswd* (NIS) o *nispasswd* (NIS+), aunque suele ser habitual que el administrador los renombre (por un enlace) a *passwd*, con lo cual los usuarios no notarán la diferencia.

Veremos este y otros modos de configuración en las unidades de administración de red.

7. Servidores de impresión

Potencia y flexibilidad

Los sistemas UNIX disponen, quizás, de los sistemas de impresión más potentes y complejos, que aportan una gran flexibilidad a los entornos de impresión.

El sistema de impresión de GNU/Linux [Gt] [Smi02] está heredado de la variante BSD de UNIX. Este sistema se denominaba LPD (Line Printer Daemon). Es un sistema de impresión muy potente, ya que integra capacidades para gestionar tanto impresoras locales como de red y ofrece dentro del mismo tanto el cliente como el servidor de impresión. De forma semejante también UNIX ha dispuesto generalmente del System V Line Printer (o LPR), que era el sistema común en las otras variantes de UNIX. GNU/Linux ha integrado originalmente ambos sistemas, bien usando principalmente LPD y emulando LPR o dependiendo de la distribución integrando por defecto uno u otro.

LPD es un sistema bastante antiguo, puesto que se remonta a los orígenes de la rama BSD de UNIX (mediados de los ochenta). Por lo tanto, a LPD le suele faltar soporte para los dispositivos modernos, ya que en origen el sistema no estuvo pensado para el tipo de impresoras actuales. Tampoco fue concebido como un sistema basado en controladores de dispositivo, pues se producían sólo impresoras serie o paralelas de escritura de caracteres texto.

Para la situación actual, el sistema LPD se combina con otro software común, como el sistema Ghostscript, que ofrece salida de tipo *postscript* para un rango muy amplio de impresoras para las que posee controladores. Además, se suele combinar con algún software de filtraje, que según el tipo de documento a imprimir, selecciona filtros adecuados para adaptar la impresión de documentos o formatos binarios, al sistema de impresión de destino. Así, normalmente el proceso que se sigue es (básicamente):

1) El trabajo es iniciado por un comando del sistema LPD.

2) El sistema de filtro identifica qué tipo de trabajo (o fichero) es utilizado y convierte el trabajo a un fichero *postscript* de salida, que es el que se envía a la impresora. En GNU/Linux y UNIX, la mayoría de aplicaciones suponen que la salida será hacia una impresora *postscript*, y muchas de ellas generan salida *postscript* directamente, y por esta razón se necesita el siguiente paso.

3) Ghostscript se encarga de interpretar el fichero *postscript* recibido, y según el controlador de la impresora a la que ha sido enviado el trabajo, realiza la conversión al formato propio de la impresora. Si es de tipo *postscript*, la impresión es directa; si no, habrá que realizar la traducción. El trabajo se manda a la cola de impresión.

Como hemos dicho, además del sistema de impresión LPD (con origen en los BSD UNIX), también existe el denominado sistema SystemV (de origen en la otra rama UNIX de SystemV) o LPR. Por compatibilidad, actualmente

la mayor parte de UNIX integra ambos, de manera que o bien uno u otro es el principal, y el otro se simula sobre el principal. En el caso de GNU/Linux, pasa algo parecido; según la instalación que hagamos podemos tener sólo los comandos LPD de sistema de impresión, pero también será habitual disponer de los comandos SystemV. Una forma sencilla de identificar los dos sistemas (BSD o SystemV) es con el comando principal de impresión (el que envía los trabajos al sistema), en BSD es *lpr*, y en SystemV es *lp*.

Este era el panorama inicial de los sistemas de impresión de GNU/Linux, pero en los últimos años han surgido más sistemas, que permiten una mayor flexibilidad y una mayor disposición de controladores para las impresoras. Los dos principales sistemas son CUPS y, en menor grado, LPRng (de hecho, ya obsoleto, se utilizó en algunas versiones de Fedora, y ya no lo comentaremos en esta revisión del material; puede encontrase en ediciones anteriores). Últimamente es CUPS el estándar de facto para GNU/Linux, aunque los otros sistemas han de ser soportados por compatibilidad con sistemas UNIX existentes.

Los dos (tanto CUPS como LPRng) son una especie de sistema de más alto nivel, pero que no se diferencian en mucho de cara al usuario respecto a los BSD y SystemV estándar. Por ejemplo, se utilizan los mismos comandos clientes (o compatibles en opciones) para imprimir. Para el administrador sí que supondrá diferencias, ya que los sistemas de configuración son diferentes. En cierto modo, podemos considerar a LPRng y CUPS como nuevas arquitecturas de sistemas de impresión, que son compatibles de cara al usuario con los comandos antiguos.

En las distribuciones GNU/Linux actuales podemos encontrarnos con los diferentes sistemas de impresión. Si la distribución es antigua, puede que lleve incorporado tan sólo el sistema BSD LPD. En las actuales, tanto Debian como Fedora/Red Hat utilizan CUPS. En algunas versiones de Red Hat existía una herramienta, *Print switch*, que permitia cambiar el sistema, conmutar de sistema de impresión, aunque últimamente sólo está disponible CUPS. En Debian pueden instalarse ambos sistemas, pero son exclusivos, sólo uno de ellos puede gestionar la impresión.

En el caso de Fedora, el sistema de impresión por defecto es CUPS (desapareciendo LPRng en Fedora Core 4), y la herramienta *Print switch* ya no existe por no ser necesaria. Se utiliza *system-config-printer* para la configuración de dispositivos. Debian, por defecto, utilizaba BSD LPD, pero ya es común instalar CUPS (y es la opción por defecto en nuevas versiones), y también puede utilizar LPRng. Además, cabe recordar que también teníamos la posibilidad (vista en la unidad de migración) de interaccionar con sistemas Windows mediante protocolos Samba, que permitían compartir las impresoras y el acceso a éstas.

Respecto a cada uno de los sistemas [Gt]:

- **BSD LPD**: es uno de los estándares de UNIX, y algunas aplicaciones asumen que tendrán los comandos y el sistema de impresión disponibles, por lo cual, tanto LPRng como CUPS emulan el funcionamiento y los comandos de BDS LPD. El sistema LPD es utilizable, pero no muy configurable, sobre todo en el control de acceso; por eso las distribuciones se han movido a los otros sistemas más modernos.

- **LPRng**: se diseñó para ser un reemplazo del BSD; por lo tanto, la mayor parte de la configuración es parecida y sólo difiere en algunos ficheros de configuración.

- **CUPS**: se trata de una desviación mayor del BSD original, y la configuración es propia. Se proporciona información a las aplicaciones sobre las impresoras disponibles (también en LPRng). En CUPS, tanto el cliente como el servidor tienen que disponer de software CUPS.

Los dos sistemas tienen emulación de los comandos de impresión de SystemV.

Para la impresión en GNU/Linux, hay que tener en cuenta varios aspectos:

- **Sistema de impresión que se utiliza**: BSD, CUPS, o LPRng (hoy prácticamente obsoleto).

Nota

Podéis encontrar información de las impresoras más adecuadas y de los controladores en: http://www.openprinting.org/printer_list.cgi

- **Dispositivo de impresión** (impresora): puede disponer de conexión local a una máquina o estar colocada en red. Las impresoras actuales pueden estar colocadas por conexiones locales a una máquina mediante interfaces serie, paralelo, USB, etc., o disponibles simplemente en red, como una máquina más, o con protocolos especiales propietarios. Las conectadas a red pueden actuar ellas mismas de servidor de impresión (por ejemplo, muchas láser son servidores BSD LPD), o bien pueden colgarse de una máquina que actúe de servidor de impresión para ellas.

- **Protocolos de comunicación** utilizados con la impresora o el sistema de impresión: ya sea TCP/IP directo (por ejemplo, una HP con LPD), o bien otros de más alto nivel sobre TCP/IP, como IPP (CUPS), JetDirect (algunas impresoras HP), etc. Este parámetro es importante, ya que lo debemos conocer para instalar la impresora en un sistema.

- **Sistema de filtros usado**: cada sistema de impresión soporta uno o varios.

- **Y los controladores de las impresoras**: en GNU/Linux hay bastantes tipos diferentes; podemos mencionar, por ejemplo, controladores de CUPS, propios o de los fabricantes (por ejemplo, HP y Epson los proporcionan); Gimp, el programa de retoque de imágenes, también posee controladores optimizados para la impresión de imágenes; Foomatic, un sistema de gestión de controladores que funciona con la mayoría de sistemas (CUPS,

LPD, LPRng y otros); los controladores de Ghostscript, etc. En casi todas las impresoras tienen uno o más controladores de estos conjuntos.

Respecto a la parte cliente del sistema, los comandos básicos son iguales para los diferentes sistemas. Estos son los comandos del sistema BSD (cada sistema soporta emulación de estos comandos):

- ***lpr***: envía un trabajo a la cola de la impresora por defecto (o a la que se selecciona); el *daemon* de impresión (*lpd*) se encarga de enviarlo a la cola correspondiente y asigna un número de trabajo, que será usado con los otros comandos. La impresora, por defecto, estaría indicada por una variable de sistema PRINTER, o se utilizará la primera que exista definida. En algunos sistemas se utiliza la cola *lp* (como nombre por defecto).

```
lpr -Pepson datos.txt
```

Esta instrucción mandaría el fichero *datos.txt* a la cola de impresión asociada a una impresora que hemos definido como "epson".

- ***lpq***: nos permite examinar los trabajos existentes en la cola.

```
# lpq -P epson
Rank  Owner        Job   Files          Total Size
1st   juan         15    datos.txt      74578 bytes
2nd   marta        16    fpppp.F        12394 bytes
```

Este comando nos muestra los trabajos en cola, con el orden y tamaños de éstos. Los ficheros pueden aparecer con nombres diferentes, ya que depende de si los hemos enviado con *lpr* o con otra aplicación que puede cambiar los trabajos de nombre al enviarlos, o si han tenido que pasar por algún filtro al convertirlos.

- ***lprm***: elimina trabajos de la cola. Podemos especificar un número de trabajo, o un usuario para cancelar los trabajos.

```
lprm -Pepson 15
```

Eliminar el trabajo con id 15 de la cola.

Respecto a la parte administrativa (en BSD), el comando principal sería *lpc*. Este comando permite activar y desactivar colas, mover trabajos en el orden de las colas y activar o desactivar las impresoras (se pueden recibir trabajos en las colas, pero no se envían a las impresoras).

Cabe mencionar asimismo que, para el caso de SystemV, los comandos de impresión suelen también estar disponibles, simulados sobre los de BSD. En el caso cliente, los comandos son *lp, lpstat, cancel* y, para temas de administración, *lpadmin, accept, reject, lpmove, enable, disable, lpshut*.

En las siguientes secciones veremos cómo hay que configurar un servidor de impresión para dos de los sistemas principales. Estos servidores sirven tanto para la impresión local como para atender las impresiones de clientes de red (si están habilitados).

7.1. BSD LPD

En el caso del servidor BSD LPD, hay dos ficheros principales para examinar: por una parte, la definición de las impresoras en */etc/printcap* y, por otra, los permisos de acceso por red en */etc/hosts.lpd*.

Respecto a los permisos, por defecto BSD LPD sólo deja acceso local a la impresora, y por lo tanto, hay que habilitarlo expresamente en */etc/hosts.lpd*.

El fichero podría ser:

```
#fichero hosts.lpd
second
first.the.com
192.168.1.7
+@groupnis
-three.the.com
```

Indicaría que está permitida la impresión a una serie de máquinas, listadas bien por su nombre DNS o por la dirección IP. Se pueden añadir grupos de máquinas que pertenezcan a un servidor NIS (como en el ejemplo *groupnis*) o bien no permitir acceso a determinadas máquinas indicándolo con un guión "-".

En cuanto a la configuración del servidor en */etc/printcap*, se definen entradas, donde cada una representa una cola del sistema de impresión a la que pueden ir a parar los trabajos. La cola puede estar tanto asociada a un dispositivo local como a un servidor remoto, ya sea éste una impresora u otro servidor.

En cada entrada, pueden existir las opciones:

- ***lp*=**: nos indica a qué dispositivo está conectada la impresora, por ejemplo *lp = /dev/lp0* indicaría el primer puerto paralelo. Si la impresora es de tipo LPD, por ejemplo, una impresora de red que acepta el protocolo LPD (como una HP), entonces podemos dejar el campo vacío y rellenar los siguientes.

- ***rm*=**: dirección con nombre o IP de la máquina remota que dispone de la cola de impresión. Si se trata de una impresora de red, será la dirección de ésta.

- ***rp*=**: nombre de la cola remota, en la máquina indica antes con *rm*.

```
# Entrada de una impresora local
lp|epson|Epson C62:\
   :lp=/dev/lp1:sd=/var/spool/lpd/epson:\
   :sh:pw#80:pl#72:px#1440:mx#0:\
   :if = /etc/magicfilter/StylusColor@720dpi-filter:\filtro
   :af = /var/log/lp-acct:lf = /var/log/lp-errs:
# Entrada de impresora remota
hpremota|hpr|hp remota del departamento|:\
   :lp = :\
   :rm = servidor:rp = colahp:\
   :lf = /var/adm/lpd_rem_errs:\fichero de log.
   :sd = /var/spool/lpd/hpremota:spool local asociado
```

7.2. CUPS

CUPS es una nueva arquitectura para el sistema de impresión bastante diferente; tiene una capa de compatibilidad hacia BSD LPD, que le permite interaccionar con servidores de este tipo. Soporta también un nuevo protocolo de impresión llamado IPP (basado en http), pero sólo disponible cuando cliente y servidor son de tipo CUPS. Además, utiliza un tipo de *drivers* denominados PPD que identifican las capacidades de la impresora. CUPS ya trae algunos de estos controladores, y algunos fabricantes también los ofrecen (como HP y Epson).

CUPS tiene un sistema de administración completamente diferente, basado en diferentes ficheros: */etc/cups/cupsd.conf* centraliza la configuración del sistema de impresión, */etc/cups/printers.conf* controla la definición de impresoras y */etc/cups/classes.conf* los grupos de éstas.

En */etc/cups/cupsd.conf* configuramos el sistema según una serie de secciones del archivo y las directivas de las diferentes acciones. El archivo es bastante grande; destacaremos algunas directivas importantes:

- ***Allow***: nos permite especificar qué máquinas podrán acceder al servidor, ya sean grupos o máquinas individuales, o segmentos IP de red.

- ***AuthClass***: permite indicar si se pedirá que se autentifiquen los usuarios clientes o no.

- ***BrowseXXX***: hay una serie de directivas relacionadas con la posibilidad de examinar la red para encontrar impresoras servidas. Esta posibilidad está activada por defecto (*browsing* en *on*); por lo tanto, encontraremos disponibles todas las impresoras disponibles en la red. Podemos desactivarla, para solamente observar las impresoras que hayamos definido. Otra opción importante es *BrowseAllow*, que dice a quién le damos la posibilidad de preguntar por nuestras impresoras. Por defecto está habilitada, por lo que cualquiera puede ver nuestra impresora desde nuestra red.

Cabe señalar que CUPS, en principio, está pensado para que tanto clientes como el servidor funcionen bajo el mismo sistema; si los clientes utilizan LPD, hay que instalar un *daemon* de compatibilidad llamado *cups-lpd* (en paquetes

como *cupsys-bsd*). En este caso, CUPS acepta trabajos que provengan de un sistema LPD, pero no controla los accesos (*cupsd.conf* sólo sirve para el propio sistema CUPS), por lo que habrá que implementar alguna estrategia de control de acceso, tipo *firewall*.

Para la administración desde línea de comandos, CUPS es un tanto peculiar, ya que acepta tanto comandos LPD como SystemV en los clientes, y la administración suele hacerse con el comando *lpadmin* de SystemV. En cuanto a herramientas gráficas, disponemos de *gnome-cups-manager*, *gtklp*, utilidades como *system-config-printers* o la interfaz por web que trae el mismo sistema CUPS, accesible en http://localhost:631.

Interfaz para la administración del sistema CUPS.

Respecto a los paquetes software relacionados con CUPS, en una Debian encontramos (entre otros):

```
cupsys - Common UNIX Printing System(tm) - server
cupsys-bsd - Common UNIX Printing System(tm) - BSD commands
cupsys-client - Common UNIX Printing System(tm) - client programs (SysV)
cupsys-driver-gimpprint - Gimp-Print printer drivers for CUPS
cupsys-pt - Tool for viewing/managing print jobs under CUPS
foomatic-db - linuxprinting.org printer support - database
foomatic-db-engine - linuxprinting.org printer support - programs
foomatic-db-gimp-print - linuxprinting - db Gimp-Print printer drivers
foomatic-db-hpijs - linuxprinting - db HPIJS printers
foomatic-filters - linuxprinting.org printer support - filters
foomatic-filters-ppds - linuxprinting - prebuilt PPD files
foomatic-gui - GNOME interface for Foomatic printer filter system
gimpprint-doc - Users' Guide for GIMP-Print and CUPS
gimpprint-locales - Locale data files for gimp-print
```

```
gnome-cups-manager - CUPS printer admin tool for GNOME
gtklp - Frontend for cups written in gtk
```

8. Discos y gestión *filesystems*

Respecto a las unidades de almacenamiento, como hemos visto, poseen una serie de dispositivos asociados, dependiendo del tipo de interfaz:

- **IDE**: dispositivos como:
  /dev/hda disco master, primer conector IDE;
  /dev/hdb disco slave, del primer conector;
  /dev/hdc master, segundo conector;
  /dev/hdd slave, segundo conector.

- **SCSI**: dispositivos */dev/sda*, */dev/sdb*... siguiendo la numeración que tengan los periféricos en el Bus SCSI. Los discos SATA e IDE del mismo sistema también suelen seguir esta nomenclatura, debido a la capa de emulación scsi presente en el *kernel* para estos dispositivos.

- **Disquetes**: dispositivos */dev/fdx*, con *x* número de disquetera (comenzando en 0). Hay diferentes dispositivos, dependiendo de la capacidad del disquete, por ejemplo, el disquete de 1.44 MB en la disquetera A sería */dev/fd0H1440*.

Respecto a las particiones presentes, el número que sigue al dispositivo representa el índice de la partición dentro del disco, y es tratado como un dispositivo independiente: */dev/hda1* primera partición del primer disco IDE, o */dev/sdc2*, segunda partición del tercer dispositivo SCSI. En el caso de los discos IDE, éstos permiten cuatro particiones denominadas primarias y un mayor número en extendidas (o lógicas). Así, si */dev/hdan*, *n* será inferior o igual a 4, se tratará de una partición primaria; si no, se tratará de una partición lógica con *n* superior o igual a 5.

Con los discos y los sistemas de ficheros (*filesystems*) asociados, los procesos básicos que podemos realizar se engloban en:

- **Creación de particiones**, o modificación de éstas. Mediante comandos como *fdisk* o parecidos (*cfdisk, sfdisk)*.

- **Formateo de disquetes**: en caso de disquetes, pueden utilizarse diferentes herramientas: *fdformat (*formateo de bajo nivel), *superformat (*formateo a diferentes capacidades en formato msdos), *mformat (*formateo específico creando *filesystem* msdos estándar).

- **Creación de *filesystems* Linux**, en particiones, mediante el comando *mkfs*. Hay versiones específicas para crear *filesystems* diferentes: *mkfs.ext2*, *mkfs.ext3*, y también *filesystems* no Linux: *mkfs.ntfs*, *mkfs.vfat, mkfs.msdos*,

mkfs.minix u otros. Para CD-ROM como *mkisofs,* a la hora de crear los iso9660 (con extensiones *joliet* o *rock ridge*), que puedan ser una imagen de lo que después se acabará grabando sobre un CD/DVD, y junto con comandos como *cdrecord* (o *wodim*) permitirá finalmente crear/grabar los CD/DVD. Otro caso particular es la orden *mkswap,* que permite crear áreas de *swap* en particiones que, más tarde, se pueden activar o desactivar con *swapon* y *swapoff*.

- **Montaje de los *filesystems***: comandos *mount, umount*.

- **Verificación de estado**: la principal herramienta de verificación de *filesystems* Linux es el comando *fsck*. Este comando comprueba las diferentes áreas del sistema de ficheros para verificar la consistencia y comprobar posibles errores y, en los casos en que sea posible, corregirlos. El propio sistema activa automáticamente el comando *fsck* en el arranque cuando detecta situaciones donde se ha producido una parada incorrecta (un apagón eléctrico o accidental de la máquina), o bien pasado un cierto número de veces en que el sistema se ha arrancado. Esta comprobación suele comportar cierto tiempo, normalmente algunos minutos (dependiendo del tamaño de datos). También existen versiones particulares para otros sistemas de ficheros: *fsck.ext2*, *fsck.ext3*, *fsck.vfat*, *fsck.msdos*, etc. El proceso del *fsck* se realiza con el dispositivo en modo de "sólo lectura" con particiones montadas. Se recomienda desmontar las particiones para realizar el proceso si se detectan errores y hay que aplicar correcciones. En determinados casos, por ejemplo, si el sistema por comprobar es la raíz / y se detecta algún error crítico, se nos pedirá que cambiemos de modo de ejecución del sistema (*runlevel*) hacia modo sólo *root* y hagamos allí la verificación. En general, si hay que hacer la verificación, se recomienda hacer éstas en modo superusuario (podemos conmutar de modo de *runlevel* con los comandos *init* o *telinit*).

- **Procesos de *backup***: ya sean del disco, bloques de disco, particiones, *filesystems*, ficheros... Hay varias herramientas útiles para ello: *tar* nos permite copiar ficheros hacia un fichero o a unidades de cinta; *cpio,* de forma parecida, puede realizar *backups* de ficheros hacia un fichero; tanto *cpio* como *tar* mantienen información de permisos y propietarios de los ficheros; *dd* permite copias, ya sea de ficheros, dispositivos, particiones o discos a fichero; es un poco complejo y hay que conocer información de bajo nivel, tipo, tamaños, número de bloques y/o sectores... Puede enviarse también a cintas.

- **Utilidades diversas**: ciertos comandos individuales, algunos de ellos utilizados por los procesos anteriores para hacer tratamientos distintos: *badblocks* para encontrar bloques defectuosos en el dispositivo; *dumpe2fs* para obtener información sobre *filesystems* Linux; *tune2fs* permite hacer proce-

sos de *tunning* de *filesystems* Linux de tipo ext2, ext3 o ext4 y ajustar diferentes parámetros de comportamiento.

A continuación, destacamos dos temas relacionados con la concepción del espacio de almacenamiento, que son utilizados en varios ambientes para la creación base del espacio de almacenamiento: el uso de RAID software y la creación de volúmenes dinámicos.

8.1. RAID software

La configuración de discos mediante esquemas RAID es uno de los esquemas de almacenamiento de alta disponibilidad más usados actualmente, cuando disponemos de varios discos para implementar nuestros sistemas de ficheros.

El enfoque principal de las diferentes técnicas existentes está en la tolerancia a fallos que se proporciona desde un nivel de dispositivo, el conjunto de discos, a diferentes tipos posibles de fallos tanto físicos como de sistema, para evitar las pérdidas de datos o los fallos de coherencia en el sistema. Así también algunos esquemas que están diseñados para aumentar las prestaciones del sistema de discos, ampliando el ancho de banda de estos disponible hacia el sistema y las aplicaciones.

Típicamente, hoy en día RAID en hardware (mediante tarjetas controladoras hardware), se puede encontrar en servidores empresariales (cuando comienzan a tener cierta presencia en equipos de escritorio, con placas base con capacidades para algunos RAID), donde se hallan disponibles diferentes soluciones hardware que cumplen estos requerimientos de fiabilidad y de maximizar prestaciones. En particular, para ambientes con aplicaciones intensivas en disco, como *streaming* de audio y/o vídeo, o grandes bases de datos.

Conviene destacar un error común en el tema RAID, que proporciona ciertas capacidades de tolerancia a fallos, pero no evita la realización de *backups* de los datos disponibles de forma periódica. Si se supera la capacidad de tolerancia a fallos, también se pierden datos (en algunos casos, completamente).

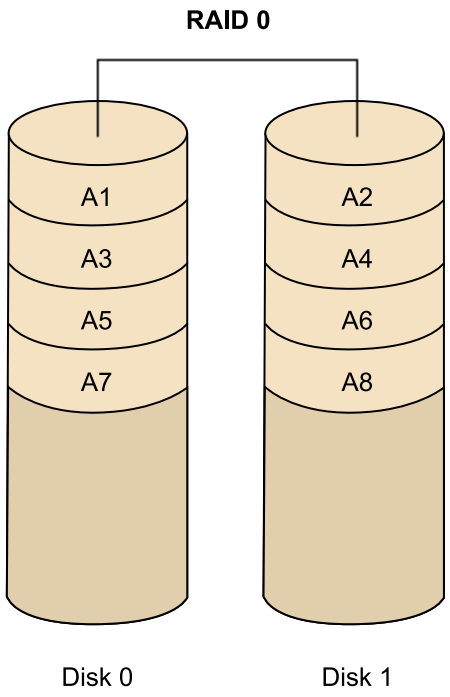
En general, este hardware se encuentra en forma de tarjetas (o integrado en la máquina) de tipo controladora RAID de discos, que implementan la gestión de uno o más niveles (de la especificación RAID), sobre un conjunto de discos (desde mínimo dos discos) administrado por esta controladora.

En RAID se distingue una serie de niveles (o configuraciones posibles) que pueden proporcionarse (cada fabricante de hardware, o el software concreto, puede soportar uno o varios de estos niveles). Cada nivel de RAID se aplica sobre un conjunto de discos, a veces denominado array RAID (o matriz de discos RAID), los cuales suelen ser (idealmente) discos iguales en tamaño (o iguales a tamaños por grupos). Por ejemplo, para realizar un un caso de array podrían utilizarse cuatro discos de 500GB, o en otro caso, dos grupos (a 500GB) de

dos discos, uno de 150GB y otro de 350GB. En algunos casos de controladores hardware, no se permite que los discos (o en grupos) sean de diferentes tamaños; en otros pueden utilizarse, pero el array queda definido por el tamaño del disco (o grupo) más pequeño.

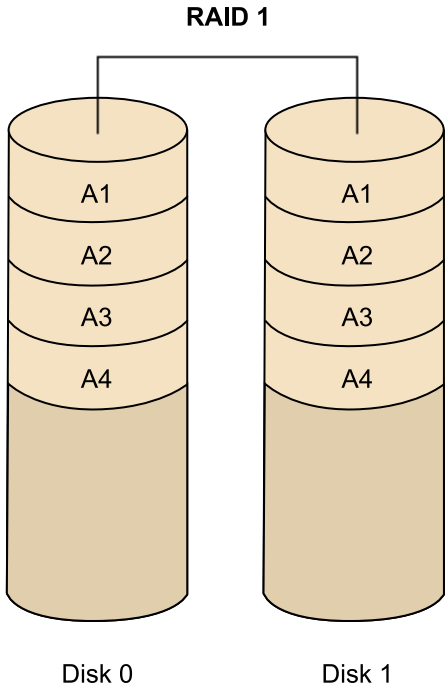
Describimos conceptos básicos de algunos niveles en la siguiente lista (tened en cuenta que, en algunos casos, la terminología no es plenamente aceptada, y puede depender de cada fabricante):

- **RAID 0**: se distribuyen los datos equitativamente entre uno o más discos sin información de paridad o redundancia; no se está ofreciendo tolerancia al fallo. Sólo se están repartiendo datos; si el disco falla físicamente la información se pierde y debemos recuperarla desde copias de seguridad. Lo que sí aumenta es el rendimiento, dependiendo de la implementación de RAID0, ya que las operaciones de lectura y escritura se dividirán entre los diferentes discos.



- **RAID 1**: se crea una copia exacta (*mirror)* en un conjunto de dos o más discos (denominado array RAID). En este caso, resulta útil para el rendimiento de lectura (que puede llegar a incrementarse de forma lineal con el número de discos), y en especial por disponer de tolerancia al fallo de uno de los discos, ya que (por ejemplo, con dos discos) se dispone de la misma información. RAID 1 suele ser adecuado para alta disponibilidad, como entornos de 24x7, donde debemos disponer críticamente de los recursos. Esta configuración nos permite también (si el hardware lo soporta) el intercambio en caliente (*hot-swap*) de los discos. Si detectamos el fallo

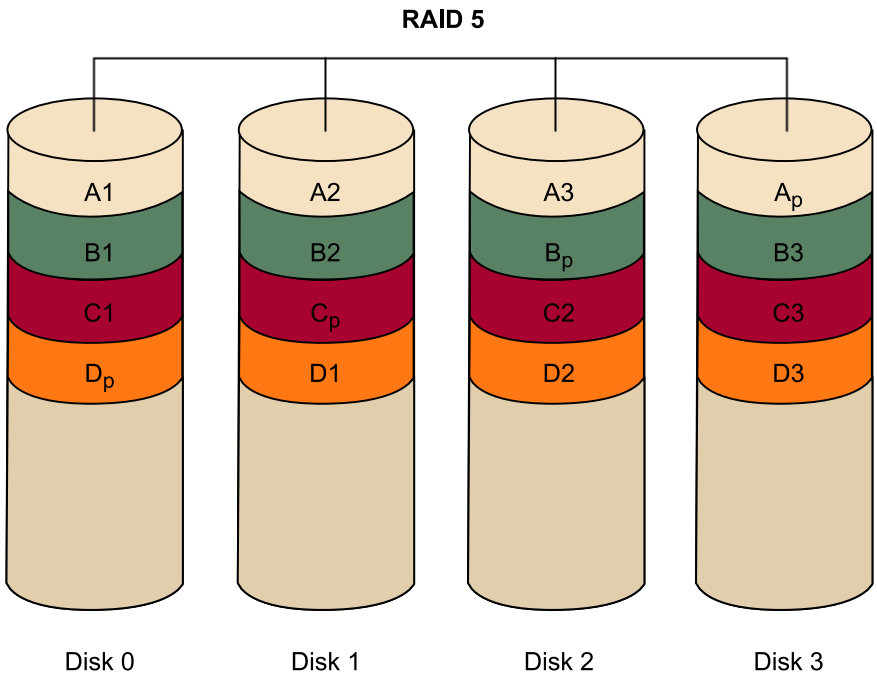
en uno de ellos, podemos sustituirlo, sin apagar el sistema, por un disco nuevo.



- **RAID 2**: en los anteriores se dividen los datos en bloques a repartir. Aquí se divide en bits y se utilizan códigos de redundancia para la corrección de datos. No es prácticamente utilizado, a pesar de las altas prestaciones que alcanzaría, ya que necesita idealmente un número muy alto de discos, uno por bit de datos y varios para el cálculo de la redundancia (por ejemplo, en un sistema de 32 bits, llegaría a usar 39 discos).

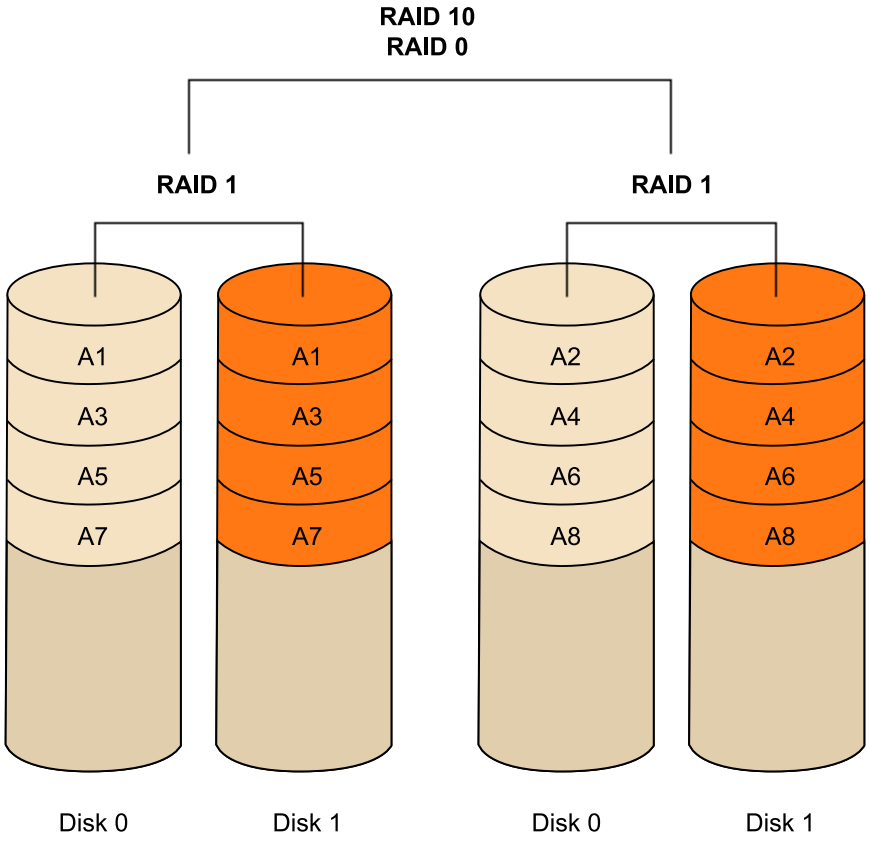
- **RAID 3**: utiliza división en bytes con un disco dedicado a la paridad de los bloques. Tampoco es muy utilizada, ya que según el tamaño de los datos y posiciones no permite accesos simultáneos. RAID 4 es semejante, aunque divide a nivel de bloques en lugar de bytes, lo que permite que sí se puedan servir peticiones simultáneas cuando se solicita un único bloque.

- **RAID 5**: Se usa división a nivel de bloques, distribuyendo la paridad entre los discos. Tiene amplio uso, debido al esquema sencillo de paridad, y a que este cálculo se implementa de forma sencilla por hardware, con buenas prestaciones.



- **RAID 0+1 (o 01)**: un *mirror* de divisiones es un nivel de RAID anidado. Se implementan, por ejemplo, dos grupos de RAID 0, los cuales son usados en RAID 1 para crear *mirror* entre ellos. Una ventaja es que, en caso de fallo, puede reconstruirse el nivel de RAID 0 usado gracias a la otra copia, pero si quieren añadirse discos hay que añadirlos a todos los grupos de RAID 0 de igual forma.

- **RAID 10 (1+0)**: división de *mirrors*, grupos de RAID 1 bajo RAID 0. Así, en cada grupo de RAID 1 puede llegar a fallar un disco sin que se pierdan datos. Claro que esto obliga a reemplazarlos, ya que si no el disco que quede en el grupo se convierte en posible punto de fallo de todo el sistema. Es una configuración que suele usarse para base de datos de altas prestaciones (por la tolerancia a fallos y la velocidad, al no estar basada en cálculos de paridad).



Algunas consideraciones a tener en cuenta sobre RAID en general:

- RAID mejora el *uptime* del sistema, ya que algunos de los niveles permiten que discos fallen y el sistema siga siendo consistente, y dependiendo del hardware, incluso puede cambiarse el hardware problemático en caliente sin necesidad de parar el sistema, cuestión especialmente importante en sistema críticos.

- RAID puede mejorar el rendimiento de las aplicaciones; en especial, en los sistemas con implementaciones de *mirror* es posible que la división de datos permita que las operaciones lineales de lectura se incrementen significativamente, debido a la posibilidad de que los discos ofrezcan, de modo simultáneo, partes de esta lectura, aumentando la tasa de transferencia de datos.

- RAID no protege los datos; evidentemente, la destrucción por otros medios (virus, mal funcionamiento general o desastres naturales) no está protegida. Hemos de basarnos en esquemas de copias de seguridad. Tengamos en cuenta que algunos esquemas protegen contra uno o dos fallos de discos del array, pero si hay más o dependiendo del esquema, directamente (caso RAID 0) se perderán datos.

- No se simplifica la recuperación de datos; si un disco pertenece a un array RAID, tiene que intentar recuperarse en ese ambiente. Se necesita software específico o los controladores hardware para acceder a los datos.

- Por el contrario, no suele mejorar aplicaciones típicas de usuario –sean de escritorio– debido a que estas aplicaciones tienen componentes altos de acceso aleatorio a datos, o a conjuntos de datos pequeños; puede que no se beneficien de lecturas lineales o de transferencias de datos sostenidas. En estos ambientes, es posible que no se note apenas mejoría de prestaciones.

- Algunos esquemas aumentan de velocidad las operaciones de lectura, pero por otra parte penalizan las de escritura (caso RAID 5 por el cálculo de paridad que hay que escribir). Si el uso es básicamente de escritura, habrá que buscar qué esquemas no penalizan o consiguen el ratio de escritura que necesitamos (algunos casos, como RAID 0,1, o algunas modalidades de RAID 10 son equivalentes a escribir en un disco único, o incluso aumentan las prestaciones en este sentido).

- No se facilita el traslado de información; sin RAID es bastante fácil trasladar datos, simplemente moviendo el disco de un sistema a otro. En el caso de RAID es casi imposible (a no ser que dispongamos del mismo hardware controlador) mover un array de discos a otro sistema.

En el caso de GNU/Linux, se da soporte al hardware RAID mediante varios módulos de *kernel*, asociados a diferentes conjuntos de fabricantes o circuitos base, *chipsets*, de estas controladoras RAID. Se permite así al sistema abstraerse de los mecanismos hardware y hacerlos transparentes al sistema y al usuario final. Así, estos módulos de *kernel* nos permiten el acceso a los detalles de estas controladoras, y a su configuración de parámetros de muy bajo nivel, que en algunos casos (especialmente en servidores que soportan carga elevada de E/S), pueden ser interesantes para procesos de *tunning* del sistema de discos que use el servidor. Se busca maximizar las prestaciones del sistema.

La otra posibilidad que analizaremos aquí es la realización de estos procesos mediante componentes software, en concreto el componente software RAID de GNU/Linux.

GNU/Linux dispone en *kernel* del *driver* llamado Multiple Device (*md*), que podemos considerar como el soporte a nivel *kernel* para RAID. Mediante este *driver* podemos implementar niveles de RAID generalmente 0,1,4,5,6 y anidados (por ejemplo, RAID 10) sobre diferentes dispositivos de bloque como discos IDE, SATA o SCSI. También dispone del nivel *linear*, como nivel donde se produce una combinación lineal de los discos disponibles (donde no importa que sean de diferentes tamaños), de manera que se escribe consecutivamente en los discos.

Para la utilización del RAID software en Linux, debemos disponer del soporte RAID en el *kernel*, y en su caso los módulos *md* activos (además de algunos *drivers* específicos según el nivel, ved *drivers* disponibles asociados a RAID, por ejemplo en Debian con *modconf*). El método preferido para la implementación de arrays de discos RAID, mediante el software RAID ofrecido por Linux, es mediante el proceso de instalación del sistema inicial, o bien mediante la utilidad *mdadm*. Esta utilidad nos permite crear los arrays y gestionarlos.

Webs recomendadas

Para consultar conceptos de software RAID, ved Linux RAID wiki:

https://raid.wiki.kernel.org/index.php/Linux_Raid

http://www.linuxfoundation.org/collaborate/workgroups/linux-raid

Ved también niveles RAID soportados:

https://raid.wiki.kernel.org/index.php/Introduction#The_RAID_levels

Observemos algunos casos prácticos. Supongamos unos discos SCSI */dev/sda, /dev/sdb*... en los cuales disponemos de varias particiones disponibles para implementar RAID:

Creación de un array *linear:*

```
# mdadm -create -verbose /dev/md0 -level=linear -raid-devices=2 /dev/sda1 /dev/sdb1
```

donde se genera un array *linear* a partir de las particiones primeras de */dev/sda* y */dev/sdb*, creando el nuevo dispositivo */dev/md0*, que ya puede ser usado como nuevo disco (suponiendo que exista el punto de montaje */media/discoRAID*):

```
# mkfs.ext2fs /dev/md0
# mount /dev/md0 /media/discoRAID
```

Para un RAID0 o RAID1 podemos cambiar simplemente el nivel (*-level*) a *raid0* o *raid1*. Con *mdadm –detail /dev/md0* podremos comprobar los parámetros del nuevo array creado.

También podemos consultar la entrada *mdstat* en */proc* para determinar los arrays activos, así como sus parámetros. En especial, con los casos con *mirror* (por ejemplo, en los niveles 1, 5...) podremos observar en su creación la construcción inicial del array de las copias, en */proc/mdstat* indicará el nivel de reconstrucción (y el tiempo aproximado de finalización).

mdadm dispone de muchas opciones que nos permiten examinar y gestionar los diferentes arrays RAID software creados (podemos ver una descripción y ejemplos en *man mdadm*).

Otra cuestión importante es que los arrays RAID por software no son automáticamente reconocidos en arranque (como sí pasa con el RAID hardware soportado), ya que de hecho dependen de la construcción con *mdadm*. Para que la definición de un array software sea persistente, hace falta registrarla en el fichero de configuración */etc/mdadm.conf* (la ubicación puede depender de la distribución). Un paso sencillo es crearlo automáticamente a partir del resultado de la orden.

```
mdadm -detail -scan (puede añadirse tambien --verbose)
```

Otra consideración importante son las optimizaciones a que se pueden someter los arrays RAID para mejorar su rendimiento, tanto por monitorizar su comportamiento como por optimizar parámetros del sistema de ficheros, para realizar un uso más efectivo de los niveles RAID y sus características.

Nota

La optimización de los arrays RAID puede ser una fuente importante de sintonización del sistema. Examinad algunas cuestiones en https://raid.wiki.kernel.org/index.php/Performance o en la propia página man de *mdadm*.

Pasaremos a detallar otro caso de configuración RAID más elaborado basado en la elaboración de un RAID 5.

Este nivel RAID (de como mínimo tres discos) presenta una configuración distribuida de datos y pariedad en los discos que forman el array. Se presenta muy buen rendimiento en lectura, pero disminuye la escritura (respecto a un único disco) ya que hay que calcular la pariedad y distribuirla entre los discos (en casos RAID hardware, este caso de cálculo de pariedad se accelera mediante hardware). Otro dato a tener en cuenta es que la capacidad final obtenida es la suma de N-1 discos, y esta configuración es resistente a un fallo de disco presente en el array, el cual puede reemplazarse y volver a reconstruir el array. El nivel no soporta dos fallos de disco (probabilidad que de hecho puede aumentar si en el array RAID se integra un número elevado de discos), con lo cual, como ya comentamos, disponer de RAID no nos inhibe del proceso de *backup* de datos. Si quisiéramos disponer de más fiabilidad, podríamos movernos a RAID 6, que posee cálculo con pariedad dual, a costa de bajar rendimiento, o a configuraciones con RAID 10, que tienen un buen compromiso entre rendimiento y fiabilidad.

Realizaremos, en este caso, el proceso de construcción con cuatro discos SATA de alta capacidad 1,5TB, obteniendo un almacenamiento final de 4,5TB (aproximadamente). El array RAID es complementario al sistema existente; en casos que tenga que incluir particiones de *boot*, el proceso es más delicado (ved las recomendaciones de la RAID software wiki, comentada anteriormente).

Utilizaremos cuatro discos SATA presentes en el sistema como */dev/sdb*, *sdc*, *sdd*, *sde*. El disco */dev/sda* se supone que incluye el sistema y las particiones de arranque, y no forma parte del RAID. Primero hemos de pasar por la inicializacion de los discos (este proceso borra cualquier contenido previo); definiremos una partición única (en este caso, vamos a integrar todo el almacenamiento disponible; en otros esquemas podrían realizarse varias particiones), creada con *fdisk*. Fundamentalmente, seleccionamos nueva particion prima-

ria, requisitos por defecto y cambiamos el tipo de partición a "*Linux Raid autodetect*" (código *fd*), haciendo el proceso con *fdisk*, sea *x* cada disco diferente del array, y el paréntesis incluye las opciones mediante teclado de *fdisk*:

```
# fdisk /dev/sdx            (d n p 1 ENTER ENTER t fd w)
```

(para nuestro caso 4 veces con *x=b,c,d,e*)

Respecto a esta inicialización, hay que tener cuidado con los tamaños de disco, ya que *fdisk* con particiones tipo *msdos* sólo soporta particiones hasta 2TB. Si los discos fueran mayores deberíamos movernos a otras herramientas de particionado, como *gparted/parted,* que soportan nuevos tipos de particiones como GPT, que ya no disponen de estas restricciones de tamaño.

Una vez realizada esta inicialización, ya podemos pasar a construir el array:

```
# mdadm --create /dev/md0 --level=5 --verbose --force --chunk=512
--raid-devices=4 /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1
```

Un parámetro importante en RAID más avanzados es el tamaño del parámetro *chunk* (por defecto, 64 unidad en kb), que tiene especial relevancia en las prestaciones del RAID. Existen estudios que determinan tamaños adecuados de *chunk* en función de la funcionalidad final del RAID (escritura o lectura, tamaño promedio de los accesos, acceso principalmente aleatorio o secuencial, etc.). En general, en RAID5 se recomiendan tamaños de 128 en adelante, e incluso más elevados 512-1024, dependiendo del tamaño promedio de los archivos.

Una vez lanzado el comando anterior, comenzará la construcción del RAID en segundo plano. Podemos ir consultando */proc/mdstat*, que nos mencionará la velocidad de construcción y el porcentaje, así como una estimación de tiempo para finalizar. Los tiempos de construcción son bastante grandes para altas capacidades de espacio, y pueden ir desde algunas horas hasta días, dependiendo de las capacidades de los discos y máquina.

Podríamos estar viendo en */proc/mdstat*:

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdb1[0] sde1[3] sdd1[2] sdc1[1]
   4395406848 blocks level 5, 512k chunk, algorithm 2 [4/3] [UUU_]
   [==>..................] recovery = 12.6% (37043392/292945152) finish=127.5min
speed=33440K/sec

unused devices: <none>
```

Y una vez finalizado:

```
# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid5 sdb1[0] sde1[3] sdd1[2] sdc1[1]
   4395406848 blocks level 5, 512k chunk, algorithm 2 [4/4] [UUUU]

unused devices: <none>
```

El siguiente paso es crear un *filesystem* en nuestro recientemente creado array, visto por el sistema como el dispositivo */dev/md0* (este paso podría iniciarse simultáneamente durante la reconstrucción, pero por velocidad y seguridad en algunos pasos críticos iniciales se recomienda realizarlo al final del proceso).

En nuestro caso, vamos a optimizar ligeramente esta creación examinando algunos parámetros; crearemos un *filesystem ext3* en el array completo (es recomendable usar un gestor de volúmenes primero, como LVM, que veremos a continuación, porque nos permitirá crear varias particiones lógicas, que puedan en el futuro expandirse o contraerse según las necesidades).

Para la creación del *filesystem* hay algunas recomendaciones generales que provienen a partir del tamaño de *chunk* (número de datos consecutivos que reside en un disco en KB), que de hecho nos define cómo se accede al array, y de los discos presentes, un posible esquema de creación de un *filesystem ext3*:

```
mkfs.ext3 -v -b 4096 -E stride=128,stripe-width=384 /dev/md0
```

Donde *-b 4096* es el tamaño de bloque de disco, recomendado para *filesystems* muy grandes, y *-E* define opciones del *filesystem* para optimizar su rendimiento. Estas opciones pueden variarse después con el comando *tune2fs*.

En estas opciones suele sugerirse un cálculo relacionado con el *chunk*, que puede ser en nuestro ejemplo:

- *chunk size* = 512kB (colocado en el comando *mdadm* en la creación del array)
- *block size* = 4kB (recomendado para grandes *filesystems*)
- *stride* = *chunk / block* = 512kB / 4k = 128kB
- *stripe-width* = *stride* * ((*n* discos en el raid5) - 1) = 128kB * ((4) - 1) = 128kB * 3 = 384kB

Esto nos permite ajustar los parámetros del sistema de ficheros al uso del *chunk* escogido.

Una vez creado el sistema de ficheros, ya tenemos el array disponible para que pueda ser montado en una ubicación del sistema (suponiendo un directorio previo */mnt/raid5*):

```
# mount -t ext3 /mnt/raid5 /dev/md0
```

Y ya lo tenemos disponible en el sistema. Si queremos hacerlo fijo, recordad introducir los parámetros en */etc/fstab* para que se monte en arranque. Y en este caso de array software es recomendable (aunque algunos *kernels* recientes ya soportan autodetección de RAID software), colocar la información RAID en el fichero */etc/mdadm.conf*, mediante consulta del identificador del array por medio de los comandos *# mdadm –detail –scan* (con o sin *–verbose* si se necesita la identificación de dispositivos):

```
#mdadm --detail --scan --verbose
ARRAY        /dev/md0         level=raid5        num-devices=4         metadata=0.90
UUID=dcc77e17:94093185:66731ad6:6353ec0b
  devices=/dev/sdb1,/dev/sdc1,/dev/sdd1,/dev/sde1
```

En el siguiente reinicio del sistema ya dispondremos del array montado en arranque. También, como punto final, podemos obtener la información del array con *mdadm –detail*:

```
# mdadm --detail /dev/md0
/dev/md0:
   Version : 0.90
 Creation Time : Sat May 8 09:33:06 2010
   Raid Level : raid5
   Array Size : 4395406848 (4191.79 GiB 4500.90 GB)
 Used Dev Size : 1465135616 (1397.26 GiB 1500.30 GB)
   Raid Devices : 4
  Total Devices : 4
Preferred Minor : 0
   Persistence : Superblock is persistent

   Update Time : Fri May 21 12:14:31 2010
      State : clean
  Active Devices : 4
Working Devices : 4
   Failed Devices : 0
   Spare Devices : 0
   Layout : left-symmetric
   Chunk Size : 512K

   UUID : dcc77e17:94093185:66731ad6:6353ec0b (local to host kaoscore)
   Events : 0.125
```

```
    Number  Major  Minor  RaidDevice  State
     0       8      17         0      active sync  /dev/sdb1
     1       8      33         1      active sync  /dev/sdc1
     2       8      49         2      active sync  /dev/sdd1
     3       8      65         3      active sync  /dev/sde1
```

Respecto al funcionamiento del sistema, hay que examinar tanto este informe como el proporcionado por */proc/mdstat* para controlar de forma periódica (manualmente, mediante *cron*, o por notificaciones mediante mail) el estado del RAID (en este caso activo), para determinar si se produce algun fallo de disco. En este estado el array pasaría a *degraded,* y el sistema perdería su capacidad de tolerancia a un siguiente fallo. Es entonces necesario detectar qué disco está fallando y sustituirlo, para que comience la reconstrucción del array (mediante *mdadm* puede eliminarse un disco del array, y añadir una vez realizado el cambio de hardware el nuevo disco). También es posible mediante *mdadm* simular degradaciones o fallos, lo que nos permite testear nuestro array delante de condiciones extremas.

Finalmente, destacamos que la creación de arrays RAID es muy interesante para la realización de grandes soportes de discos para entornos empresariales donde se intenta maximizar la tolerancia a fallos, la recuperación rápida delante de problemas y un soporte continuo de servicios sin interrupciones. Es un campo también donde se necesita un entorno cuidadoso de análisis de rendimiento de las soluciones, de los requisitos iniciales del servicio y experimentación con las diferentes soluciones posibles.

8.2. Volúmenes lógicos (LVM)

En un momento determinado, surge la necesidad de abstraerse del sistema físico de discos, y de su configuración y número de dispositivos, para que el sistema (operativo) se encargue de este trabajo y no nos tengamos que preocupar de estos parámetros directamente. En este sentido, puede verse el sistema de volúmenes lógicos como una capa de virtualización del almacenamiento que permite una visión más simple que facilite la utilización fluida y sencilla.

En el *kernel* Linux se dispone de LVM (*logical volume manager*), que se basó en ideas desarrolladas de gestores de volúmenes de almacenamiento usados en HP-UX (una versión UNIX propietaria de HP). Actualmente existen dos versiones, de las que la LVM2 es la más utilizada, por una serie de prestaciones añadidas que incorpora.

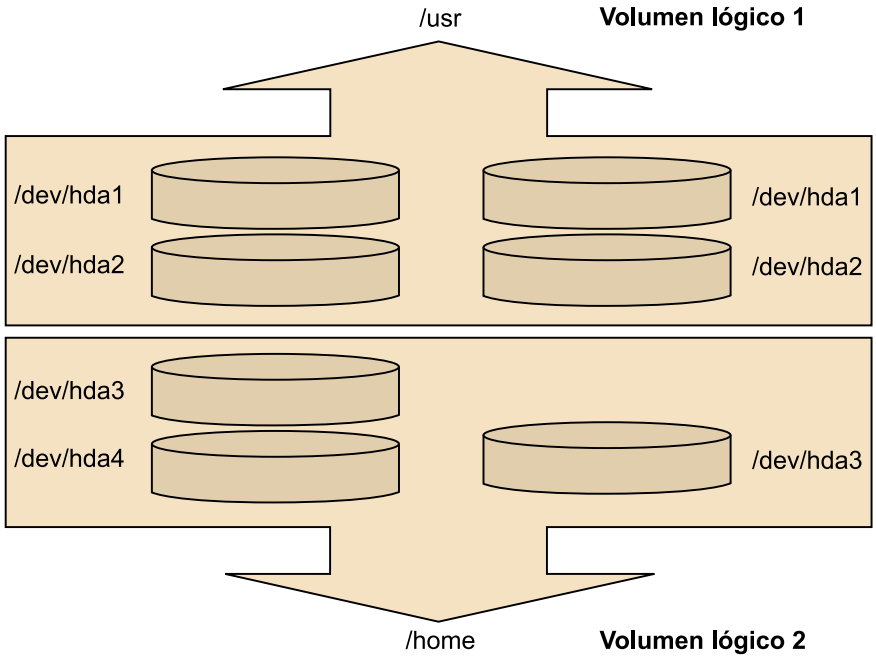
La arquitectura de una LVM consiste típicamente en los componentes (principales):

1) Volúmenes físicos (PV): son los discos duros, o particiones de éstos, o cualquier otro elemento que aparezca como un disco duro de cara al sistema (por ejemplo, un RAID software o hardware).

2) Volúmenes lógicos (LV): es el equivalente a la partición del disco físico. Esta LV es visible en el sistema como un dispositivo de bloques (absolutamente equivalente a una partición física), y puede contener un sistema de ficheros (por ejemplo, el */home* de los usuarios). Los volúmenes tienen más sentido para los administradores, ya que pueden usarse nombres para identificarlos (así podemos utilizar un dispositivo lógico, llamado "*stock*" o "*marketing*" en lugar de *hda6* o *sdc3*).

3) Grupos de volúmenes (VG): es el elemento de la capa superior. La unidad administrativa que engloba nuestros recursos, ya sean volúmenes lógicos (LV) o físicos (PV). En esta unidad se guardan los datos de los PV disponibles, y cómo se forman las LV a partir de los PV. Evidentemente, para poder utilizar un grupo VG, hemos de disponer de soportes físicos PV, que se organizen en diferentes unidades lógicas LV.

En la siguiente figura observamos un grupo de volúmenes, donde disponemos de siete PV, en forma de particiones de discos, que se han agrupado para formar dos volúmenes lógicos (que se han acabado utilizando para formar los sistemas de ficheros de */usr* y */home*):



Esquema de un ejemplo de LVM.

Con el uso de los volúmenes lógicos, permitimos un tratamiento más flexible del espacio en el sistema de almacenamiento (que podría tener un gran número de discos y particiones diferentes), según las necesidades que nos aparezcan.

Podemos gestionar el espacio, tanto por identificadores más adecuados como por operaciones que nos permitan adecuar las necesidades al espacio disponible en cada momento.

Los sistemas de gestión de volúmenes nos permiten:

1) Redimensionar dinámicamente grupos y volúmenes lógicos, aprovechando nuevos PV, o extrayendo algunos de los disponibles inicialmente.

2) Instantáneas del sistema de archivos (lectura en LVM1 y lectura y/o escritura en LVM2). Esto facilita la creación de un nuevo dispositivo que sea una instantánea en el tiempo de la situación de una LV. Se puede, por ejemplo, crear la instantánea, montarla, probar varias operaciones o configuración nueva de software, u otros elementos, y si no funciona como esperábamos, devolver el volumen original a su estado antes de las pruebas.

3) RAID 0 de volúmenes lógicos.

En LVM no se implementan configuraciones de RAID de tipos 1 o 5, si son necesarias (o sea redundancia y tolerancia a fallo). El proceso es utilizar software de RAID o controladora hardware RAID que lo implemente en un determinado nivel de RAID, y luego colocar LVM como capa superior al RAID creado previamente.

Hagamos un breve ejemplo de creación típica (en muchos casos, el instalador de la distribución realiza un proceso parecido, si permitimos un LVM como sistema inicial de almacenamiento). Básicamente, se realiza: 1) la creación de los volumenes físicos (PV), 2) creación del grupo lógico (VG), 3) creación del volumen lógico (LV), y 4) utilización para creación de un sistema de ficheros, y posterior montaje:

1) Disponemos, por ejemplo, de tres particiones de diferentes discos. Creamos tres PV (los pasos iniciales borran cualquier contenido de los discos) e inicializamos el contenido:

```
# dd if=/dev/zero of=/dev/hda1 bs=1k count=1
# dd if=/dev/zero of=/dev/hda2 bs=1k count=1
# dd if=/dev/zero of=/dev/hdb1 bs=1k count=1

# pvcreate /dev/hda1
Physical volume "/dev/hda1" successfully created
# pvcreate /dev/hda2
Physical volume "/dev/hda2" successfully created
# pvcreate /dev/hdb1
Physical volume "/dev/hdb1" successfully created
```

2) Colocamos en un VG creado de los diferentes PV:

```
# vgcreate grupo_discos /dev/hda1 /dev/hda2 /dev/hdb1
Volume group "grupo_discos" successfully created
```

3) Creamos la LV (en este caso, de tamaño de 1GB) a partir de los elementos que tenemos en el grupo VG (*-n* indica el nombre del volumen):

```
# lvcreate -L1G -n volumen_logico grupo_discos
lvcreate -- doing automatic backup of "grupo_discos"
lvcreate -- logical volume "/dev/grupo_discos/volumen_logico" successfully created
```

4) Y finalmente, creamos un sistema de ficheros (un *reiser* en este caso):

```
# mkfs.reiserfs /dev/grupo_discos/volumen_logico
```

El cual, por ejemplo, podríamos colocar de espacio de *backup*:

```
mkdir /mnt/backup
mount -t reiserfs /dev/grupo_discos/volumen_logico /mnt/backup
```

Y disponemos finalmente del dispositivo como un volumen lógico que implementa un sistema de ficheros.

9. Software: actualización

Ved también

En el apartado 1, "Herramientas básicas para el administrador", desarrollamos en profundidad estos conceptos.

Para la administración de instalación o actualización de software en nuestro sistema, vamos a depender en primera instancia del tipo de paquetes software que utilice nuestro sistema:

- **RPM**: paquetes que utiliza la distribución Fedora/Red Hat (y derivadas). Se suelen manejar mediante el comando rpm. Contienen información de dependencias del software con otros. A alto nivel mediante yum (o *up2date* en algunas distribuciones derivadas de Red Hat).

- **DEB**: paquetes de Debian, se suelen manejar con un conjunto de herramientas que trabajan a diferentes niveles con paquetes individuales o grupos. Entre estas, cabe mencionar *dselect, tasksel, dpkg, apt-get* y *aptitude*.

- **Tar**, o bien los tgz (también *tar.gz*): son puramente paquetes de ficheros unidos y comprimidos mediante comandos estándar como *tar* y *gzip* (se usan los mismos para la descompresión). Estos paquetes no contienen información de dependencias y pueden instalarse en diferentes lugares, si no es que llevan información de ruta (*path*) absoluta.

Para manejar estos paquetes existen varias herramientas gráficas, como RPM: *Kpackage;* DEB: *Synaptic, Gnome*-apt; Tgz: *Kpackage* o desde el propio gestor de ficheros gráficos (en Gnome o KDE). También suelen existir utilidades de conversiones de paquetes. Por ejemplo, en Debian tenemos el comando *alien,* que permite convertir paquetes RPM a DEB. Aunque hay que tomar las debidas precauciones para que el paquete, por tener una distribución destino diferente, no modifique algún comportamiento o fichero de sistema no esperado.

Dependiendo del uso de los tipos de paquetes o herramientas, la actualización o instalación de software de nuestro sistema se podrá producir de diferentes maneras:

1) Desde los propios CD de instalación del sistema. Todas las distribuciones buscan el software en sus CD. Pero hay que tener en cuenta que este software no sea antiguo y no incluya, por esta razón, algunos parches como actualizaciones, o nuevas versiones con más prestaciones, con lo cual, si se instala a partir de CD, es bastante común verificar después que no exista alguna versión más reciente.

2) Mediante servicios de actualización o búsqueda de software, ya sea de forma gratuita, como el caso de la herramienta *apt-get* de Debian, o yum en Fedora, o servicios de suscripción (de pago o con facilidades básicas), como el Red Hat Network de las versiones Red Hat comerciales.

3) Por repositorios de software que ofrecen paquetes de software preconstruidos para una distribución determinada.

4) Por el propio creador o distribuidor del software, que ofrece una serie de paquetes de instalación de su software. Podemos no encontrar el tipo de paquetes necesario para nuestra distribución.

5) Software sin empaquetamiento o con un empaquetamiento de sólo compresión sin ningún tipo de dependencias.

6) Sólo código fuente, en forma de paquete o bien fichero comprimido.

10. Trabajos no interactivos

En las tareas de administración, suele ser necesaria la ejecución a intervalos temporales de ciertas tareas, ya sea por programar las tareas para realizarlas en horarios de menor uso de la máquina, o bien por la propia naturaleza periódica de las tareas que se quieran desarrollar.

Para realizar este tipo de trabajos "fuera de horas", como servicios periódicos o programados, hay varios sistemas que nos permiten construir un tipo de agenda de tareas (planificación de ejecución de tareas):

- ***nohup***: es quizás el caso más simple utilizado por los usuarios. Les permite la ejecución de una tarea no interactiva una vez hayan salido de su cuenta. Al salir de la cuenta, el usuario pierde sus procesos, y *nohup* permite dejarlos en ejecución, a pesar de que el usuario se desconecte.

- ***at***: nos deja lanzar una acción para más tarde, programando un determinado instante en el que va a iniciarse, especificándose la hora (*hh:mm*) y fecha, o bien si se hará hoy (*today*) o mañana (*tomorrow*).

```
# at 10pm tarea
```

realizar la tarea a las diez de la noche.

```
# at 2am tomorrow tarea
```

realizar la tarea a las dos de la madrugada.

- ***cron***: permite establecer una lista de trabajos por realizar con su programación; esta configuración se guarda en */etc/crontab*. Concretamente, en cada entrada de este fichero tenemos: minutos y hora en que se va a efectuar la tarea, qué día del mes, qué mes, qué día de la semana, junto con qué (ya sea una tarea, o bien un directorio donde estarán las tareas a ejecutar).

  De forma estándar, el contenido es parecido a:

```
25 6 * * * root test -e /usr/sbin/anacron ||
run-parts --report /etc/cron.daily
47 6 * * 7 root test -e /usr/sbin/anacron ||
run-parts --report /etc/cron.weekly
52 6 1 * * root test -e /usr/sbin/anacron ||
run-parts --report /etc/cron.monthl
```

  donde se está programando que una serie de tareas van a hacerse: cada día ("*" indica "cualquiera"), semanalmente (el séptimo día de la semana), o mensualmente (el primero de cada mes). Los trabajos serían ejecutados por el comando *crontab,* pero el sistema *cron* supone que la máquina está siempre encendida. Si no es así, es mejor utilizar *anacron,* que verifica si la acción no se realizó cuando habría debido hacerlo, y la ejecuta. En cada línea del anterior fichero se verifica que exista el comando *anacron* y se ejecutan los *scripts* asociados a cada acción, que en este caso están guardados en unos directorios asignados para ello, los *cron*.

También pueden existir unos ficheros *cron.allow, cron.deny*, para limitar quién puede colocar (o no) trabajos en *cron*. Mediante el comando *crontab,* un usuario puede definir trabajos en el mismo formato que hemos visto antes, que se guardarán en */var/spool/cron/crontabs*. En algunos casos, existe también un directorio */etc/cron.d* donde se pueden colocar trabajos y que es tratado como si fuera una extensión del fichero */etc/crontab*. En algunas versiones, el sistema ya especifica sus trabajos periódicos de sistema directamente sobre subdirectorios del */etc* como *cron.hourly/, cron.daily/, cron.weekly y cron.montly/,* donde se colocan los trabajos de sistema que necesitan esta periodicidad.

11. Taller: prácticas combinadas de los apartados

Comenzaremos por examinar el estado general de nuestro sistema. Vamos a realizar los diferentes pasos en un sistema Debian. Aunque se trata de un sistema Debian, los procedimientos son en su mayor parte trasladables a otras distribuciones, como Fedora/Red Hat (mencionaremos algunos de los cambios más importantes). El hardware consiste en un Pentium 4 a 2.66Mhz con 768 MB y varios discos, DVD y grabador de CD, además de otros periféricos, pero ya iremos obteniendo esta información paso a paso.

Veamos primero cómo ha arrancado nuestro sistema la última vez:

```
# uptime

17:38:22 up 2:46, 5 users, load average: 0.05, 0.03, 0.04
```

Este comando nos da el tiempo que lleva el sistema funcionando desde que se arrancó la última vez, 2 horas 46 minutos. En nuestro caso tenemos cinco usuarios. Éstos no tienen por qué ser cinco usuarios diferentes, sino que serán las sesiones de usuario abiertas (por ejemplo, mediante un terminal). El comando *who* permite listar estos usuarios. El *load average* es la carga media del sistema en los últimos 1, 5 y 15 minutos.

Veamos el log del arranque del sistema (comando *dmesg*), las líneas que se iban generando en la carga del sistema (se han suprimido diferentes líneas por claridad):

```
Linux version 2.6.20-1-686 (Debian 2.6.20-2) (waldi@debian.org) (gcc version
4.1.2 20061115 (prerelease) (Debian 4.1.1-21)) #1 SMP Sun Apr 15 21:03:57 UTC
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
 BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000ce000 - 00000000000d0000 (reserved)
 BIOS-e820: 00000000000dc000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 000000002f6e0000 (usable)
 BIOS-e820: 000000002f6e0000 - 000000002f6f0000 (ACPI data)
 BIOS-e820: 000000002f6f0000 - 000000002f700000 (ACPI NVS)
 BIOS-e820: 000000002f700000 - 000000002f780000 (usable)
 BIOS-e820: 000000002f780000 - 0000000030000000 (reserved)
 BIOS-e820: 00000000ff800000 - 00000000ffc00000 (reserved)
BIOS-e820: 00000000fffffc00 - 0000000100000000 (reserved)
0MB HIGHMEM available.
```

```
759MB LOWMEM available.
```

Estas primeras líneas ya nos indican varios datos interesantes: la versión del *kernel* Linux es la 2.6.20-1-686, una versión 2.6 revisión 20 a revisión 1 de Debian, y para máquinas 686 (arquitectura Intel x86 32bits). Indica también que estamos arrancando un sistema Debian, con este *kernel* que fue compilado con un compilador GNU gcc versión 4.1.2. A continuación, existe un mapa de zonas de memoria usadas (reservadas) por la BIOS, y a continuación el total de memoria detectada en la máquina: 759 MB, a las que habría que sumar el primer 1 MB, total de 760 MB.

```
Kernel command line: BOOT_IMAGE=LinuxNEW ro root=302 lang=es acpi=force
Initializing CPU#0
Console: colour dummy device 80x25
Memory: 766132k/777728k available (1641k kernel code, 10968k reserved, 619k data,
208k init, 0k highmem)
Calibrating delay using timer specific routine.. 5320.63 BogoMIPS (lpj=10641275)
```

Aquí nos refiere cómo ha sido el arranque de la máquina, qué línea de comandos se le ha pasado al *kernel* (pueden pasarse diferentes opciones, por ejemplo desde el *lilo* o *grub*). Y estamos arrancando en modo consola de 80 x 25 caracteres (esto se puede cambiar). Los *BogoMIPS* son una medida interna del *kernel* de la velocidad de la CPU; hay arquitecturas donde es difícil detectar a cuántos Mhz o GHz funciona la CPU, y por eso se utiliza esta medida de velocidad. Después nos da más datos de la memoria principal, y nos dice para qué se está usando en este momento del arranque.

```
CPU: Trace cache: 12K uops, L1 D cache: 8K
CPU: L2 cache: 512K
CPU: Hyper-Threading is disabled
Intel machine check architecture supported.
Intel machine check reporting enabled on CPU#0.
CPU0: Intel P4/Xeon Extended MCE MSRs (12) available
CPU0: Intel(R) Pentium(R) 4 CPU 2.66GHz stepping 09
```

Además, nos proporciona datos varios de la CPU, el tamaño de las caché de primer nivel, caché interna CPU, L1 dividida en una *TraceCache* del Pentium4 (o *Instruction Cache*), y la caché de datos, y caché unificada de segundo nivel (*L2*), tipo de CPU, velocidad de ésta y del bus del sistema.

```
PCI: PCI BIOS revision 2.10 entry at 0xfd994, last bus=3
Setting up standard PCI resources
...
NET: Registered protocol
IP route cache hash table entries: 32768 (order: 5, 131072 bytes)
TCP: Hash tables configured (established 131072 bind 65536)
checking if image is initramfs... it is
```

```
Freeing initrd memory: 1270k freed
fb0: VESA VGA frame buffer device
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing enabled
serial8250: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
00:09: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
RAMDISK driver initialized: 16 RAM disks of 8192K size 1024 blocksize
PNP: PS/2 Controller [PNP0303:KBC0,PNP0f13:MSE0] at 0x60,0x64 irq 1,12
i8042.c: Detected active multiplexing controller, rev 1.1.
serio: i8042 KBD port at 0x60,0x64 irq 1
serio: i8042 AUX0 port at 0x60,0x64 irq 12
serio: i8042 AUX1 port at 0x60,0x64 irq 12
serio: i8042 AUX2 port at 0x60,0x64 irq 12
serio: i8042 AUX3 port at 0x60,0x64 irq 12
mice: PS/2 mouse device common for all mice
```

Continúan las inicializaciones del *kernel* y dispositivos, menciona la inicialización de protocolos de red. Los terminales, los puertos serie *ttyS0* (sería el *com1*), *ttyS01* (el *com2*), da información de los discos RAM que usamos y detecta dispositivos PS2, teclado y ratón.

```
ICH4: IDE controller at PCI slot 0000:00:1f.1


   ide0: BM-DMA at 0x1860-0x1867, BIOS settings: hda:DMA, hdb:pio
   ide1: BM-DMA at 0x1868-0x186f, BIOS settings: hdc:DMA, hdd:pio
Probing IDE interface ide0...
hda: FUJITSU MHT2030AT, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
Probing IDE interface ide1...
hdc: SAMSUNG CDRW/DVD SN-324F, ATAPI CD/DVD-ROM drive
ide1 at 0x170-0x177,0x376 on irq 15
SCSI subsystem initialized
libata version 2.00 loaded.
hda: max request size: 128KiB
hda: 58605120 sectors (30005 MB) w/2048KiB Cache, CHS=58140/16/63<6>hda: hw_config=600b
, UDMA(100)
hda: cache flushes supported
hda: hda1 hda2 hda3
kjournald starting. Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
hdc: ATAPI 24X DVD-ROM CD-R/RW drive, 2048kB Cache, UDMA(33)
Uniform CD-ROM driver Revision: 3.20
Addinf 618492 swap on /dev/hda3.
```

Detección de dispositivos IDE, detecta el chip IDE en el bus PCI e informa de que está controlando dos dispositivos: *hda* y *hdc*, que son respectivamente un disco duro (fujitsu), un segundo disco duro, un DVD Samsung y una grabadora de CD (ya que en este caso se trata de una unidad combo). Indica particiones

activas en el disco. Más adelante, detecta el sistema principal de ficheros de Linux, un *ext3* con *journal*, que activa y añade el espacio de *swap* disponible en una partición.

```
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
input: PC Speaker as /class/input/input1
USB Universal Host Controller Interface driver v3.0
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
uhci_hcd 0000:00:1d.1: UHCI Host Controller
uhci_hcd 0000:00:1d.1: new USB bus registered, assigned bus number 2
uhci_hcd 0000:00:1d.1: irq 11, io base 0x00001820
usb usb2: configuration #1 chosen from 1 choice
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 2 ports detected
hub 4-0:1.0: USB hub found
hub 4-0:1.0: 6 ports detected
```

Más detección de dispositivos, USB en este caso (y los módulos que corresponden). Ha detectado dos dispositivos *hub* (con un total de 8 USB *ports*).

```
parport: PnPBIOS parport detected.
parport0: PC-style at 0x378 (0x778), irq 7, dma 1 [PCSPP,TRISTATE,COMPAT,EPP,ECP,DMA]
input: ImPS/2 Logitech Wheel Mouse as /class/input/input2
ieee1394: Initialized config rom entry `ip1394'
eepro100.c:v1.09j-t 9/29/99 Donald Becker
Synaptics Touchpad, model: 1, fw: 5.9, id: 0x2e6eb1, caps: 0x944713/0xc0000
input: SynPS/2 Synaptics TouchPad as /class/input/input3

agpgart: Detected an Intel 845G Chipset
agpgart: Detected 8060K stolen Memory
agpgart: AGP aperture is 128M
eth0: OEM i82557/i82558 10/100 Ethernet, 00:00:F0:84:D3:A9, IRQ 11.
Board assembly 000000-000, Physical connectors present: RJ45
e100: Intel(R) PRO/100 Network Driver, 3.5.17-k2-NAPI
usbcore: registered new interface driver usbkbd
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.

lp0: using parport0 (interrupt-driven).

ppdev: user-space parallel port driver
```

Y la detección final del resto de dispositivos: puerto paralelo, modelo de ratón, puerto *firewire* (ieee1394) tarjeta de red (Intel), un panel táctil, la tarjeta de vídeo AGP (i845). Más datos de la tarjeta de red una intel pro 100, registro de usb como *mass storage* (indica un dispositivo de almacenamiento por usb, como un disco externo) y detección del puerto paralelo.

Toda esta información que hemos visto mediante el comando *dmesg,* también la podemos encontrar volcada en el log principal del sistema */var/log/*messages. En este log, entre otros, encontraremos los mensajes del *kernel* y de los *daemons*, y errores de red o dispositivos, los cuales comunican sus mensajes a un *daemon* especial llamado *syslogd,* que es el encargado de escribir los mensajes en este fichero. Si hemos arrancado la máquina recientemente, observaremos que las últimas líneas contienen exactamente la misma información que el comando *dmesg,* por ejemplo, si nos quedamos con la parte final del fichero (suele ser muy grande):

```
tail 200 /var/log/messages
```

Observamos las líneas de antes, y también algunas informaciones más, como por ejemplo:

```
shutdown[13325]: shutting down for system reboot
kernel: usb 4-1: USB disconnect, address 3
kernel: nfsd: last server has exited
kernel: nfsd: unexporting all filesystems
kernel: Kernel logging (proc) stopped.
kernel: Kernel log daemon terminating.

exiting on signal 15
syslogd 1.4.1#20: restart.

kernel: klogd 1.4.1#20, log source = /proc/kmsg started.
Linux version 2.6.20-1-686 (Debian 2.6.20-2) (waldi@debian.org) (gcc version
4.1.2 20061115 (prerelease) (Debian 4.1.1-21)) #1 SMP Sun Apr 15 21:03:57

kernel: BIOS-provided physical RAM map:
```

La primera parte corresponde a la parada anterior del sistema; nos informa de que el *kernel* ha dejado de colocar información en */proc*, se está parando el sistema... Al principio del arranque nuevo, se activa el *daemon Syslogd* que genera el log y comienza la carga del sistema, que nos dice que el *kernel* comenzará a escribir información en su sistema, */proc*. Vemos las primeras líneas de *dmesg* de mención de la versión que se está cargando de *kernel*, y luego encontraremos lo que hemos visto con *dmesg*.

En este punto, otro comando útil para saber cómo se ha producido la carga es *lsmod*, que nos permitirá saber qué módulos dinámicos se han cargado con el *kernel* (versión resumida):

```
# lsmod
Module            Size      Used by
nfs               219468    0
nfsd              202192    17
exportfs          5632      1 nfsd
lockd             58216     3 nfs,nfsd
nfs_acl           3616      2 nfs,nfsd
sunrpc            148380    13 nfs,nfsd,lockd,nfs_acl
ppdev             8740      0
lp                11044     0
button            7856      0
ac                5220      0
battery           9924      0
md_mod            71860     1
dm_snapshot       16580     0
dm_mirror         20340     0
dm_mod            52812     2 dm_snapshot,dm_mirror
i810fb            30268     0
vgastate          8512      1 i810fb
eeprom            7184      0
thermal           13928     0
processor         30536     1 thermal
fan               4772      0
udf               75876     0
ntfs              205364    0
usb_storage       75552     0
hid               22784     0
usbkbd            6752      0
eth1394           18468     0
e100              32648     0
eepro100          30096     0
ohci1394          32656     0
ieee1394          89208     2 eth1394,ohci1394
snd_intel8x0      31420     1
snd_ac97_codec    89412     1 snd_intel8x0
ac97_bus          2432      1 snd_ac97_codec
parport_pc        32772     1
snd               48196     6 snd_intel8x0,snd_ac97_codec,snd_pcm,snd_timer
ehci_hcd          29132     0
ide_cd            36672     0
cdrom             32960     1 ide_cd
soundcore         7616      1 snd
psmouse           35208     0
```

```
uhci_hcd          22160     0
parport           33672     3 ppdev,lp,parport_pc
intelfb           34596     0
serio_raw         6724      0
pcspkr            3264      0
pci_hotplug       29312     1 shpchp
usbcore           122312    6 dvb_usb,usb_storage,usbkbd,ehci_hcd,uhci_hcd
intel_agp         22748     1
agpgart           30504     5 i810fb,drm,intelfb,intel_agp
ext3              121032    1
jbd               55368     1 ext3
ide_disk          15744     3
ata_generic       7876      0
ata_piix          15044     0
libata            100052    2 ata_generic,ata_piix
scsi_mod          133100    2 usb_storage,libata
generic           4932      0 [permanent]
piix              9540      0 [permanent]
ide_core          114728    5 usb_storage,ide_cd,ide_disk,generic,piix
```

Vemos que disponemos de los controladores para el hardware que hemos detectado, y de otros relacionados o necesarios por dependencias.

Ya tenemos, pues, una idea de cómo se han cargado el *kernel* y sus módulos. En este proceso puede que ya hubiésemos observado algún error; si hay hardware mal configurado o módulos del *kernel* mal compilados (no eran para la versión del *kernel* adecuada), inexistentes, etc.

El paso siguiente será la observación de los procesos en el sistema, con el comando *ps* (Process Status), por ejemplo (sólo se han listado los procesos de sistema, no los de los usuarios):

```
#ps -ef

UID PID PPID C STIME TTY TIME CMD
```

Información de los procesos, *UID*, usuario que ha lanzado el proceso (o con qué identificador se ha lanzado), *PID*, código del proceso asignado por el sistema, son consecutivos a medida que se lanzan los procesos. El primero siempre es el 0, que corresponde al proceso de *init*. *PPID* es el id del proceso padre del actual. *STIME*, tiempo en que fue arrancado el proceso, *TTY*, terminal asignado al proceso (si tiene alguno), *CMD*, línea de comando con que fue lanzado.

```
root     1    0 0 14:52 ?     00:00:00 init [2]
root     3    1 0 14:52 ?     00:00:00 [ksoftirqd/0]
root    143   6 0 14:52 ?     00:00:00 [bdflush]
root    145   6 0 14:52 ?     00:00:00 [kswapd0]
```

```
root    357   6 0 14:52 ?     00:00:01 [kjournald]
root    477   1 0 14:52 ?     00:00:00 udevd --daemon
root    719   6 0 14:52 ?     00:00:00 [khubd]
```

Varios *daemons* de sistema, como *kswapd daemon*, que controla el intercambio de páginas con memoria virtual. Gestión de *buffers* del sistema (*bdflush*). Gestión de *journal* de *filesystem* (*kjournald*), gestión de USB (*khubd*). O el demonio de *udev* que controla la conexión en caliente de dispositivos. En general (no siempre) los *daemons* suelen identificarse por una *d* final, y si llevan un *k* inicial, normalmente son hilos (*threads*) internos del *kernel*.

```
root    1567  1 0 14:52 ?     00:00:00 dhclient -e -pf ...
root    1653  1 0 14:52 ?     00:00:00 /sbin/portmap
root    1829  1 0 14:52 ?     00:00:00 /sbin/syslogd
root    1839  1 0 14:52 ?     00:00:00 /sbin/klogd -x
root    1983  1 0 14:52 ?     00:00:09 /usr/sbin/cupsd
root    2178  1 0 14:53 ?     00:00:00 /usr/sbin/inetd
```

Tenemos: *dhclient*, lo que indica que esta máquina es cliente de un servidor DHCP, para obtener su IP; *Syslogd*, *daemon* que envía mensajes al log; el *daemon* de *cups*, como vimos, está relacionado con el sistema de impresión; e *inetd*, que, como veremos en la parte de redes, es una especie de "superservidor" o intermediario de otros *daemons* relacionados con servicios de red.

```
root      2154  1 0 14:53 ?     00:00:00 /usr/sbin/rpc.mountd
root      2241  1 0 14:53 ?     00:00:00 /usr/sbin/sshd
root      2257  1 0 14:53 ?     00:00:00 /usr/bin/xfs -daemon
root      2573  1 0 14:53 ?     00:00:00 /usr/sbin/atd
root      2580  1 0 14:53 ?     00:00:00 /usr/sbin/cron
root      2675  1 0 14:53 ?     00:00:00 /usr/sbin/apache
www-data  2684 2675 0 14:53 ?   00:00:00 /usr/sbin/apache
www-data  2685 2675 0 14:53 ?   00:00:00 /usr/sbin/apache
```

También está *sshd*, servidor de acceso remoto seguro (una versión mejorada que permite servicios compatibles con telnet y ftp); *xfs* es el servidor de fuentes (tipos de letra) de X Window. Los comandos *atd* y *cron* sirven para manejar tareas programadas en un momento determinado. Apache es el servidor web, que puede tener varios hilos activos (*threads*) para atender diferentes peticiones.

```
root    2499 2493 0 14:53 ?         00:00:00 /usr/sbin/gdm
root    2502 2499 4 14:53 tty7      00:09:18 /usr/bin/X :0 -dpi 96 ...
root    2848   1  0 14:53 tty2      00:00:00 /sbin/getty 38400 tty2
root    2849   1  0 14:53 tty3      00:00:00 /sbin/getty 38400 tty3
root    3941 2847 0 14:57 tty1      00:00:00 -bash
root    16453 12970 0 18:10 pts/2   00:00:00 ps -ef
```

Así, *gdm* es el login gráfico del sistema de escritorio Gnome (la entrada que nos pide el *login* y contraseña); los procesos *getty* son los que gestionan los terminales virtuales de texto (los que podemos ver con las teclas Alt+Fn, o Ctrl+Alt+Fn si estamos en modo gráfico); *X* es el proceso de servidor gráfico de X Window System, imprescindible para que se ejecute cualquier entorno de escritorio por encima de él. Un *shell* abierto (*bash*), y finalmente el proceso que hemos generado al pedir este *ps* desde la línea de comandos.

El comando *ps* tiene muchas opciones de línea de comandos para ajustar la información que queremos de cada proceso, ya sea tiempo que hace que se ejecuta, porcentaje de CPU usado, memoria utilizada, etc. (ved man de *ps*). Otro comando muy interesante es *top,* que hace lo mismo que *ps* pero de forma dinámica; se actualiza cada cierto intervalo; podemos clasificar los procesos por uso de CPU, de memoria, y también nos da información del estado de la memoria global.

Nota

Ver man de los comandos para interpretar las salidas.

Otros comandos útiles para uso de recursos son *free* y *vmstat*, que nos dan información sobre la memoria utilizada y el sistema de memoria virtual:

```
# free
             total       used       free     shared    buffers     cached
Mem:        767736     745232      22504          0      89564     457612
-/+ buffers/cache:     198056     569680
Swap:       618492       1732     616760

# vmstat
procs -----------memory----------   ---swap-- -----io-- --system-- ----cpu----
r  b   swpd free     buff     cache   si so bi bo    in   cs us sy   id wa
1  0   1732 22444    89584    457640 0  0  68 137   291  418 7 1    85 7
```

En el comando *free* también se puede observar el tamaño del *swap* presente, aproximadamente de unas 600 MB, que de momento no se está usando intensamente por tener suficiente espacio de memoria física. Todavía quedan unas 22 MB libres (lo que indica una alta utilización de la memoria física y un uso de *swap* próximamente). El espacio de memoria y el *swap* (a partir de los *kernels* 2.4) son aditivos para componer el total de memoria del sistema, haciendo en este caso un total de 1,4 GB de memoria disponible.

Actividades

1. Haced una lectura rápida del estándar FHS, que nos servirá para tener una buena guía a la hora de buscar archivos por nuestra distribución.

Nota

Ver FHS en:
http://www.pathname.com/fhs

2. Para los paquetes RPM, ¿cómo haríais algunas de las siguientes tareas?

a) Conocer qué paquete instaló un determinado comando.

b) Obtener la descripción del paquete que instaló un comando.

c) Borrar un paquete cuyo nombre completo no conocemos.

d) Mostrar todos los archivos que estaban en el mismo paquete que un determinado archivo.

3. Efectuad las mismas tareas que en la actividad anterior, pero para paquetes Debian, usando herramientas *APT*.

4. Actualizad una distribución Debian (o Fedora).

5. Instalad en nuestra distribución alguna herramienta genérica de administración, como *Webadmin*. ¿Qué nos ofrece? ¿Entendéis las tareas ejecutadas y los efectos que provocan?

6. El espacio de *swap* permite complementar la memoria física para disponer de mayor memoria virtual. Dependiendo de los tamaños de memoria física y *swap*, ¿puede llegar a agotarse la memoria? ¿Podemos solucionarlo de otro modo, que no sea añadiendo más memoria física?

7. Supongamos que tenemos un sistema con dos particiones Linux, una */* y la otra de *swap*. ¿Cómo solucionaríais el caso de que las cuentas de los usuarios agotasen el espacio de disco? Y en el caso de tener una partición */home* aislada que se estuviera agotando, ¿cómo lo solucionaríais?

8. Instalad el sistema de impresión CUPS, definid nuestra impresora para que funcione con CUPS y probad la administración vía interfaz web. Tal como está el sistema, ¿sería recomendable modificar de algún modo la configuración que trae por defecto CUPS? ¿Por qué?

9. Analizad el sistema Upstart presente en una distribución Fedora. ¿Qué eventos y *jobs* trae predefinidos? ¿Hay compatibilidad con el *init* de SystemV?

10. Examinad la configuración por defecto que traiga el sistema GNU/Linux de trabajos no interactivos por *cron*. ¿Qué trabajos y cuándo se están realizando? ¿Alguna idea para nuevos trabajos que haya que añadir?

11. Reproducid el análisis del taller (más los otros apartados de la unidad) sobre la máquina de que dispongáis; ¿se observan en el sistema algunos errores o situaciones anómalas? En tal caso, ¿cómo las corregís?

Bibliografía

[Bai03] Bailey, E. C. (2003). *RedHat Maximum RPM*.

<http://www.redhat.com/docs/books/max-rpm/index.html>

Ofrece una amplia visión de los sistemas de paquetes de software de las distribuciones Debian y Fedora/Red Hat.

[Debb] Comunidad Debian. "Distribución Debian". http://www.debian.org

[Coo] Cooper, M. (2006). "Advanced bashScripting Guide". *The Linux Documentation Project* (guías).

Ofrece una amplia introducción (y conceptos avanzados) a la programación de *shell scripts* en *bash*, así como numerosos ejemplos.

[Deb] Debian. "Sitio Seguridad de Debian".http://www.debian.org/security/

Ofrece una amplia visión de los sistemas de paquetes de software de las distribuciones Debian y Fedora/Red Hat.

[lin03b] *FHS Standard*, 2003. http://www.pathname.com/fhs

[Fri02] Frisch, A. (2002). *Essential System Administration*. O'Reilly.

Administración de GNU/Linux y UNIX. Comenta de forma amplia aspectos de administración local y gestión de sistemas de impresión.

[Hin00] Hinner, M. "Filesystems HOWTO". *The Linux Documentation Project.*

Información sobre los diferentes sistemas de ficheros disponibles y los esquemas de creación de particiones para la instalación del sistema.

[Koe] Koehntopp, K. "Linux Partition HOWTO". *The Linux Documentation Project.*

Información sobre los diferentes sistemas de ficheros disponibles y los esquemas de creación de particiones para la instalación del sistema.

[Linc] *Linux Standards Base project*.http://www.linux-foundation.org/en/LSB

[Bas] Mike, G. "BASH Programming - Introduction HOWTO". *The Linux Documentation Project.*

Ofrece una amplia introducción (y conceptos avanzados) a la programación de *shell scripts* en *bash*, así como numerosos ejemplos.

[Mor03] Morill, D. (2003). *Configuración de sistemas Linux*. Anaya Multimedia.

[Nem06] Nemeth, E.; Snyder, G.; Hein, T. R. (2006). "Linux Administration Handbook" (2.ª ed.). Prentice Hall.

Trata de forma amplia la mayoría de aspectos de administración y es una buena guia genérica para cualquier distribución.

[Qui01] Quigley, E. (2001). *Linux shells by Example*. Prentice Hall.

Comenta los diferentes *shells* de programación en GNU/Linux, así como sus semejanzas y diferencias.

[SM02] Schwartz, M. y otros (2002). *Multitool Linux - Practical Uses for Open Source Software*. Addison Wesley.

[Smi02] Smith, R. (2002). *Advanced Linux Networking*. Addison Wesley.

Administración de GNU/Linux y UNIX. Comenta de forma amplia aspectos de administración local y gestión de sistemas de impresión.

[Sob10] Sobell, M. G. (2010). *A Practical Guide to Fedora and Red Hat Enterprise Linux*. Prentice Hall.

Es una buena guía de administración local para distribuciones Red Hat y Fedora.

[Stu] Stutz, M. "The Linux Cookbook: Tips and Techniques for Everyday Use". *The Linux Documentation Project* (guías).

Es una amplia introducción a las herramientas disponibles en GNU/Linux.

[Gt] Taylor, G.; Allaert, D. "The Linux Printing HOWTO". *The Linux Documentation Project.*

Ofrece información actualizada de los sistemas de impresión y su configuración, así como detalles de algunas impresoras. Para detalles concretos de modelos de impresora y controladores, podéis dirigiros a:

<http://www.linuxprinting.org/>

[Fed] The Fedora Project. http://fedoraproject.org

[Wm02] Welsh, M. y otros (2002). *Running Linux 4th edition*. O'Reilly.

Administración de GNU/Linux y UNIX. Comenta de forma amplia aspectos de administración local y gestión de sistemas de impresión.

Administración de red

Remo Suppi Boldrito

PID_00167544

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

El sistema operativo UNIX (GNU/Linux) se toma como ejemplo de una arquitectura de comunicaciones estándar. Desde el mítico UUCP (servicio de copia entre sistemas operativos UNIX) hasta las redes actuales, UNIX siempre ha mostrado su versatilidad en aspectos relacionados con la comunicación y el intercambio de información. Con la introducción de redes de ordenadores (área local LAN, área amplia WAN o las más actuales, área metropolitana MAN) ofreciendo enlaces multipunto a diferentes velocidades (56kbits/seg hasta 1Gbit/seg), han ido surgiendo nuevos servicios basados en protocolos más rápidos, portables entre diferentes ordenadores y mejor adaptados, como el TCP/IP (*transport control program / internet protocol*) [Com01, Mal96, Cis00, Gar98, KD00].

1. Introducción a TCP/IP (TCP/IP suite)

El protocolo TCP/IP sintetiza un ejemplo de estandarización y una voluntad de comunicación a nivel global.

El protocolo TCP/IP es en realidad un conjunto de protocolos básicos que se han ido agregando al principal para satisfacer las diferentes necesidades en la comunicación ordenador-ordenador como son TCP, UDP, IP, ICMP, ARP. [Mal96]

La utilización más frecuente de TCP/IP para el usuario en la actualidad son la conexión remota a otros ordenadores (telnet, SSH[1]), la utilización de ficheros remotos (NFS[2]) o su transferencia (FTP[3], HTTP[4]).

(1)Del inglés *secure shell*.

(2)Del inglés *network file system*.

(3)Del inglés *file transfer protocol*.

(4)Del inglés *hipertext markup protocol*.

1.1. Servicios sobre TCP/IP

Los servicios TCP/IP tradicionales más importantes son [Gar98]:

a) Transferencia de archivos: el FTP permite a un usuario de un ordenador obtener/enviar archivos de un ordenador hacia otro ordenador. Para ello, el usuario deberá tener una cuenta, en el ordenador remoto, identificarse a través de su nombre (*login*) y una palabra clave (*password*) o en ordenadores donde existe un repositorio de información (software, documentación...), el usuario se conectará como anónimo (*anonymous*) para transferir (leer) estos archivos a su ordenador. Esto no es lo mismo que los más recientes sistemas de archivos de red, NFS (o protocolos *netbios* sobre tcp/ip, "invento" totalmente inseguro sobre Windows y que es mejor reemplazar por una versión más antigua pero más segura del mismo concepto llamado *netbeui*), que permiten virtualizar el sistema de archivos de una máquina para que pueda ser accedido en forma interactiva sobre otro ordenador.

b) Conexión (*login*) remota: el protocolo de terminal de red (telnet) permite a un usuario conectarse a un ordenador remotamente. El ordenador local se utiliza como terminal del ordenador remoto y todo es ejecutado sobre éste permaneciendo el ordenador local invisible desde el punto de vista de la sesión. Este servicio en la actualidad se ha reemplazado por el SHH por razones de seguridad. En una conexión remota mediante telnet, los mensajes circulan tal cual (texto plano), o sea, si alguien "observa" los mensajes en la red, equivaldrá a mirar la pantalla del usuario. SSH codifica la información (que significa un coste añadido a la comunicación), que hace que los paquetes en la red sean ilegibles a un nodo extraño.

TCP/IP

Utilización típica de TCP/IP remote *login*:

```
telnet localhost
Debian GNU/Linux 4.0
login:
```

c) eMail: este servicio permite enviar mensajes a los usuarios de otros ordenadores. Este modo de comunicación se ha transformado en un elemento vital en la vida de los usuarios y permiten que los *e-mails* (correos electrónicos) sean enviados a un servidor central para que después puedan ser recuperados por medio de programas específicos (clientes) o leídos a través de una conexión web.

El avance de la tecnología y el bajo coste de los ordenadores han permitido que determinados servicios se hayan especializado en ello y se ofrecen configurados sobre determinados ordenadores trabajando en un modelo cliente-servidor. Un servidor es un sistema que ofrece un servicio específico para el resto de la red. Un cliente es otro ordenador que utiliza este servicio. Todos estos servicios generalmente son ofrecidos dentro de TCP/IP:

- **Sistemas de archivos en red** (NFS): permite a un sistema acceder a los archivos sobre un sistema remoto en una forma más integrada que FTP. Los dispositivos de almacenamiento (o parte de ellos) son exportados hacia el sistema que desea acceder y éste los puede "ver" como si fueran dispositivos locales. Este protocolo permite a quien exporta poner las reglas y las formas de acceso, lo que (bien configurado) hace independiente el lugar donde se encuentra la información físicamente del sitio donde se "ve" la información.

- **Impresión remota**: permite acceder a impresoras conectadas a otros ordenadores.

- **Ejecución remota**: permite que un usuario ejecute un programa sobre otro ordenador. Existen diferentes maneras de realizar esta ejecución: o bien a través de un comando (*rsh*, ssh, *rexec*) o a través de sistemas con RPC[5] que permiten a un programa en un ordenador local ejecutar una función de un programa sobre otro ordenador. Los mecanismos RPC han sido objeto de estudio y existen diversas implementaciones, pero las más comunes son Xerox's Courier y Sun's RPC (ésta última adoptada por la mayoría de los UNIX).

- **Servidores de nombre** (*name servers*): en grandes instalaciones existen un conjunto de datos que necesitan ser centralizados para mejorar su utilización, por ejemplo, nombre de usuarios, palabras clave, direcciones de red, etc. Todo ello facilita que un usuario disponga de una cuenta para todas las máquinas de una organización. Por ejemplo, Sun's Yellow Pages (NIS en las versiones actuales de *Sun*) está diseñado para manejar todo este tipo de datos y se encuentra disponible para la mayoría de UNIX. El DNS[6] es otro servicio de nombres pero que guarda una relación entre el nombre de la máquina y la identificación lógica de esta máquina (dirección IP).

(5) Del inglés *remote procedure call*.

(6) Del inglés *domain name system*.

- **Servidores de terminal** (*terminal servers*): conecta terminales a un servidor que ejecuta telnet para conectarse al ordenador central. Este tipo de instalaciones permite básicamente reducir costes y mejorar las conexiones al ordenador central (en determinados casos).

- **Servidores de terminales gráficas** (*network-oriented window systems*): permiten que un ordenador pueda visualizar información gráfica sobre un display que está conectado a otro ordenador. El más común de estos sistemas es X Window.

1.2. ¿Qué es TCP/IP?

TCP/IP son en realidad dos protocolos de comunicación entre ordenadores independientes uno del otro.

Por un lado, TCP[7], define las reglas de comunicación para que un ordenador (*host*) pueda 'hablar' con otro (si se toma como referencia el modelo de comunicaciones OSI/ISO se describe la capa 4, ver tabla siguiente). TCP es orientado a conexión, es decir, equivalente a un teléfono, y la comunicación se trata como un flujo de datos (*stream*).

(7)Del inglés *transmission control protocol.*

Por otro lado, IP[8], define el protocolo que permite identificar las redes y establecer los caminos entre los diferentes ordenadores. Es decir, encamina los datos entre dos ordenadores a través de las redes. Corresponde a la capa 3 del modelo OSI/ISO y es un protocolo sin conexión (ver tabla siguiente). [Com01, Rid00, Dra99]

(8)Del inglés *internet protocol.*

Una alternativa al TCP la conforma el protocolo UDP[9], el cual trata los datos como un mensaje (*datagrama*) y envía paquetes. Es un protocolo sin conexión[10] y tiene la ventaja de que ejerce una menor sobrecarga en la red que las conexiones de TCP, pero la comunicación no es fiable (los paquetes pueden no llegar o llegar duplicados).

(9)Del inglés *user datagram protocol.*

(10)El ordenador destino no debe necesariamente estar escuchando cuando un ordenador establece comunicación con él.

Existe otro protocolo alternativo llamado ICMP[11]. ICMP se utiliza para mensajes de error o control. Por ejemplo, si uno intenta conectarse a un equipo (*host*), el ordenador local puede recibir un mensaje ICMP indicando "*host unreachable*". ICMP también puede ser utilizado para extraer información sobre una red. ICMP es similar a UDP, ya que maneja mensajes (datagramas), pero es más simple que UDP, ya que no posee identificación de puertos[12] en el encabezamiento del mensaje.

(11)Del inglés *internet control message protocol.*

(12)Son buzones donde se depositan los paquetes de datos y desde donde las aplicaciones servidoras leen dichos paquetes.

En el modelo de comunicaciones de la OSI[13]/ISO[14], es un modelo teórico adoptado por muchas redes. Existen siete capas de comunicación, de las que cada una tiene una interfaz para comunicarse con la anterior y la posterior:

(13)Del inglés *open systems interconnection reference model.*

Nivel	Nombre	Utilización
7	Aplicación	SMTP[15], el servicio propiamente dicho
6	Presentación	Telnet, FTP implementa el protocolo del servicio
5	Sesión	Generalmente no se utiliza
4	Transporte	TCP, UDP transformación de acuerdo a protocolo de comunicación
3	Red	IP permite encaminar el paquete (*routing*)
2	Link	Controladores (*drivers*) transformación de acuerdo al protocolo físico
1	Físico	Ethernet, ADSL, ... envía del paquete físicamente

(14)Del inglés *international standards organization.*

(15)Del inglés *simple mail transfer protocol.*

En resumen, TCP/IP es una familia de protocolos (que incluyen IP, TCP, UDP) que proveen un conjunto de funciones a bajo nivel utilizadas por la mayoría de las aplicaciones. [KD00, Dra99].

Algunos de los protocolos que utilizan los servicios mencionados han sido diseñados por Berkeley, Sun u otras organizaciones. Ellos no forman oficialmente parte de ***Internet Protocol Suite*** (IPS). Sin embargo, son implementados utilizando TCP/IP y por lo tanto considerados como parte formal de IPS. Una descripción de los protocolos disponibles en Internet puede consultarse la RFC 1011 (ver referencias sobre RFC [IET]), que lista todos los protocolos disponibles. Existe actualmente una nueva versión del protocolo **IPv6**, también llamado IPng[16] que reemplaza al **IPv4**. Este protocolo mejora notablemente el anterior en temas tales como mayor número de nodos, control de tráfico, seguridad o mejoras en aspectos de routing.

(16)Del inglés *IP next generation.*

1.3. Dispositivos físicos (hardware) de red

Desde el punto de vista físico (capa 1 del modelo OSI), el hardware más utilizado para LAN es conocido como Ethernet (o FastEthernet o GigaEthernet). Sus ventajas son su bajo coste, velocidades aceptables (10, 100, o 1.000 megabits por segundo) y facilidad en su instalación.

Existen tres modos de conexión en función del tipo de cable de interconexión: grueso (*thick*), fino (*thin*) y par trenzado (*twisted par*).

Las dos primeras están obsoletas (utilizan cable coaxial), mientras que la última se realiza a través de cables (pares) trenzados y conectores similares a los telefónicos (se conocen como RJ45). La conexión par trenzado es conocida como 10baseT o 100baseT (según la velocidad) y utiliza repetidores llamados *hubs* como puntos de interconexión. La tecnología Ethernet utiliza elementos intermedios de comunicación (*hubs, switchs, routers*) para configurar múltiples

(17)Del inglés *fiber distributed data interface.*

segmentos de red y dividir el tráfico para mejorar las prestaciones de transferencia de información. Normalmente, en las grandes instituciones estas LAN Ethernet están interconectadas a través de fibra óptica utilizando tecnología FDDI[17] que es mucho más cara y compleja de instalar, pero se pueden obtener velocidades de transmisión equivalentes a Ethernet y no tienen la limitación de la distancia de ésta (FDDI admite distancias de hasta 200 km). Su coste se justifica para enlaces entre edificios o entre segmentos de red muy congestionados. [Rid00, KD00]

Existe además otro tipo de hardware menos común, pero no menos interesante como es ATM[18]. Este hardware permite montar LAN con una calidad de servicio elevada y es una buena opción cuando deben montarse redes de alta velocidad y baja latencia, como por ejemplo aquellas que involucren distribución de vídeo en tiempo real.

(18)Del inglés *asynchronous transfer mode*.

Existe otro hardware soportado por GNU/Linux para la interconexión de ordenadores, entre los cuales podemos mencionar: *Frame Relay* o X.25, utilizada en ordenadores que acceden o interconectan WAN y para servidores con grandes necesidades de transferencias de datos; *Packet Radio*, interconexión vía radio utilizando protocolos como AX.25, NetRom o Rose, o dispositivos dialing up, que utilizan líneas series, lentas pero muy baratas, a través de un módem analógico o digital (RDSI, DSL, ADSL, etc.). Estas últimas son las que comúnmente se utilizan en pymes o uso doméstico y requieren otro protocolo para la transmisión de paquetes, tal como SLIP o PPP. Para virtualizar la diversidad de hardware sobre una red, TCP/IP define una interfaz abstracta mediante la cual se concentrarán todos los paquetes que serán enviados por un dispositivo físico (lo cual también significa una red o un segmento de esta red). Por ello, por cada dispositivo de comunicación en la máquina tenderemos una interfaz correspondiente en el kernel del sistema operativo.

Ethernet

Ethernet en GNU/Linux se llaman con ethx (donde en todas, x indica un número de orden comenzando por 0), la interfaz a líneas series (módem) se llaman por pppx (para PPP) o slx (para SLIP), para FDDI son fddix. Estos nombres son utilizados por los comandos para configurar sus parámetros y asignarles el número de identificación que posteriormente permitirá comunicarse con otros dispositivos en la red.

En GNU/Linux puede significar tener que incluir los módulos adecuados para el dispositivo (NIC[19]) adecuado (en el kernel o como módulos), esto significa compilar el kernel después de haber escogido con, por ejemplo, **make menuconfig** el NIC adecuado, indicándole como interno o como módulo (en este último caso se deberá compilar el módulo adecuado también).

(19)Del inglés *network interface card*.

ifconfig -a

Para ver las interfaces de red disponibles hay que aplicar el comando *ifconfig -a*. Este comando muestra todas las interfaces/parámetros por defecto de cada una.

Los dispositivos de red se pueden mirar en el directorio */dev*, que es donde existe un archivo (especial, ya sea de bloque o de caracteres según su transferencia), que representa a cada dispositivo hardware [KD00, Dra99].

2. Conceptos en TCP/IP

Como se ha observado, la comunicación significa una serie de conceptos que ampliaremos a continuación [Mal96, Com01]:

- **Internet/intranet**: el término *intranet* se refiere a la aplicación de tecnologías de Internet (red de redes) dentro de una organización, básicamente para distribuir y tener disponible información dentro de la compañía. Por ejemplo, los servicios ofrecidos por GNU/Linux como servicios Internet e intranet incluyen correo electrónico, WWW, *news*, etc.

- **Nodo**: se denomina nodo (*host*) a una máquina que se conecta a la red (en un sentido amplio, un nodo puede ser un ordenador, una impresora, una torre (*rack*) de CD, etc.), es decir, un elemento activo y diferenciable en la red que reclama o presta algún servicio y/o comparte información.

- **Dirección de red Ethernet** (*Ethernet address* o *MAC address*): un número de 48 bits (por ejemplo 00:88:40:73:AB:FF -en octal- 0000 0000 1000 1000 0100 0000 0111 0011 1010 1011 1111 1111 -en binario-) que se encuentra en el dispositivo físico (hardware) del controlador (NIC) de red Ethernet y es grabado por el fabricante del mismo (este número debe ser único en el mundo, por lo que cada fabricante de NIC tiene un rango preasignado).

- **Host name**: cada nodo debe tener además un único nombre en la red. Ellos pueden ser sólo nombres o bien utilizar un esquema de nombres jerárquico basado en dominios (*hierarchical domain naming scheme*). Los nombres de los nodos deben ser únicos, lo cual resulta fácil en pequeñas redes, más dificultoso en redes extensas, e imposible en Internet si no se realiza algún control. Los nombres deben ser de un máximo de 32 caracteres entre a-zA-Z0-9.-, y que no contengan espacios o # comenzando por un carácter alfabético.

Nota

Nombre de la máquina: **more /etc/hostname**.

- **Dirección de Internet** (*IP address*): está compuesto por cuatro números en el rango 0-255 separados por puntos (por ejemplo, 192.168.0.1) y se utiliza universalmente para identificar los ordenadores sobre una red o Internet. La traslación de nombres en direcciones IP la realiza un servidor DNS (*domain name system*) que transforma los nombres de nodo (legibles por humanos) en direcciones IP (este servicio lo realiza una aplicación denominada named).

Nota

Dirección IP de la máquina: **more /etc/hosts**.

- **Puerto** (*port*): identificador numérico del buzón en un nodo que permite que un mensaje (TCP, UDP) pueda ser leído por una aplicación concreta dentro de este nodo (por ejemplo, dos máquinas que se comuniquen por telnet lo harán por el puerto 23, pero estas mismas máquinas pueden tener una comunicación ftp por el puerto 21). Se pueden tener diferentes aplicaciones comunicándose entre dos nodos a través de diferentes puertos simultáneamente.

Nota

Puertos preasignados en UNIX: **more /etc/services**. Este comando muestra los puertos predefinidos por orden y según soporten TCP o UDP.

- **Nodo router** (*gateway*): es un nodo que realiza encaminamientos (transferencia de datos *routing*). Un *router*, según sus características, podrá transferir información entre dos redes de protocolos similares o diferentes y puede ser además selectivo.

Nota

Visualización de la configuración del routing: **netstat -r**.

- ***Domain name system*** (DNS): permite asegurar un único nombre y facilitar la administración de las bases de datos que realizan la traslación entre nombre y dirección de Internet y se estructuran en forma de árbol. Para ello, se especifican dominios separados por puntos, de los que el más alto (de derecha a izquierda) describe una categoría, institución o país (COM, comercial, EDU, educación, GOV, gubernamental, MIL, militar (gobierno), ORG, sin fin de lucro, XX dos letras por país, o en casos especiales tres letras CAT lengua y cultura catalana...). El segundo nivel representa la organización, el tercero y restantes departamentos, secciones o divisiones dentro de una organización (por ejemplo, www.uoc.edu o nteum@pirulo.remix.es). Los dos primeros nombres (de derecha a izquierda, *uoc.edu* en el primer caso, *remix.es* (en el segundo) deben ser asignados (aprobados) por el SRI-NIC (órgano mundial gestor de Internet) y los restantes pueden ser configurados/asignados por la institución.

Nota

Dominio y quién es nuestro servidor de DNS: **more /etc/default domain**; **more /etc/resolv.conf**.

- **DHCP, bootp:** DHCP y bootp son protocolos que permiten a un nodo cliente obtener información de la red (tal como la dirección IP del nodo). Muchas organizaciones con gran cantidad de máquinas utilizan este mecanismo para facilitar la administración en grandes redes o donde existe una gran cantidad de usuarios móviles.

- **ARP, RARP:** en algunas redes (como por ejemplo IEEE 802 LAN, que es el estándar para Ethernet), las direcciones IP son descubiertas automáticamente a través de dos protocolos miembros de IPS: ARP[20] y RARP[21]. ARP utiliza mensajes (*broadcast messages*) para determinar la dirección Ethernet (especificación MAC de la capa 3 del modelo OSI) correspondiente a una dirección de red particular (IP). RARP utiliza mensajes de tipo broadcast (mensaje que llega a todos los nodos) para determinar la dirección de red asociada con una dirección hardware en particular. RARP es especialmente importante en máquinas sin disco, en las cuales la dirección de red generalmente no se conoce en el momento del inicio (*boot*).

[20]Del inglés *address resolution protocol.*

[21]Del inglés *reverse address resolution protocol.*

Nota

Tablas de arp:
arp a *NombreNodo*.

- **Biblioteca de sockets**: en UNIX toda la implementación de TCP/IP forma parte del kernel del sistema operativo (o bien dentro del mismo o como un

módulo que se carga en el momento del inicio como el caso de GNU/Linux con los controladores de dispositivos).

La forma de utilizarlas por un programador es a través de la API[22] que implementa ese operativo. Para TCP/IP, la API más común es la Berkeley Socket Library (Windows utiliza una librería equivalente que se llama Winsocks). Esta biblioteca permite crear un punto de comunicación (*socket*), asociar éste a una dirección de un nodo remoto/puerto (*bind*) y ofrecer el servicio de comunicación (a través de *connect*, *listen*, *accept, send, sendto, recv, recvfrom,* por ejemplo). La biblioteca provee, además de la forma más general de comunicación (familia AF INET), comunicaciones más optimizadas para casos en que los procesos se comunican en la misma máquina (familia AF UNIX). En GNU/Linux, la biblioteca de sockets es parte de la biblioteca estándar de C, Libc, (Libc6 en las versiones actuales), y soporta AF_INET, AF_UNIX, AF_IPX (para protocolos de redes Novell), AF_X25 (para el protocolo X.25), AF_ATMPVC-AF_ATMSVC (para el protocolo ATM) y AF_AX25, F_NETROM, AF_ROSE (para el *amateur radio protocol*).

(22)Del inglés *application programming interface.*

3. ¿Cómo se asigna una dirección Internet?

Esta dirección es asignada por el SRI-NIC y tiene dos campos. El izquierdo representa la identificación de la red y el derecho la identificación del nodo. Considerando lo mencionado anteriormente (4 números entre 0-255, o sea 32 bits o cuatro bytes), cada byte representa o bien la red o bien el nodo. La parte de red es asignada por el SRI-NIC y la parte del nodo es asignada por la institución o el proveedor).

Existen algunas restricciones: **0** (por ejemplo, 0.0.0.0) en el campo de red está reservado para el routing por defecto y **127** (por ejemplo, 127.0.0.1) es reservado para la autorreferencia (*local loopback* o *local host*), **0** en la parte de nodo se refiere a esta red (por ejemplo, 192.168.0.0) y **255** es reservado para paquetes de envío a todas las máquinas (broadcast) (por ejemplo, 198.162.255.255). En las diferentes asignaciones se puede tener diferentes tipos de redes o direcciones:

- **Clase A** (*red.host.host.host*): 1.0.0.1 a 126.254.254.254 (126 redes, 16 millones de nodos) definen las grandes redes. El patrón binario es: **0** + 7 bits red + 24 bits de nodos.

- **Clase B** (*red.red.host.host*): 128.1.0.1 a 191.255.254.254 (16K redes, 65K nodos) generalmente se utiliza el primer byte de nodo para identificar subredes dentro de una institución). El patrón binario es **10** + 14 bits de red + 16 bits de nodos.

- **Clase C** (*red.red.red.host*): 192.1.1.1 a 223.255.255.254 (2 millones de bits de redes, 254 de nodos). El patrón binario es **110** + 21 bits red + 8 bits de nodos.

- **Clase D y E** (*red.red.red.host*): 224.1.1.1 a 255.255.255.254 reservado para *multicast* (desde un nodo a un conjunto de nodos que forman parte de un grupo) y propósitos experimentales.

Algunos rangos de direcciones han sido reservados para que no correspondan a redes públicas, sino a redes privadas y los mensajes no serán encaminados a través de Internet, lo cual es comúnmente conocido como Intranet. Éstas son para la **clase A** 10.0.0.0 hasta 10.255.255.255, **clase B** 172.16.0.0 hasta 172.31.0.0 y **clase C** 192.168.0.0 hasta 192.168.255.0.

Redes privadas

Máquinas que se conectan entre ellas sin tener conexión con el exterior.

La dirección de broadcast es especial, ya que cada nodo en una red escucha todos los mensajes (además de su propia dirección). Esta dirección permite que datagramas, generalmente información de *routing* y mensajes de aviso, puedan ser enviados a una red y todos los nodos del mismo segmento de red los puedan leer. Por ejemplo, cuando ARP busca encontrar la dirección Ethernet correspondiente a una IP, éste utiliza un mensaje de *broadcast*, el cual es enviado a todas las máquinas de la red simultáneamente. Cada nodo en la red lee este mensaje y compara la IP que se busca con la propia y le retorna un mensaje al nodo que hizo la pregunta si hay coincidencia.

Dos conceptos complementarios a lo descrito anteriormente es el de **subredes** y **routing** entre ellas. Subredes significa subdividir la parte del nodo en pequeñas redes dentro de la misma red para, por ejemplo, mejorar el tráfico. Una subred toma la responsabilidad de enviar el tráfico a ciertos rangos de direcciones IP extendiendo el mismo concepto de redes A, B, C, pero sólo aplicando esta redirección en la parte nodo de la IP. El número de bits que son interpretados como identificador de la subred es dado por una máscara de red (*netmask*) que es un número de 32 bits (igual que la IP).

Para obtener el identificador de la subred, se deberá hacer una operación lógica **Y** (AND) entre la máscara y la IP, lo cual dará la IP de la subred.

Por ejemplo, sea una institución que tiene una red clase B con número 172.17.0.0, y su netmask es, por lo tanto, 255.255.0.0. Internamente, esta red está formada por pequeñas redes (una planta del edificio, por ejemplo). Así, el rango de direcciones es reasignado en 20 subnets (plantas para nosotros) 172.17.1.0 hasta 172.17.20.0. El punto que conecta todas estas plantas (*backbone*) tiene su propia dirección, por ejemplo 172.17.1.0.

Estas subredes comparten el mismo IP de red, mientras que el tercero es utilizado para identificar cada una de las subredes dentro de ella (por eso se utilizará una máscara de red 255.255.255.0).

El segundo concepto, routing, representa el modo en que los mensajes son enviados a través de las subredes.

Por ejemplo, sean tres departamentos con subredes Ethernet:

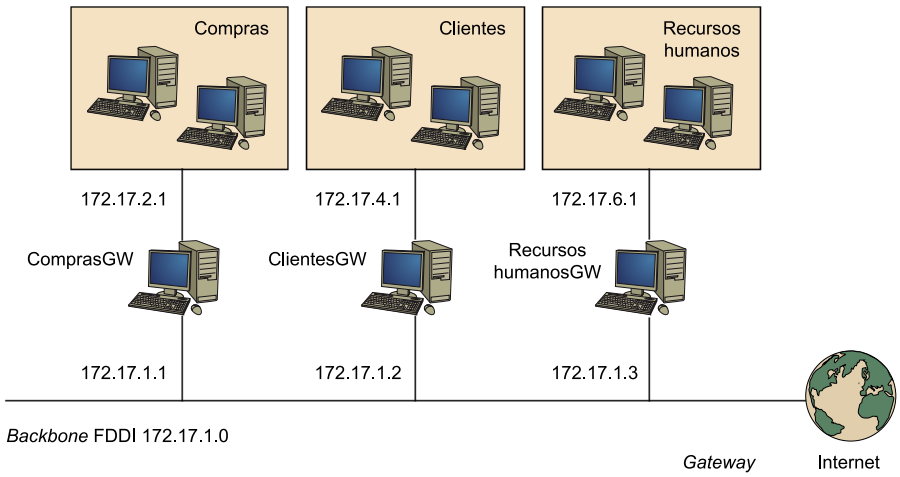
- Compras (subred 172.17.2.0).
- Clientes (subred 172.17.4.0).
- Recursos humanos, RR.HH., (subred 172.17.6.0).
- Backbone con FFDI (subred 172.17.1.0).

Para encaminar los mensajes entre los ordenadores de las tres redes, se necesitarán tres puertas de intercambio (*gateways*), que tendrán cada una dos interfaces de red para cambiar entre Ethernet y FFDI. Éstas serán:

- CromprasGW IPs:172.17.2.1 y 172.17.1.1,
- ClientesGW IPs:172.17.4.1 y 172.17.1.2
- RRHHGW IPs:172.17.6.1 y 172.17.1.3, es decir, una IP hacia el lado de la subnet y otra hacia el backbone.

Cuando se envían mensajes entre máquinas de compras, no es necesario salir al gateway, ya que el protocolo TCP/IP encontrará la máquina directamente. El problema está cuando la máquina Compras0 quiere enviar un mensaje a RRHH3. El mensaje debe circular por los dos gateways respectivos. Cuando Compras0 "ve" que RRHH3 está en otra red, envía el paquete a través del gateway ComprasGW, que a su vez se lo enviará a RRHHGW y que a su vez se lo enviará a RRHH3. La ventaja de las subredes es clara, ya que el tráfico entre todas las máquinas de compras, por ejemplo, no afectará a las máquinas de clientes o de recursos humanos (si bien significa un planteamiento más complejo y caro a la hora de diseñar, y construir la red).

Figura 1. Configuración de segmentos y gateways en una Intranet



IP utiliza una tabla para hacer el routing de los paquetes entre las diferentes redes y en la cual existe un routing por defecto asociado a la red 0.0.0.0. Todas las direcciones que coinciden con ésta, ya que ninguno de los 32 bits son necesarios, son enviadas por el *gateway* por defecto (*default gateway*) hacia la red indicada. Sobre comprasGW, por ejemplo, la tabla podría ser:

Dirección	Máscara	Gateway	Interfaz
172.17.1.0	255.255.255.0	-	fddi0
172.17.4.0	255.255.255.0	172.17.1.2	fddi0
172.17.6.0	255.255.255.0	172.17.1.3	fddi0
0.0.0.0	0.0.0.0	172.17.2.1	fddi0
172.17.2.0	255.255.255.0	-	eth0

El '-' significa que la máquina está directamente conectada y no necesita routing. El procedimiento para identificar si se realiza el routing o no, se lleva a cabo a través de una operación muy simple con dos AND lógicos (subred AND mask y origen AND mask) y una comparación entre los dos resultados. Si son iguales no hay routing, sino que se debe enviar la máquina definida como gateway en cada máquina para que ésta realice el routing del mensaje.

Por ejemplo, un mensaje de la 172.17.2.4 hacia la 172.17.2.6 significará:

```
172.17.2.4 AND 255.255.255.0 = 172.17.2.0
172.17.2.6 AND 255.255.255.0 = 172.17.2.0
```

Como los resultados son iguales, no habrá routing. En cambio, si hacemos lo mismo con 172.17.2.4 hacia 172.17.6.6 podemos ver que habrá un routing a través del 172.17.2.1 con un cambio de interfaz (eth0 a ffdi0) a la 172.17.1.1 y de ésta hacia la 172.17.1.2 con otro cambio de interfaz (fddi0 a eth0) y luego hacia la 172.17.6.6. El routing, por defecto, se utilizará cuando ninguna regla satisfaga la coincidencia. En caso de que dos reglas coincidan, se utilizará aquella que lo haga de modo más preciso, es decir, la que menos ceros tenga. Para construir las tablas de routing, se puede utilizar el comando ***route*** durante el arranque de la máquina, pero si es necesario utilizar reglas más complejas (o routing automático), se puede utilizar el RIP[23] o entre sistemas autónomos, el EGP[24] o también el BGP[25]. Estos protocolos se implementan en el comando ***gated***.

[23] Del inglés *routing information protocol*.

[24] Del inglés *external gateway protocol*.

[25] Del inglés *border gateway protocol*.

Para instalar una máquina sobre una red existente, es necesario, por lo tanto, disponer de la siguiente información obtenida del proveedor de red o de su administrador: dirección IP del nodo, dirección de la red IP, dirección de broadcast, dirección de máscara de red, dirección de router, dirección del DNS.

Si se construye una red que nunca tendrá conexión a Internet, se pueden escoger las direcciones que se prefieran, pero es recomendable mantener un orden adecuado en función del tamaño de red que se desee tener y para evitar problemas de administración dentro de dicha red. A continuación, se verá cómo se define la red y el nodo para una red privada (hay que ser cuidadoso, ya que si se tiene la máquina conectada a la red, se podría perjudicar a otro usuario que tuviera asignada esta dirección).

4. ¿Cómo se debe configurar la red?

4.1. Configuración de la interfaz (NIC)

Una vez cargado el kernel de GNU/Linux, éste ejecuta el comando ***init*** que a su vez lee el archivo de configuración */etc/inittab* y comienza el proceso de inicialización. Generalmente, el *inittab* tiene secuencias tales como: *si::sysinit:/etc/init.d/boot*, que representa el nombre del archivo de comandos (script) que controla las secuencias de inicialización. Generalmente este script llama a otros scripts, entre los cuales se encuentra la inicialización de la red.

Ejemplo

En Debian se ejecuta *etc/init.d/network* para la configuración de la interfaz de red y en función del nivel de arranque; por ejemplo, en el 2 se ejecutarán todos los ficheros S* del directorio */etc/rc2.d* (que son enlaces al directorio */etc/initd*), y en el nivel de apagado, todos los K* del mismo directorio. De este modo, el *script* está sólo una vez (*/etc/init.d*) y de acuerdo a los servicios deseados en ese estado se crea un enlace en el directorio correspondiente a la configuración del nodo-estado.

Los dispositivos de red se crean automáticamente cuando se inicializa el hardware correspondiente. Por ejemplo, el controlador de Ethernet crea las interfaces eth[0..n] secuencialmente cuando se localiza el hardware correspondiente.

Nota

Consultar **man ifconfig** para las diferentes opciones del comando.

A partir de este momento, se puede configurar la interfaz de red, lo cual implica dos pasos: asignar la dirección de red al dispositivo e inicializar los parámetros de la red al sistema. El comando utilizado para ello es el ***ifconfig*** (interfaz configure). Un ejemplo será:

```
ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up
```

Lo cual indica configurar el dispositivo eth0 con dirección IP 192.168.110.23 y máscara de red 255.255.255.0. El up indica que la interfaz pasará al estado activo (para desactivarla debería ejecutarse ***ifconfig eth0 down***). El comando asume que si algunos valores no se indican, son tomados por defecto. En este caso, el kernel configurará esta máquina como Tipo-C y configurará la red con 192.168.110.23 y la dirección de broadcast con 192.168.110.255. Por ejemplo:

```
ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up
```

Existen comandos como el ***ifup*** e ***ifdown***, que permite configurar-desactivar la red en forma más simple utilizando el archivo */etc/network/interfaces* para obtener todos los parámetros necesarios (consultar *man interfaces* para su sintaxis).

En Debian, con el fin de facilitar la configuración de la red, existe otra forma de configurar la red (considerada de alto nivel) que utiliza los comandos mencionados anteriormente *ifup*, *ifdown* y el archivo */etc/network/interfaces*. Si se decide utilizar estos comandos, no se debería configurar la red a bajo nivel, ya que estos comandos son suficientes para configurar/desactivar la red.

Para modificar los parámetros[26] de red de la interfaz eth0, se puede hacer:

```
ifdown eth0
para todos los servicios de red sobre eth0
vi /etc/network/interfaces
edite y modifique los que necesite
ifup eth0
pone en marcha los servicios de red sobre eth0
```

(26)Consultar *man interfaces* en la sección 5 del manual para más información del formato.

Supongamos que desea configurar sobre Debian una interfaz eth0 que tiene una dirección IP fija 192.168.0.123 y con 192.168.0.1 como puerta de enlace (gateway). Se debe editar */etc/network/interfaces* de modo que incluya una sección como:

```
iface eth0 inet static
      address 192.168.0.123
      netmask 255.255.255.0
      gateway 192.168.0.1
```

Si tiene instalado el paquete *resolvconf* puede añadir líneas para especificar la información relativa al DNS. Por ejemplo:

```
iface eth0 inet static
      address 192.168.0.123
      netmask 255.255.255.0
      gateway 192.168.0.1
      dns-search remix.org
      dns-nameservers 195.238.2.21 195.238.2.22
```

Después de que se active la interfaz, los argumentos de las opciones *dns-search* y *dns-nameservers* quedan disponibles para la inclusión en *resolv.conf*. El argumento remix.org de la opción *dns-search* corresponde al argumento de la opción *search* en *resolv.conf* y los argumentos 195.238.2.21 y 195.238.2.22 de la opción *dns-nameservers* corresponden a los argumentos de las opciones *nameserver* en *resolv.conf*. También se puede configurar la red a bajo nivel a través del comando ***ip*** (que es equivalente a **ifconfig** y **route**). Si bien este

Nota: *resolv.conf*

Deberéis consultar el manual para consultar *man resolv.conf*.

comando es mucho más versátil y potente (permite establecer túneles, routing alternativos, etc.) es más complejo y se recomienda utilizar los procedimientos anteriores para configuraciones básicas de la red.

4.1.1. Configuración de red en (estilo) Fedora

Red Hat y Fedora utilizan diferente estructura de ficheros para la configuración de la red: /etc/sysconfig/network. Por ejemplo, para la configuración estática de la red:

```
NETWORKING=yes
HOSTNAME=my-hostname
   Nombre del host definido por el cmd hostname
FORWARD_IPV4=true
   True para NAT firewall gateways y routers.
   False para cualquier otro caso
GATEWAY="XXX.XXX.XXX.YYY"
   Dirección IP de la Puerta de salida a Internet.
```

Para configuración por DHCP se debe quitar la línea de gateway, ya que será asignada por el servidor. Y en caso de incorporar NIS debe agregarse una línea con el servidor de dominio: NISDOMAIN=NISProject1.

Para configurar la interfaz eth0 en el archivo */etc/sysconfig/network-scripts/ifcfg-eth0* (reemplazar las X con los valores adecuados):

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=XXX.XXX.XXX.255
IPADDR=XXX.XXX.XXX.XXX
NETMASK=255.255.255.0
NETWORK=XXX.XXX.XXX.0
ONBOOT=yes    Activará la red en el boot
```

También a partir de FC3 se pueden agregar:

```
TYPE=Ethernet
HWADDR=XX:XX:XX:XX:XX:XX
GATEWAY=XXX.XXX.XXX.XXX
IPV6INIT=no
USERCTL=no
PEERDNS=yes
```

O sino para configuración por DHCP:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

Para deshabilitar DCHP, hay que cambiar BOOTPROTO=dhcp a BOOTPROTO=none. Cualquier cambio en estos ficheros deberá reiniciar los servicios con ***service network restart*** (o sino ***/etc/init.d/network restart***).

Para cambiar el nombre del host se deben seguir estos tres pasos:

1) El comando ***hostname nombre-nuevo***.

2) Cambiar la configuración de la red en */etc/sysconfig/network* editando: HOSTNAME=nombre-nuevo.

3) Restaurando los servicios (o haciendo un reboot):

- ***service network restart*** (*o /etc/init.d/network restart*).
- Reiniciando el *desktop* pasando a modo consola **init 3** y cambiando a modo GUI **init 5**.

Verificar si el nombre tampoco está dado de alta en el */etc/hosts*. El hostname puede ser cambiado en tiempo de ejecución con *sysctl -w kernel.hostname="nombre-nuevo"*.

4.1.2. Configuración de una red Wi-Fi (inalámbrica)

Para la configuración de interfaces Wi-Fi se utilizan básicamente el paquete ***wireless-tools*** (además de **ifconfig** o **ip**). Este paquete utiliza el comando *iwconfig* para configurar una interfaz inalámbrica, pero también se puede hacer a través del */etc/network/interfaces*.

Ejemplo: Configurar una WiFi en Debian (similar en FC)

Supongamos que queremos configurar una tarjeta de red inalámbrica Intel Pro/Wireless 2200BG (muy común en una gran cantidad de portátiles –p. ej., Dell, HP, ...–). Normalmente, el software que controla las tarjetas se divide en dos partes: el módulo software que se cargará en el kernel a través del comando *modprobe* y el *firmware*, que es el código que se cargará en la tarjeta y que nos da el fabricante (consultar la página de Intel para este modelo). Como estamos hablando de módulos, es interesante utilizar el paquete de Debian *module-assistant*, que nos permite crear e instalar fácilmente un módulo (otra opción

sería instalar las fuentes y crear el módulo correspondiente). El software (lo encontramos en la página del fabricante y lo denomina ipw2200) lo compilaremos e instalaremos con el comando ***m-a*** del paquete *module-assistant*.

```
aptget install module-assistant instalación del paquete
m-a -t update
m-a -t -f get ipw2200
m-a -t -build ipw2200
m-a -t install ipw2200
```

Desde la dirección indicada por el fabricante (en su documentación), se descarga la versión del firmaware compatible con la versión del driver, en nuestro caso para el driver versión 1.8 el firmaware es la 2.0.4 obtenida desde la página: http://ipw2200.sourceforge.net/firmware.php. Y a continuación se descomprime e instala el *firmaware*:

```
tar xzvf ipw2200fw2.4.tgz C /tmp/fwr/
cp /tmp/fwr/*.fw /usr/lib/hotplug/firmware/
```

Con esto se copiarán tres paquetes (ipw2200-bss.fw, ipw2200-ibss.fw y ipw2200-sniffer.fw). Luego se carga el módulo con: **modprobe ipw2200**, se reinicia el sistema (reboot) y luego, desde la consola, podemos hacer ***dmesg | grep ipw***, este comando nos mostrará algunas líneas similares a las que se muestran a continuación y que indicarán que el módulo está cargado (se puede verificar con lsmod):

```
ipw2200: Intel(R) PRO/Wireless 2200/2915 Network Driver, git1.0.8
ipw2200: Detected Intel PRO/Wireless 2200BG Network Connection
...
```

Luego se descarga el paquete wirelesstools que contiene iwconfig y, entre otras, con **aptget install wirelesstools** y ejecutamos **iwconfig**; saldrá algo parecido a:

```
eth1 IEEE 802.11b ESSID:"Nombre-de-la-Wifi"
Mode:Managed Frequency:2.437 GHz
Access Point:00:0E:38:84:C8:72
Bit Rate=11 Mb/s TxPower=20 dBm
Security mode:open
...
```

A continuación, hay que configurar el archivo de redes, por ejemplo, **gedit /etc/network/interfaces** y añadir la interfaz wifi eth1, por ejemplo:

```
iface eth1 inet dhcp
   pre-up iwconfig eth1 essid "Nombre de la Wifi"
   pre-up iwconfig eth1 key open XXXXXXXXXX
```

La línea pre-up ejecuta el comando *iwconfig* antes de activar la interfaz. Esta configuración es para cuando se quiere utilizar un servicio en modo DHCP (asignación automática de IP); se debe utilizar en vez de dhcp la palabra static y además poner las siguientes líneas, por ejemplo (como en una tarjeta de cable):

```
address 192.168.1.132
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.1.1
```

Un método alternativo para configurar la interfaz es:

```
iface eth1 inet dhcp
   wireless-essid "Nombre de la Wifi"
   wireless-key 123456789e
```

A continuación se puede poner en marcha la red con *ifup eth1* y nos dará información sobre la conexión y nos indicará su estado y calidad de recepción. Para buscar (scan) las redes WiFi disponibles (puntos de acceso) podemos utilizar **iwlist scan**, lo que nos mostrará información de las redes disponibles, y si nos queremos conectar a una diferente, se puede utilizar el comando ***iwconfig*** para cambiar de red o punto de acceso (access point).

4.2. Configuración del name resolver

El siguiente paso es configurar el name resolver, que convierte nombres tales como *pirulo.remix.com* en 192.168.110.23. El archivo */etc/resolv.conf* es el utilizado para tal fin. Su formato es muy simple (una línea de texto por sentencia). Existen tres palabras clave para tal fin: *domain* (dominio local), *search* (lista de dominios alternativos) y *name server* (la dirección IP del *domain name server*).

4.2.1. Ejemplo de */etc/resolv.conf*

```
domain remix.com
search remix.com piru.com
```

```
name server 192.168.110.1
name server 192.168.110.65
```

Esta lista de servidores de nombre a menudo dependen del entorno de red, que puede cambiar dependiendo de dónde esté o se conecte la máquina. Los programas de conexión a líneas telefónicas (pppd) u obtención de direcciones IP automáticamente (dhclient) son capaces de modificar resolv.conf para insertar o eliminar servidores, pero esta característica no siempre funciona adecuadamente y a veces puede entrar en conflicto y generar configuraciones erróneas. El paquete **resolvconf** (aún en unstable) soluciona de forma adecuada el problema y permite una configuración simple de los servidores de nombre en forma dinámica. **resolvconf** está diseñado para funcionar sin que sea necesaria ninguna configuración manual, no obstante, el paquete es bastante nuevo y puede requerir alguna intervención para lograr que funcione adecuadamente.

Nota

Para más información sobre el paquete resolvconf, podéis consultar la web explicativa: http://packages.debian.org/unstable/net/resolvconf

Un archivo importante es el /etc/*host*.conf, que permite configurar el comportamiento del *name resolver*. Su importancia reside en indicar dónde se resuelve primero la dirección o el nombre de un nodo. Esta consulta puede hacerse al servidor DNS o a tablas locales dentro de la máquina actual (/etc/hosts).

4.2.2. Ejemplo de */etc/host.conf*

```
order hosts,bind
multi on
```

Esta configuración indica que primero se verifique el */etc/hosts* antes de solicitar una petición al DNS y también indica (2.ª línea) que retorne todas las direcciones válidas que se encuentren en */etc/hosts*. Por lo cual, el archivo /etc/hosts es donde se colocan las direcciones locales o también sirve para acceder a nodos sin tener que consultar al DNS.

La consulta es mucho más rápida, pero tiene la desventaja de que si el nodo cambia, la dirección será incorrecta. En un sistema correctamente configurado, sólo deberán aparecer el nodo local y una entrada para la interfaz *loopback*.

4.2.3. Ejemplo de */etc/hosts*

```
127.0.0.1 localhost loopback
192.168.1.2 pirulo.remix.com pirulo
```

Para el nombre de una máquina pueden utilizarse alias, que significa que esa máquina puede llamarse de diferentes maneras para la misma dirección IP. En referencia a la interfaz *loopback*, éste es un tipo especial de interfaz que permite realizar a nodo conexiones consigo misma (por ejemplo, para verificar que el subsistema de red funciona sin acceder a la red). Por defecto, la dirección

IP 127.0.0.1 ha sido asignada específicamente al *loopback* (un comando telnet 127.0.0.1 conectará con la misma máquina). Su configuración es muy fácil (la realizan generalmente los *script* de inicialización de red).

4.2.4. Ejemplo del loopback

```
ifconfig lo 127.0.0.1
route add host 127.0.0.1 lo
```

En la versión 2 de la biblioteca GNU existe un reemplazo importante con respecto a la funcionalidad del archivo *host.conf*. Esta mejora incluye la centralización de información de diferentes servicios para la resolución de nombres, lo cual presenta grandes ventajas para el administrador de red. Toda la información de consulta de nombres y servicios ha sido centralizada en el archivo */etc/nsswitch.conf*, el cual permite al administrador configurar el orden y las bases de datos de modo muy simple. En este archivo cada servicio aparece uno por línea con un conjunto de opciones, donde, por ejemplo, la resolución de nombres de nodo es una de ellas. En éste se indica que el orden de consulta de las bases de datos para obtener el IP del nodo o su nombre será primero el servicio de DNS, que utilizará el archivo */etc/resolv.conf* para determinar la IP del nodo DNS, y en caso de que no pueda obtenerlo, utilizará el de las bases de datos local (/etc/*hosts*). Otras opciones para ello podrían ser *nis, nisplus*, que son otros servicios de información que se describirán en unidades posteriores. También se puede controlar por medio de acciones (entre []) el comportamiento de cada consulta, por ejemplo:

Nota

Ejemplo de nsswitch.conf:

```
hosts: dns files
...
networks: files
```

```
hosts: xfn nisplus dns [NOTFOUND = return] files
```

Esto indica que cuando se realice la consulta al DNS, si no existe un registro para esta consulta, retorne al programa que la hizo con un cero. Puede utilizarse el '!' para negar la acción, por ejemplo:

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

4.3. Configuración del *routing*

Otro aspecto que hay que configurar es el *routing*. Si bien existe el tópico sobre su dificultad, generalmente se necesitan unos requerimientos de *routing* muy simples. En un nodo con múltiples conexiones, el *routing* consiste en decidir dónde hay que enviar y qué se recibe. Un nodo simple (una sola conexión de red) también necesita routing, ya que todos los nodos disponen de un *loopback* y una conexión de red (por ejemplo, Ethernet, PPP, SLIP...). Como se explicó anteriormente, existe una tabla llamada *routing table*, que contiene filas con

Nota

Consulta de tablas de *routing*:
route -n o también
netstat -r

diversos campos, pero con tres de ellos sumamente importantes: dirección de destino, interfaz por donde saldrá el mensaje y dirección IP, que efectuará el siguiente paso en la red (*gateway*).

El comando *route* permite modificar esta tabla para realizar las tareas de *routing* adecuadas. Cuando llega un mensaje, se mira su dirección destino, se compara con las entradas en la tabla y se envía por la interfaz, en la que la dirección coincide mejor con el destino del paquete. Si un *gateway* es especificado, se envía a la interfaz adecuada.

Consideremos, por ejemplo, que nuestro nodo está en una red clase C con dirección 192.168.110.0 y tiene una dirección 192.168.110.23; y el *router* con conexión a Internet es 192.168.110.3. La configuración será:

- Primero la interfaz:

```
ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up
```

- Más adelante, indicar que todos los paquetes con direcciones 192.168.0.* deben ser enviados al dispositivo de red:

```
route add -net 192.1 ethernetmask 255.255.255.0 eth0
```

El *-net* indica que es una ruta de red pero también puede utilizarse *-host* 192.168.110.3. Esta configuración permitirá conectarse a todos los nodos dentro del segmento de red (192.1), pero ¿qué pasará si se desea conectar con otro nodo fuera de este segmento? Sería muy difícil tener todas las entradas adecuadas para todas las máquinas a las cuales se quiere conectar. Para simplificar esta tarea, existe el *default route*, que se utiliza cuando la dirección destino no coincide en la tabla con ninguna de las entradas. Una posibilidad de configuración sería:

```
route add default gw 192.168.110.3 eth0
```

Nota

El gw es la IP o nombre de un *gateway* o nodo *router*.

Una forma alternativa de hacerlo es:

```
ifconfig eth0 inet down deshabilito la interfaz
ifconfig
lo Link encap:Local Loopback
```

```
... (no mostrará ninguna entrada para eth0)
route
... (no mostrará ninguna entrada en la tabla de rutas)
```

Luego se habilita la interfaz con una nueva IP y una la nueva ruta:

```
ifconfig eth0 inet up 192.168.0.111 \
      netmask 255.255.0.0 broadcast 192.168.255.255
route add -net 10.0.0.0 netmask 255.0.0.0 \
      gw 192.168.0.1 dev eth0
```

La barra (\) indica que el comando continúa en la siguiente línea. El resultado:

```
ifconfig
      eth0 Link encap:Ethernet HWaddr 08:00:46:7A:02:B0
      inet addr:192.168.0.111 Bcast: 192.168.255.255 Mask:255.255.0.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      ...
      lo Link encap:Local Loopback
      inet addr:127.0.0.1 Mask:255.0.0.0
      ...
route
Kernel IP routing table
      Destination Gateway Genmask Flags Metric Ref Use Iface
      192.168.0.0 * 255.255.0.0 U 0 0 0 eth0
      10.0.0.0 192.168.0.1 255.0.0.0 UG 0 0 0 eth0
```

Para más información ver los comandos *ifconfig*(8) y *route*(8).

4.4. Configuración del inetd

El siguiente paso en la configuración de red es la configuración de los servidores y servicios que permitirán a otro usuario acceder a la máquina local o a sus servicios. Los programas servidores utilizarán los puertos para escuchar las peticiones de los clientes, los cuales se dirigirán a este servicio como *IP:port*. Los servidores pueden funcionar de dos maneras diferentes: *standalone* (en esta manera el servicio escucha en el puerto asignado y siempre se encuentra activo) o a través del inetd.

El inetd es un servidor que controla y gestiona las conexiones de red de los servicios especificados en el archivo */etc/inetd.conf*, el cual, ante una petición de servicio, pone en marcha el servidor adecuado y le transfiere la comunicación.

Dos archivos importantes necesitan ser configurados: */etc/services* y */etc/inetd.conf*. En el primero se asocian los servicios, los puertos y el protocolo y en el segundo, qué programas servidores responderán ante una petición a un puerto determinado. El formato de */etc/services* es *name port/protocol aliases*, donde el primer campo es nombre del servicio, el segundo, el puerto donde atiende este servicio y el protocolo que utiliza, y el siguiente, un alias del nombre. Por defecto existen una serie de servicios que ya están preconfigurados. A continuación se muestra un ejemplo de */etc/services* (# indica que lo que existe a continuación es un comentario):

```
tcpmux        1/tcp        # TCP port service multiplexer
echo          7/tcp
echo          7/udp
discard       9/tcp        sink null
discard       9/udp        sink null
systat        11/tcp       users
...
ftp           21/tcp
ssh           22/tcp       # SSH Remote Login Protocol
ssh           22/udp       # SSH Remote Login Protocol
telnet        23/tcp
                           # 24 - private
smtp          25/tcp       mail
...
```

El archivo */etc/inetd.conf* es la configuración para el servicio maestro de red (*inetd server daemon*). Cada línea contiene siete campos separados por espacios: *service socket_type proto flags user server_path server_args*, donde service es el servicio descrito en la primera columna de */etc/services*, *socket_type* es el tipo de socket (valores posibles *stream*, *dgram*, *raw*, *rdm*, o *seqpacket*), proto es el protocolo válido para esta entrada (debe coincidir con el de */etc/services*), flags indica la acción que tomar cuando existe una nueva conexión sobre un servicio que se encuentra atendiendo a otra conexión (*wait* le dice a inetd no poner en marcha un nuevo servidor o *nowait* significa que inetd debe poner en marcha un nuevo servidor). User será el usuario con el cual se identificará quién ha puesto en marcha el servicio, *server_path* es el directorio donde se encuentra el servidor y *server_args* son argumentos posibles que serán pasados al servidor.

Un ejemplo de algunas líneas de */etc/inetd.conf es* (recordar que # significa comentario, por lo cual, si un servicio tiene # antes de nombre, significa que no se encuentra disponible):

```
...
telnet   stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/sbin/in.telnetd
ftp      stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/sbin/in.ftpd
# fsp    dgram    udp   wait     root   /usr/sbin/tcpd   /usr/sbin/in.fspd
shell    stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/sbin/in.rshd
login    stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/sbin/in.rlogind
# exec   stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/sbin/in.rexecd ...
...
```

A partir de Debian Woody 3.0 r1, la funcionalidad de inetd ha sido reemplazada por xinetd (recomendable), el cual necesita el archivo de configuración */etc/xinetd.conf* (ver el final del módulo). Si se desea poner en marcha el servicio de inetd, se debe ejecutar (y crear los links adecuados en los directorios */etc/rcX.d*) */etc/init.d/inetd.real start* (ved un ejemplo de configuraciones en el punto 15 del apartado 12 "Configuraciones avanzadas y herremientas").

Además de la configuración de inetd o xinetd, la configuración típica de los servicios de red en un entorno de escritorio o servidor básico podría incluir además:

Ved también

Para ver más sobre la configuración típica de los servicios de red en un entorno de escritorio o servidor básico, podéis ver el módulo de servidores (módulo 2) de la asignatura *Administración avanzada de sistemas GNU-Linux*.

- **ssh**: conexión interactiva segura como reemplazo de telnet e incluye dos archivos de configuración */etc/ssh/ssh_config* (para el cliente) y */etc/ssh/sshd_config* (para el servidor).

- **exim**: agente de transporte de correo (MTA), incluye los archivos de configuración: */etc/exim/exim.conf, /etc/mailname, /etc/aliases, /etc/email-addresses*.

- **fetchmail**: daemon para descargar el correo de una cuenta POP3, */etc/fetchmailrc*.

- **procmail**: programa para filtrar y distribuir el correo local, *~/.procmailrc*.

- **tcpd**: servicios de filtros de máquinas y dominios habilitados y deshabilitados para conectarse al servidor (wrappers): */etc/hosts.allow, /etc/hosts.deny*.

- **DHCP**: servicio para la gestión (servidor) u obtención de IP (cliente), */etc/dhcp3/dhclient.conf* (cliente), */etc/default/dhcp3-server* (servidor), */etc/dhcp3/dhcpd.conf* (servidor).

- **CVS**: sistema de control de versiones concurrentes, */etc/cvs-cron.conf,*

/etc/cvs-pserver.conf.

- **NFS**: sistema de archivos de red, */etc/exports*.
- **Samba**: sistema de archivos de red y compartición de impresoras en redes Windows, */etc/samba/smb.conf*.
- **lpr**: daemon para el sistema de impresión, */etc/printcap* (para el sistema **lpr** –no para CUPS–).
- **Apache** y **Apache2**: servidor de web, */etc/apache/** y */etc/apache2/**.
- **squid**: servidor *proxy-caché*, */etc/squid/**.

4.5. Configuración adicional: protocols y networks

Existen otros archivos de configuración que en la mayoría de los casos no se utilizan pero que pueden ser interesantes. El */etc/protocols* es un archivo que relaciona identificadores de protocolos con nombres de protocolos, así, los programadores pueden especificar los protocolos por sus nombres en los programas.

Ejemplo de */etc/protocols*

```
ip          0     IP   # internet protocol, pseudo protocol number
#hopopt     0     HOPOPT   # IPv6 Hop-by-Hop Option [RFC1883]
icmp        1     ICMP   # internet control message protocol
```

El archivo */etc/networks* tiene una función similar a */etc/hosts*, pero con respecto a las redes, indica nombres de red con relación a su dirección IP (el comando *route* mostrará el nombre de la red y no su dirección en este caso).

Ejemplo de */etc/networks*

```
loopnet 127.0.0.0
localnet 192.168.0.0
amprnet 44.0.0.0 ...
```

4.6. Aspectos de seguridad

Es importante tener en cuenta los aspectos de seguridad en las conexiones a red, ya que una fuente de ataques importantes se produce a través de la red. Ya se hablará más sobre este tema en la unidad correspondiente a seguridad; sin embargo, hay unas cuantas recomendaciones básicas que deben tenerse en cuenta para minimizar los riesgos inmediatamente antes y después de configurar la red de nuestro ordenador:

a) No activar servicios en */etc/inetd.conf* que no se utilizarán, insertar un # antes del nombre para evitar fuentes de riesgo.

b) Modificar el archivo */etc/ftpusers* para denegar que ciertos usuarios puedan tener conexión vía ftp con su máquina.

c) Modificar el archivo */etc/securetty* para indicar desde qué terminales (un nombre por línea), por ejemplo: tty1 tty2 tty3 tty4, se permite la conexión del superusuario (*root*). Desde los terminales restantes, *root* no podrá conectarse.

d) Utilizar el programa **tcpd**. Este servidor es un wrapper que permite aceptar-negar un servicio desde un determinado nodo y se coloca en el */etc/inetd.conf* como intermediario de un servicio. El **tcpd** verifica unas reglas de acceso en dos archivos: */etc/hosts.allow /etc/host.deny*

Si se acepta la conexión, pone en marcha el servicio adecuado pasado como argumento, por ejemplo, la línea del servicio de ftp antes mostrada en inetd.conf: *ftp stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.ftpd*.

tcpd primero busca */etc/hosts.allow* y luego */etc/hosts.deny*. El archivo *hosts.deny* contiene la información sobre cuáles son los nodos que no tienen acceso a un servicio dentro de esta máquina. Una configuración restrictiva es ALL: ALL, ya que sólo se permitirá el acceso a los servicios desde los nodos declarados en */etc/hosts.allow*.

El archivo */etc/hosts.equiv* permite el acceso a esta máquina sin tener que introducir una clave de acceso (password). Se recomienda no usar este mecanismo y aconsejar a los usuarios no utilizar el equivalente desde la cuenta de usuario a través del archivo *.rhosts*.

En Debian es importante configurar */etc/security/access.conf*, el archivo que indica las reglas de quién y desde dónde se puede conectar (*login*) a esta máquina. Este archivo tiene una línea por orden con tres campos separados por ':' del tipo *permiso: usuarios:origen*. El primero será un + o - (acceso denegado), el segundo un nombre de usuario/s, grupo o user@host, y el tercero un nombre de un dispositivo, nodo, dominio, direcciones de nodo o de redes, o ALL.

Ejemplo de *access.conf*

Este comando no permite root logins sobre tty1:

```
ALL EXCEPT root:tty1 ...
```

Permite acceder a *u1*, *u2*, *g1* y todos los de dominio remix.com:

```
+:u1 u2 g1 .remix.com:ALL
```

4.7. Opciones del IP

Existen una serie de opciones sobre el tráfico IP que es conveniente mencionar. Su configuración se realiza a través de la inicialización del archivo correspondiente en directorio */proc/sys/net/ipv4/*. El nombre del archivo es el mismo que el del comando y para activarlos se debe poner un 1 dentro del archivo, y un 0 para desactivarlo. Por ejemplo, si se quiere activar *ip_forward*, se debería ejecutar:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Los más utilizados son: *ip_forward* utilizado para el routing entre interfaces o con *IP Masquerading; ip_default_ttl,* que es el tiempo de vida para un paquete IP (64 milisegundos por defecto) *ip_bootp_agent* variable lógica (boolean), que acepta paquetes (o no) con dirección origen del tipo 0.b.c.d y destino de este nodo, *broadcast* o *multicast*.

Comandos para la solución de problemas con la red

Si tiene problemas en la configuración de la red, se puede comenzar verificando la salida de los siguientes comandos para obtener una primera idea:

```
ifconfig
cat /proc/pci
cat /proc/interrupts
dmesg | more
```

Para verificar la conexión a la red, se pueden utilizar los siguientes comandos (se debe tener instalado netkit-ping, traceroute, dnsutils, iptables y net-tools):

```
ping uoc.edu                  # verificar la conexión a Internet
traceroute uoc.edu            # rastrear paquetes IP
ifconfig                      # verificar la configuración del host
route -n                      # verificar la configuración de la ruta
dig [@dns.uoc.edu] www.uoc.edu # verificar registros de www.uoc.edu
                              # sobre el servidor dns.uoc.edu
iptables -L -n |less          # verificar filtrado de paquetes (kernel >=2.4)
netstat -a                    # muestra todos los puertos abiertos
netstat -l --inet             # muestra los puertos en escucha
netstat -ln --tcp             # mostrar puertos tcp en escucha (numérico)
```

5. Configuración del DHCP

DHCP son las siglas de *dynamic host configuration protocol*. Su configuración es muy simple y sirve para que, en lugar de configurar cada nodo de una red individualmente, se pueda hacer de forma centralizada y su administración sea más fácil. La configuración de un cliente es muy fácil, ya que solo se debe instalar uno de los siguientes paquetes: *dhcp3-client* (versión 3, Internet Software Consortium), *dhcpcd* (Yoichi Hariguchi y Sergei Viznyuk), *pump* (Red Hat), agregando la palabra *dhcp* en la entrada correspondiente a la interfaz que se desea que funcione bajo el cliente *dhcp* (p.e. /etc/network/interfaces debe tener iface eth0 inet dhcp...).

La configuración del servidor requiere un poco más de atención, pero no presenta complicaciones. Primero, para que el servidor pueda servir a todos los clientes DHCP (incluido Windows), deben realizarse algunas cuestiones previas relacionadas con las direcciones de broadcast. Para ello, primero el servidor debe poder enviar mensajes a la dirección 255.255.255.255, lo cual no es válido en GNU/Linux. Para probarlo, ejecútese:

```
route add -host 255.255.255.255 dev eth0
```

Si aparece el siguiente mensaje *255.255.255.255: Unknown host*, debe añadirse la siguiente entrada en */etc/hosts*: *255.255.255.255 dhcp* e intentar nuevamente:

```
route add -host dhcp dev eth0
```

La configuración de *dhcpd* se puede realizar con la interfaz gráfica de linuxconf o bien editar */etc/dhcpd.conf*. Un ejemplo de este archivo es:

```
# Ejemplo de /etc/dhcpd.conf:
default-lease-time 1200;
max-lease-time 9200;
option domain-name "remix.com";
deny unknown-clients;
deny bootp;
option broadcast-address 192.168.11.255;
option routers 192.168.11.254;
option domain-name-servers 192.168.11.1, 192.168.168.11.2;
subnet 192.168.11.0 netmask 255.255.255.0
{ not authoritative;
```

```
        range 192.168.11.1 192.168.11.254
        host marte {
           hardware ethernet 00:00:95:C7:06:4C;
           fixed address 192.168.11.146;
           option host-name "marte";
           }
        host saturno {
           hardware ethernet 00:00:95:C7:06:44;
           fixed address 192.168.11.147;
           option host-name "saturno";
           }
     }
```

Esto permitirá al servidor asignar el rango de direcciones 192.168.11.1 al 192.168.11.254 tal y como se describe cada nodo. Si no existe el segmento *host* { ... } correspondiente, se asignan aleatoriamente. Las IP son asignadas por un tiempo mínimo de 1.200 segundos y máximo de 9.200 (en caso de no existir estos parámetros, se asignan indefinidamente).

Antes de ejecutar el servidor, debe verificarse si existe el fichero */var/state/dhcp/dhcpd.leases* (en caso contrario, habrá que crearlo con ***touch /var/state/dhcp/dhcpd.leases***). Para ejecutar el servidor: ***/usr/sbin/dhcpd*** (o bien ponerlo en los *scripts* de inicialización). Con ***/usr/sbin/dhcpd -d -f*** se podrá ver la actividad del servidor sobre la consola del sistema [Mou01, Rid00, KD00, Dra99].

6. IP Aliasing

Existen algunas aplicaciones donde es útil configurar múltiples direcciones IP a un único dispositivo de red. Los ISP[27] utilizan frecuentemente esta característica para proveer de características personalizadas (por ejemplo, de World Wide Web y FTP) a sus usuarios. Para ello, el kernel debe estar compilado con las opciones de Network Aliasing e IP (aliasing support). Después de instalado el nuevo kernel, la configuración es muy fácil. Los alias son anexados a dispositivos de red virtuales asociados al nuevo dispositivo con un formato tal como:

[27] Del inglés *internet service providers*

```
dispositivo: número virtual
```

Por ejemplo:

```
eth0:0, ppp0:8
```

Consideremos que tenemos una red Ethernet que soporta dos diferentes subredes IP simultáneamente y que nuestra máquina desea tener acceso directo a ellas. Un ejemplo de configuración sería:

```
ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up
route add -net 192.168.110.0 netmask 255.255.255.0 eth0
ifconfig eth0:0 192.168.10.23 netmask 255.255.255.0 up
route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

Lo cual significa que tendremos dos IP 192.168.110.23 y 192.168.10.23 para la misma NIC. Para borrar un alias, agregar un '-' al final del nombre (por ejemplo, ifconfig eth0:0- 0) [Mou01, Ran05].

Un caso típico es que se desee configurar una única tarjeta Ethernet para que sea la interfaz de distintas subredes IP. Por ejemplo, supongamos que se tiene una máquina que se encuentra en una red LAN 192.168.0.x/24, y se desea conectar la máquina a Internet usando una dirección IP pública proporcionada con DHCP usando su tarjeta Ethernet existente. Por ejemplo, se puede hacer como en el ejemplo anterior o también editar el archivo */etc/network/interfaces*, de modo que incluya una sección similar a la siguiente:

```
iface eth0 inet static
   address 192.168.0.1
   netmask 255.255.255.0
```

```
    network 192.168.0.0
    broadcast 192.168.0.255

iface eth0:0 inet dhcp
```

La interfaz eth0:0 es una interfaz virtual y al activarse, también lo hará su padre eth0.

7. IP Masquerade

El IP Masquerade es un recurso para que un conjunto de máquinas puedan utilizar una única dirección IP. Esto permite que los nodos ocultos puedan salir hacia Internet (son los que utilizan una IP privada, por ejemplo, 198.162.10.1); pero no pueden aceptar llamadas o servicios del exterior directamente, sino a través de la máquina que tiene la IP real.

Esto significa que algunos servicios no funcionan (por ejemplo, talk) y otros deben ser configurados en modo PASV (pasivo) para que funcionen (por ejemplo, FTP). Sin embargo, WWW, telnet o irc funcionan adecuadamente. El kernel debe estar configurado con las siguientes opciones: *Network firewalls, TCP/IP networking, IP:forwarding/gatewaying, IP: masquerading*. Normalmente, la configuración más común es disponer de una máquina con una conexión SLIP o PPP y tener otro dispositivo de red (por ejemplo, una tarjeta Ethernet) con una dirección de red reservada. Como vimos, y de acuerdo a la RFC 1918, se pueden utilizar como IP privadas los siguientes rangos de direcciones (IP/Mask):

– 10.0.0.0/255.0.0.0

– 172.16.0.0/255.240.0.0

– 192.168.0.0/255.255.0.0

Los nodos que deben ser ocultados (*masqueraded*) estarán dentro de esta segunda red. Cada una de estas máquinas debería tener la dirección de la máquina que realiza el masquerade, como default gateway o router. Sobre dicha máquina podemos configurar:

- *Network route* para Ethernet considerando que la red tiene un IP=192.168.1.0/255.255.255.0:

```
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

- Default *route* para el resto de Internet:

```
route add default ppp0
```

- Todos los nodos sobre la red 192.168.1/24 serán *masqueraded*:

```
ipchains -A forward -s 192.168.1.0/24 -j MASQ
```

- Si se utiliza iptables sobre un *kernel* 2.4 o superior:

```
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Consultar las referencias y la unidad que trata sobre la seguridad de la información de ipchains e iptables. [Ran05, KD00].

8. NAT con el kernel 2.2 o superiores

El IP Network Address Translation NAT es el reemplazo que deja obsoleto las prestaciones de GNU/Linux IP Masquerade y que aporta nuevas prestaciones al servicio. Dentro de las mejoras introducidas en la pila de TCP/IP del núcleo 2.2 de GNU/Linux tenemos que el NAT forma parte del kernel. Para utilizarlo, es necesario que el kernel se compile con: CONFIG_IP_ADVANCED_ROUTER, CONFIG_IP_MULTIPLE_TABLES y CONFIG_IP_ROUTE_NAT. Y si se necesita control exhaustivo de las reglas NAT (por ejemplo, para activar el cortafuegos firewalling) debe estar también CONFIG_IP_FIREWALL y CONFIG_IP_ROUTE_FWMARK. Para trabajar con estas nuevas características, es necesario usar el programa ip (incluido en las distribuciones más importantes a partir de la versión 2.4 del kernel http://es.wikipedia.org/wiki/Iproute2). Entonces, para trasladar direcciones de paquetes de entrada se puede utilizar:

```
ip route add nat <extaddr>[/<masklen>] via <intaddr>
```

Esto hará que un paquete de entrada destinado a ext-addr (la dirección visible desde fuera de Internet) se transcribe su dirección destino a int-addr (la dirección de su red interna por medio de su gateway/firewall). El paquete se encamina de acuerdo a la tabla local de route. Se pueden trasladar direcciones simples o bloques. Por ejemplo:

```
ip route add nat 240.0.11.34 via 192.109.0.2
ip route add nat 240.0.11.32/27 via 192.109.0.0
```

El primero hace que la dirección interna 192.109.0.2 sea accesible como 240.0.11.34. El segundo reubica (*remapping*) el block 192.109.0.0-31 a 240.0.11.32-63. En este caso se han utilizado como ejemplo traslaciones a direcciones de la clase DE tal como 240.0.*.* con el fin de no utilizar ninguna dirección pública. El usuario deberá reemplazar estas direcciones (240.0.11.34 y 240.0.11.32-63) por las correspondientes direcciones públicas a las que desee realizar la traslación [Ran05].

9. ¿Cómo configurar una conexión DialUP y PPP?

Si bien hoy en día es poco habitual trabajar con módem, ya que se tienen soluciones de ASDL a mejores precios y ancho de banda, se realizará una pequeña introducción sobre la configuración de una conexión dial-up sobre PPP en GNU/Linux que es muy simple.

PPP[28] que permite realizar IP-Links entre dos ordenadores con un módem (considerar que debe ser un módem soportado por GNU/Linux, ya que no todos, especialmente los internos o los conocidos como Winmodems, se pueden configurar, puesto que muchos de ellos necesitan software adicional para establecer la comunicación) [Vas00, Law07, Sec00].

(28)Del inglés *point to point protocol*.

Como pasos previos se debe disponer de la siguiente información: el *init-string* del módem (normalmente no es necesario pero si la necesita y no la tiene disponible, se puede utilizar ATZ, que funciona en la mayoría de los módems, o consultar listas especializadas de *init-string*).

Además, necesitará los datos del ISP: identificación de conexión (*login name*), clave (*password*) y número de teléfono. Direcciones de DNS serían aconsejables, pero es opcional en las versiones actuales de pppd. Verificar además que su módem está correctamente conectado. Con un módem externo se debe ejecutar *echo > /dev/ttyS0* y mirar las luces del módem por si tienen actividad. En caso contrario, intentar con *ttyS1* por si el módem está conectado al 2.° puerto serie. Con un módem interno consultar el manual de hardware soportado para ver si este módem puede ser reconocido por GNU/Linux, y en caso afirmativo, podría tener que reconfigurar el *kernel* para utilizarlo. También puede utilizar *cat /proc/pci* por si se encuentra en el bus PCI [PPP00].

La forma más fácil de configurar ahora el módem es a través del paquete **kppp** (debe instalar los paquetes kdenetwork-ppp* y ppp*). Sobre un terminal, ejecútese ***/usr/bin/kppp***. Sobre la ventana, complétense las opciones siguientes:

- Accounts -> New Connection
- Dial -> Authentication -> 'PAP/CHAP'
- Store Password -> yes
- IP -> Dynamic IP Address
- Autoconfigure hostname -> No
- *Gateway* -> Default *Gateway* -> Assign the Default Route
- DNS -> Configuration Automatic -> Disable existing DNS
- Device -> ttyS1(com1) o ttyS2 (com2)
- Modem -> Query Modem para ver los resultados (si no obtiene resultados, cambie el dispositivo ttySx).

Entraremos *login* y *password*, y estaremos conectados a Internet (para verificar la conexión se puede ejecutar ping www.google.com, por ejemplo). Aquí se ha utilizado el paquete **kppp**, pero igualmente podría utilizarse **linuxconf** o **gnomeppp** indistintamente.

Una manera rápida de configurar *pppd* en Debian consiste en usar el programa **pppconfig**, que viene con el paquete del mismo nombre. **pppconfig** configura los archivos como los anteriores después de formular preguntas al usuario a través de una interfaz de menús. Otra opción diferente para usar *pppd* consiste en ejecutarlo desde **wvdial**, que viene con el paquete *wvdial*. En vez de hacer que **pppd** ejecute chat para marcar y negociar la conexión, *wvdial* realiza el marcado, la negociación inicial y luego inicia *pppd* para que haga el resto. En la mayoría de los casos dando solamente el número telefónico, el nombre de usuario y la contraseña, wvdial logra establecer la conexión.

Una vez configurado PPP para que funcione por ejemplo con mi_isp, se debe editar */etc/network/interfaces*, de modo que incluya una sección como la siguiente (los comandos *ifup, ifdown* utilizan los comandos *pon* y *poff* para configurar interfaces PPP):

```
iface ppp0 inet ppp
   provider mi_isp
con esta sección, ifup ppp0 hace:
pon mi_isp
```

Actualmente no es posible usar **ifup-down** para realizar una configuración auxiliar de las interfaces PPP. Como *pon* desaparece antes que pppd haya terminado de establecer la conexión, **ifup** ejecuta los *scripts* up antes que la interfaz PPP esté lista para usar. Hasta que se solucione este fallo sigue siendo necesario realizar una configuración posterior en */etc/ppp/ip-up* o */etc/ppp/ip-up.d/*.

Muchos proveedores de servicios de internet (ISPs) de banda ancha utilizan PPP para negociar las conexiones incluso cuando las máquinas de los clientes están conectados mediante Ethernet y/o redes ATM. Esto se logra mediante PPP sobre Ethernet (PPPoE), que es una técnica para el encapsulamiento del flujo PPP dentro de las tramas Ethernet. Supongamos que el ISP se llama mi_isp. Primero hay que configurar PPP y PPPoE para *mi_isp*. La manera más fácil de hacerlo consiste en instalar el paquete **pppoeconf** y ejecutar **pppoeconf** desde la consola. A continuación, editar */etc/network/interfaces* de modo que incluya un fragmento como el siguiente:

```
iface eth0 inet ppp
   provider mi_isp
```

A veces surgen problemas con PPPoE relativos a la unidad de transmisión máxima (MTU[29]) en líneas DSL[30]; se puede consultar el DSL-HOWTO para más detalles. También debe tenerse en cuenta si su módem posee un *router*, porque el módem/router maneja por sí mismo la conexión PPPoE y aparece del lado de la LAN como una simple puerta de enlace Ethernet a Internet.

(29) Del inglés *maximum transmit unit*.

(30) Del inglés *digital subscriber line*.

10. Configuración de la red mediante hotplug

El paquete *hotplug* permite el soporte de arranque en caliente (se debe tener instalado el paquete del mismo nombre). El *hardware* de red se puede conectar en caliente ya sea durante el arranque, tras haber insertado la tarjeta en la máquina (una tarjeta PCMCIA, por ejemplo), o después de que una utilidad como *discover* se haya ejecutado y cargado los módulos necesarios. Cuando el *kernel* detecta nuevo *hardware*, inicializa el controlador para este *hardware* y luego ejecuta el programa *hotplug* para configurarlo. Si más tarde se elimina el *hardware*, ejecuta nuevamente *hotplug* con parámetros diferentes. En Debian, cuando se llama a *hotplug* éste ejecuta los scripts de */etc/hotplug/* y */etc/hotplug.d/*. El hardware de red recientemente conectado es configurado por el */etc/hotplug/net.agent*. Supongamos que la tarjeta de red PCMCIA ha sido conectada lo que implica que la interfaz eth0 esta lista para usar.

/etc/hotplug/net.agent hace lo siguiente:

```
ifup eth0=hotplug
```

A menos que haya añadido una interfaz lógica llamada hotplug en */etc/network/interfaces*, este comando no hará nada. Para que este comando configure eth0, se debe añadir las siguientes líneas al */etc/network/interfaces*:

```
mapping hotplug
   script echo
```

Si sólo desea que eth0 se active en caliente y no otras interfaces, utilizar *grep* en vez de *echo* como se muestra a continuación:

```
mapping hotplug
   script grep
   map eth0
```

ifplugd activa o desactiva una interfaz según si el *hardware* subyacente está o no conectado a la red. El programa puede detectar un cable conectado a una interfaz Ethernet o un punto de acceso asociado a una interfaz Wi-Fi. Cuando **ifplugd** ve que el estado del enlace ha cambiado ejecuta un *script* que por defecto ejecuta **ifup** o **ifdown** para la interfaz. **ifplugd** funciona en combinación con **hotplug**. Al insertar una tarjeta, lo que significa que la interfaz está lista

para usar, */etc/hotplug.d/net/ifplugd.hotplug* inicia una instancia de **ifplugd** para dicha interfaz. Cuando **ifplugd** detecta que la tarjeta es conectada a una red, ejecuta ifup para esta interfaz.

Para asociar una tarjeta Wi-Fi con un punto de acceso, puede que necesite programarla con una clave de cifrado WEP adecuada. Si está utilizando **ifplugd** para controlar **ifup** como se explicó anteriormente, entonces, evidentemente, no podrá configurar la clave de cifrado usando **ifup**, ya que éste sólo es llamado después de que la tarjeta ha sido asociada. La solución más simple es usar **waproamd**, que configura la clave de cifrado WEP según los puntos de acceso disponibles, que se descubren mediante la búsqueda de la redes WiFi. Para más información consultar **man waproamd** y la información del paquete.

11. *Virtual private network* (VPN)

Una VPN[31] es una red que utiliza Internet como transporte de datos, pero impide que éstos puedan ser accedidos por miembros externos a ella.

(31)Del inglés *virtual private network*.

Tener una red con VPN significa tener nodos unidos a través de un túnel por donde viaja el tráfico y donde nadie puede interactuar con él. Se utiliza cuando se tienen usuarios remotos que acceden a una red corporativa para mantener la seguridad y privacidad de los datos. Para configurar una VPN, se pueden utilizar diversos métodos SSH (SSL), CIPE, IPSec, PPTP, que pueden consultarse en las referencias (se recomienda consultar VPN PPP-SSH HOWTO, por Scott Bronson, VPN-HOWTO de Matthew D. Wilson) [Bro01, Wil02].

Para realizar las pruebas de configuración, en este apartado se utilizará la **OpenVPN**, que es una solución basada en SSL VPN, y se puede usar para un amplio rango de soluciones, por ejemplo, acceso remoto, VPN punto a punto, redes WiFi seguras o redes distribuidas empresariales. OpenVPN implementa OSI layer 2 o 3 utilizando protocolos SSL/TLS y soporta autentificación basada en certificados, tarjetas (*smart cards*), y otros métodos de certificación. OpenVPN no es un servidor *proxy* de aplicaciones ni opera a través de un *web browser*.

Para analizar este servicio utilizaremos una opción de la OpenVPN llamada *OpenVPN for Static key* configurations, que ofrece una forma simple de configurar una VPN ideal para pruebas o para conexiones punto a punto. Sus ventajas son simplicidad, y no es necesario un certificado X509 PKI[32] para mantener la VPN. Las desventajas son que sólo permite 1 cliente y un servidor. Al no utilizar llave pública y llave privada, puede haber igualdad de claves con sesiones anteriores, debe existir, pues, una llave en modo texto en cada *peer*, y la llave secreta debe ser intercambiada anteriormente por un canal seguro.

(32)Del inglés *public key infrastructure*.

11.1. Ejemplo simple

En este ejemplo se configurará un túnel VPN sobre un servidor con IP=10.8.0.1 y un cliente con IP=10.8.0.2. La comunicación será encriptada entre el cliente y el servidor sobre UDP *port* 1194, que es el puerto por defecto de OpenVPN. Después de instalar el paquete se deberá generar la llave estática:

```
openvpn --genkey --secret static.key
```

Después se debe copiar el archivo *static.key* en el otro *peer* sobre un canal seguro (por ejemplo, utilizando **ssh** o **scp**). El archivo de configuración del servidor *openVPN_server* por ejemplo:

```
dev tun
ifconfig 10.8.0.1 10.8.0.2
secret static.key
```

El archivo de configuración del cliente, por ejemplo, *openVPN_client*:

```
remote myremote.mydomain
dev tun
ifconfig 10.8.0.2 10.8.0.1
secret static.key
```

Antes de verificar el funcionamiento de la VPN, debe asegurarse en el *firewall* que el puerto 1194 UDP está abierto sobre el servidor y que la interfaz virtual **tun0** usada por OpenVPN no está bloqueada ni sobre el cliente ni sobre el servidor. Tenga en mente que el 90% de los problemas de conexión encontrados por usuarios nuevos de OpenVPN están relacionados con el *firewall*.

Para verificar la OpenVPN entre dos máquinas, deberá cambiar las IP por las reales y el dominio por el que tenga y luego ejecutar del lado servidor:

```
openvpn [server config file]
```

El cual dará una salida como:

```
Sun Feb 6 20:46:38 2005 OpenVPN 2.0_rc12 i686-suse-linux [SSL] [LZO] [EPOLL]
built on Feb 5 2005
Sun Feb 6 20:46:38 2005 Diffie-Hellman initialized with 1024 bit key
Sun Feb 6 20:46:38 2005 TLS-Auth MTU parms [ L:1542 D:138 EF:38 EB:0 ET:0 EL:0 ]
Sun Feb 6 20:46:38 2005 TUN/TAP device tun1 opened
Sun Feb 6 20:46:38 2005 /sbin/ifconfig tun1 10.8.0.1 pointopoint 10.8.0.2 mtu 1500
Sun Feb 6 20:46:38 2005 /sbin/route add -net 10.8.0.0 netmask 255.255.255.0 gw 10.8.0.2
Sun Feb 6 20:46:38 2005 Data Channel MTU parms [ L:1542 D:1450 EF:42 EB:23
ET:0 EL:0 AF:3/1 ]
Sun Feb 6 20:46:38 2005 UDPv4 link local (bound): [undef]:1194
Sun Feb 6 20:46:38 2005 UDPv4 link remote: [undef]
Sun Feb 6 20:46:38 2005 MULTI: multi_init called, r=256 v=256
Sun Feb 6 20:46:38 2005 IFCONFIG POOL: base=10.8.0.4 size=62
Sun Feb 6 20:46:38 2005 IFCONFIG POOL LIST
```

```
Sun Feb 6 20:46:38 2005 Initialization Sequence Completed
```

Y del lado cliente:

```
openvpn [client config file]
```

Para verificar que funciona, se puede hacer ping 10.8.0.2 desde el server y ping 10.8.0.1 desde el cliente.

Nota

Para más información podéis consultar la web de OpenVPN: http://openvpn.net/howto.html

Para agregar compresión sobre el link, debe añadirse la siguiente línea a los dos archivos de configuración:

```
comp-lzo
```

Para proteger la conexión a través de un NAT *router/firewall alive*, y seguir los cambios de IP a través de un DNS, si uno de los *peers* cambia, agregar a los dos archivos de configuración:

```
keepalive 10 60
ping-timer-rem
persist-tun
persist-key
```

Para ejecutarse como daemon con los privilegios de user/group **nobody**, agregar a los archivos de configuración:

```
user nobody
group nobody
daemon
```

11.2. Configuración (manual) de un cliente Debian para acceder a un VPN sobre un túnel pptp

En primer lugar se debe instalar el cliente PPTP[33] y no es necesario tener soporte **MPPE** al kernel.

(33)Del inglés *point-to-point tunneling protocol*.

Para ello haremos:

```
apt-get update
apt-get install pptp-linux
apt-get install resolvconf
```

Contestar SI si sale el mensaje: *Append original file to dynamic file?*

Después, reiniciar las interfaces con:

```
/etc/init.d/ifupdown restart
```

O con:

```
/etc/init.d/networking restart
```

También es necesario tener instalado el paquete iproute:

```
apt-get install iproute
```

Para la configuración del cliente crearemos en su directorio (por defecto */etc/ppp/*) el fichero */etc/ppp/options.pptp* (este contendrá opciones comunes a todos los túneles que se creen en el equipo):

```
lock noauth nobsdcomp nodeflate
```

Agregamos la siguiente línea al archivo*/etc/ppp/chap-secrets*:

```
usuario PPTP passwd *
```

Donde dice usuario se debe poner el nombre del usuario del servidor de VPN y en *passwd* la palabra clave de acceso acabando la línea con un "*".

Se debe crear el archivo */etc/ppp/peers/uoc* con los parámetros de configuración del túnel:

```
pty "pptp nombre.dominio --nolaunchpppd"
name usuario
remotename PPTP
usepeerdns
defaultroute replacedefaultroute
file /etc/ppp/options.pptp
ipparam uoc
connect "route add nombre.dominio gw `ip route | \
grep default | cut -f3 -d' '`"
```

Donde dice usuario se debe poner el nombre del usuario, nombre.dominio es el nombre del servidor VPN, por ejemplo, vpngw.uoc.es VPN y uoc es el nombre del tunel que crearemos y que se referirá al archivo */etc/ppp/peers/uoc*.

Antes de finalizar debemos crear un par de *scripts* que encaminarán los paquetes por el túnel hacia Internet y lo cerrarán cuando se haya terminado.

Creamos el archivo */etc/ppp/ip-up.d/uoc* con permisos de ejecución y el siguiente contenido:

```
#!/bin/sh
cat /etc/ppp/resolv.conf | resolvconf -a tun0
```

Le cambiamos los atributos de ejecución:

```
chmod +x /etc/ppp/ip-up.d/uoc
```

Finalmente, creamos el archivo */etc/ppp/ip-down.d/uoc* con el siguiente contenido:

```
#!/bin/sh
route del nombre.dominio
# cambiar por el servidor, por ejemplo, vpngw.uoc.es
resolvconf -d tun0
```

Y también con permisos de ejecución:

```
chmod +x /etc/ppp/ip-down.d/uoc
```

Ahora sólo falta poner en marcha el túnel con ***pon uoc***. Después de poner en marcha el túnel y de hacer **ifconfig**, veremos una nueva interfaz de red con el nombre de **ppp0**. Para finalizar la utilización del túnel, solo deberemos hacer **poff uoc**, con lo cual desaparecerá la interfaz **ppp0**.

Existe una interfaz gráfica llamada **kvpnc** para configurar diferentes clientes de VPN. En esta aplicación no es necesario crear los *scripts* y no hay ningún problema en instalar un cliente VPN **pptp** o de cualquier otro tipo siguiendo las instrucciones. También es posible configurar fácilmente un cliente desde Gnome a través del Gnome Network Manager para pptp:

```
sudo apt-get install network-manager-pptp pptp-linux
```

Nota

Podéis consultar los detalles de la interfaz gráfica kvpnc en su página web: http://home.gna.org/kvpnc/en/index.html.

Otra web interesante donde encontraréis información sobre VPN es la página de NetworkManager: http://projects.gnome.org/NetworkManager/admins/.

12. Configuraciones avanzadas y herramientas

Existe un conjunto de paquetes complementarios (o que sustituyen a los convencionales) y herramientas que o bien mejoran la seguridad de la máquina (recomendados en ambientes hostiles), o bien ayudan en la configuración de red (y del sistema en general) de modo más amigable.

Estos paquetes pueden ser de gran ayuda al administrador de red para evitar intrusos o usuarios locales que se exceden de sus atribuciones (generalmente, no por el usuario local, sino a través de una suplantación de identidad) o bien ayudar al usuario novel a configurar adecuadamente los servicios.

En este sentido, es necesario contemplar:

1) Configuración avanzada de TCP/IP: a través del comando *sysctl* es posible modificar los parámetros del kernel durante su ejecución o en el inicio para ajustarlos a las necesidades del sistema. Los parámetros susceptibles de modificar son los que se encuentran en el directorio */proc/sys/* y se pueden consultar con **sysctl -a**. La forma más simple de modificar estos parámetros es a través del archivo de configuración */etc/syscntl.conf*.

Después de la modificación, se debe volver a arrancar la red:

```
/etc/init.d/networking restart
```

En este apartado veremos algunas modificaciones para mejorar las prestaciones de la red (mejoras según condiciones) o la seguridad del sistema (consultar las referencias para más detalles) [Mou01]:

```
net.ipv4.icmp_echo_ignore_all = 1
```

2) No responde paquetes ICMP, como por ejemplo el comando *ping*, que podría significar un ataque DoS (*Denial-of-Service*).

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

3) Evita congestiones de red no respondiendo el *broadcast*.

```
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.lo.accept_source_route = 0
net.ipv4.conf.eth0.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
```

4) Inhibe los paquetes de *IP Source routing* que podrían representar un problema de seguridad (en todas las interfaces).

```
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.all.accept_redirects = 0
```

5) Permite rechazar un ataque DoS por paquetes SYNC que consumiría todos los recursos del sistema forzando a hacer un reboot de la máquina.

```
net.ipv4.conf.lo.accept_redirects = 0
net.ipv4.conf.eth0.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
```

6) Útil para evitar ataques con *CMP Redirect Acceptance* (estos paquetes son utilizados cuando el routing no tiene una ruta adecuada) en todas las interfaces.

```
net.ipv4.icmp_ignore_bogus_error_responses = 1
```

7) Envía alertas sobre todos los mensajes erróneos en la red.

```
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.lo.rp_filter = 1
net.ipv4.conf.eth0.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
```

8) Habilita la protección contra el *IP Spoofing* en todas las interfaces.

```
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.lo.log_martians = 1
net.ipv4.conf.eth0.log_martians = 1
net.ipv4.conf.default.log_martians = 1
```

9) Generará log sobre todos los *Spoofed Packets, Source Routed Packets y Redirect Packets*.

10) Los siguientes parámetros permitirán que el sistema pueda atender mejor y más rápidos las conexiones TCP.

```
net.ipv4.tcp_fin_timeout = 40, Por defecto, 60.
net.ipv4.tcp_keepalive_time = 3600, Por defecto, 7.200.
net.ipv4.tcp_window_scaling = 0
net.ipv4.tcp_sack = 0
net.ipv4.tcp_timestamps = 0, Por defecto, todos a 1 (habilitados).
```

11) Iptables: las últimas versiones de GNU/Linux (kernel 2.4 o superiores) incluyen nuevos mecanismos para construir filtros de paquetes llamado *netfilter* [Mou01]. Esta nueva funcionalidad es gestionada por una herramienta denominada **iptables**, que presenta mejores características que su predecesor (ipchains). Como se verá en el módulo correspondiente a seguridad, es sumamente fácil construir un *firewall* con esta herramienta para detectar y hacer frente a los ataques más comunes *scans*, DoS, IPMAC *Spoofing*, etc. Su activación pasa primero por verificar que el *kernel* es 2.4 o superior, que el mismo está configurado para dar soporte de *ipfilter* (lo cual significará que se deberá recompilar el *kernel* para activar la opción Network packet filtering [CONFIG_NETFILTER], y todas las subopciones específicas). Las reglas específicas se deben activar durante el arranque (por ejemplo, a través del */etc/init.d* y el enlace adecuado al directorio *rc* adecuado) y tiene un formato similar (consultar las referencias sobre las capacidades y la sintaxis completa) a:

```
iptables -A Type -i Interface -p protocol -s SourceIP --source-port Port -d
DestinationIP --destination-port Port -j Action
```

12) GnuPG: esta herramienta permite encriptar datos para su posterior envío (por ejemplo, correo electrónico) o almacenamiento, y también para generar firmas digitales (cumple con el estándar de la RFC2440) y no utiliza algoritmos patentados, lo cual significa más libertad en el Open Source pero pérdida de compatibilidad con otras herramientas (por ejemplo, PGP 2.0) que utilizan algoritmos como el IDEA y RSA. Para su compilación y/o instalación, seguir las indicaciones de sus autores. En primer lugar, se debe crear una par de claves (pública y privada) ejecutando como *root* el comando *gpg --gen-key* dos veces y contestando las preguntas realizadas por el mismo. Generalmente, estas claves se almacenarán en ~/*root*. Lo siguiente es exportar (por ejemplo, a una página web) la clave pública para que otros usuarios la puedan utilizar para encriptar los correos/información que sólo podrá ver el usuario que ha generado la clave pública. Para ello, habrá que utilizar *gpg --export -ao UID*, lo cual generará un archivo ASCII de la clave pública del usuario UID.

Para importar una clave pública de otro usuario, se puede usar *gpg --import filename*, y para firmar una clave (significa indicarle al sistema que se está de acuerdo en que la clave firmada es de quien dice ser), se puede utilizar *gpg*

--sign-key UID. Para verificar una clave, se puede utilizar *gpg --verify file/data* y para encriptar/desencriptar *gpg -sear UID file g*, *gpg -d file*, respectivamente [Gnu].

13) Logcheck: una de las actividades de un administrador de red es verificar diariamente (más de una vez por día) los archivos *log* para detectar posibles ataques/intrusiones o eventos que puedan dar indicios sobre estas cuestiones. Esta herramienta selecciona (de los archivos *log*) información condensada de problemas y riesgos potenciales y luego envía esta información al responsable, por ejemplo, a través de un correo. El paquete incluye utilidades para ejecutarse de modo autónomo y recordar la última entrada verificada para las subsiguientes ejecuciones. Para información sobre la configuración/instalación, podéis consultar las referencias [Log].

14) PortSentry y Tripwire: estas herramientas ayudan en las funciones del administrador de red en cuanto a seguridad se refiere. **PortSentry** permite detectar y responder a acciones de búsqueda de puertos (paso previo a un ataque o a un *spamming*) en tiempo real y tomar diversas decisiones respecto a la acción que se está llevando a cabo. **Tripwire** es una herramienta que ayudará al administrador notificando sobre posibles modificaciones y cambios en archivos para evitar posibles daños (mayores). Esta herramienta compara las diferencias entre los archivos actuales y una base de datos generada previamente para detectar cambios (inserciones y borrado), lo cual es muy útil para detectar posibles modificaciones de archivos vitales como por ejemplo, en archivos de configuración. Consultar las referencias sobre la instalación/configuración de estas herramientas [Tri].

15) Xinetd: esta herramienta mejora notablemente la eficiencia y prestaciones de inetd y tcp *-wrappers*. Una de las grandes ventajas de **xinetd** es que puede hacer frente a ataques de DoA[34] a través de mecanismos de control para los servicios basados en la identificación de direcciones del cliente, en tiempo de acceso y tiempo de conexión (*logging*). No se debe pensar que Xinetd es el más adecuado para todos los servicios (por ejemplo, FTP y SSH es mejor que se ejecuten solos como daemons), ya que muchos de ellos generan una gran sobrecarga al sistema y disponen de mecanismos de acceso seguros que no crean interrupciones en la seguridad del sistema [Xin].

(34)Del inglés *denial-of-access*.

La compilación y/o instalación es simple, sólo es necesario configurar dos archivos: */etc/xinetd.conf* (el archivo de configuración de Xinetd) y */etc/rc.d/init.d/xinetd* (el archivo de inicialización de Xinetd). El primer archivo contiene dos secciones: *defaults*, que es donde se encuentran los parámetros que se aplicarán a todos los servicios, y *service*, que serán los servicios que se pondrán en marcha a través de Xinetd. Un ejemplo típico de la configuración podría ser:

```
# xinetd.conf
# Las opciones de configuración por defecto que se aplican a todos los
```

```
# servidores pueden modificarse para cada servicio
defaults
{
   instances = 10
   log_type = FILE /var/log/service.log
   log_on_success = HOST PID
   log_on_failure = HOST RECORD
}
# El nombre del servicio debe encontrarse en /etc/services para obtener
# el puerto correcto
# Si se trata de un servidor/puerto no estándar, usa "port = X"
service ftp
{
   socket_type = stream
   protocol = tcp
   wait = no
   user = root
   server = /usr/sbin/proftpd
}
#service telnet
#{
#  socket_type = stream
#  protocol = tcp
#  wait = no
#  user = root
#  no_access = 0.0.0.0
#  only_from = 127.0.0.1
#  banner_fail = /etc/telnet_fail
#  server = /usr/sbin/in.telnetd
#}
service ssh
{
   socket_type = stream
   protocol = tcp
   wait = no
   user = root
   port = 22
   server = /usr/sbin/sshd
   server_args = -i
}
service http
{
   socket_type = stream
   protocol = tcp
   wait = no
   user = root
   server = /usr/local/apache/bin/httpd
```

```
}
#service finger
#{
#  socket_type = stream
#  protocol = tcp
#  wait = no
#  user = root
#  no_access = 0.0.0.0
#  only_from = 127.0.0.1
#  banner_fail = /etc/finger_fail
#  server = /usr/sbin/in.fingerd
#  server_args = -l
#}
# Fin de /etc/xinetd.conf
```

Los servicios comentados (#) no estarán disponibles. En la sección defaults se pueden insertar parámetros tales como el número máximo de peticiones simultáneas de un servicio, el tipo de registro (*log*) que se desea tener, desde qué nodos se recibirán peticiones por defecto, el número máximo de peticiones por IP que se atenderán, o servicios que se ejecutarán como superservidores (*imapd* o *popd*), como por ejemplo:

```
default {
instances = 20
log_type = SYSLOG
authpriv log_on_success = HOST
log_on_failure = HOST
only_from = 192.168.0.0/16
per_source = 3
enabled = imaps
}
```

La sección service, una por cada servicio, como por ejemplo:

```
service imapd {
socket_type = stream
wait = no
user = root
server = /usr/sbin/imapd
only_from = 0.0.0.0/0 #allows every client
no_access = 192.168.0.1
instances = 30
log_on_success += DURATION USERID
log_on_failure += USERID
nice = 2
```

```
redirect = 192.168.1.1 993
#Permite redireccionar el tráfico del port 993
#hacia el nodo 192.168.1.1
bind = 192.168.10.4
#Permite indicar a qué interfaz está asociado el servicio para evitar
# problemas de suplantación de servicio.
}
```

El archivo */etc/init.d/xinetd* permitirá poner en marcha el servidor (con el enlace adecuado, según el nivel de ejecución seleccionado, por ejemplo, 3, 4 y 5). Es conveniente cambiar los atributos de ambos archivos para garantizar que no son modificados o desactivados con: chmod 700 */etc/init.d/xinetd*;
chown 0.0 */etc/init.d/xconfig*; chmod 400 */etc/xinetd.conf*;
chattr +i */etc/xinetd.conf*.

16) Linuxconf: es una herramienta de configuración y administración de un sistema GNU/Linux pero que ha quedado obsoleta, si bien se puede encontrar todavía en algunas distribuciones.

Nota

Más información en Solucorp: http://www.solucorp.qc.ca/linuxconf/.

17) Webmin: es otra herramienta (paquetes webmin-core, webmin-dhcp, webmin-inetd, webmin-sshd...) que permite a través de una interfaz web (es necesario tener por ejemplo el servidor Apache instalado), configurar y añadir aspectos relacionados con la red. Si bien se continúa su desarrollo en muchas distribuciones, no se incluye por defecto. Para ejecutarla, una vez instalada desde un navegador hay que llamar a la URL https://localhost:10000, que solicitará la aceptación del certificado SSL y el usuario (inicialmente root) y su clave (passwd).

Nota

Más información en la página de Webmin: http://www.webmin.com/.

18) System-config-*: en Fedora existe una variedad de herramientas gráficas que se llaman system-config-"algunacosa" y donde "alguna-cosa" es para lo cual están diseñadas. En general, si se está en un entorno gráfico, se puede llegar a cada una de ellas por medio de un menú; sin embargo, cada una de estas herramientas implica un menú a recordar. Una herramienta que centraliza todas las *system config* es **system-config-control** en una sola entrada de menú y en una única interfaz gráfica, desde la cual se puede seleccionar de acuerdo a una organización de iconos Para ello, es necesario Applications -> Add/Remove Software, y en este se arranca como root el gestor gráfico de software Pirut (y se debe tener habilitados el repositorio Fedora Extras). En la interfaz de Pirut, se usa, por ejemplo, la búsqueda de paquetes disponibles con el patrón system-config-* haga su selección de system-config-control* y haga un clic en *Apply*. Entre otras opciones, allí se podrán configurar casi todos los aspectos de red y servicios.

19) Networkmanager: es una herramienta que permite manejar fácilmente redes inalámbricas y por cable en forma simple y sin grandes complicaciones, pero no es indicado para servidores (solo para máquinas de escritorio). Su instalación es muy fácil: **apt-get install network-manager-xx**, donde xx es

gnome o kde, según el escritorio instalado. Para configurarlo se deben comentar todas las entradas en (Debian) */etc/network/interfaces*, excepto la interfaz de loopback interface, por ejemplo dejando solo:

```
auto lo
iface lo inet loopback
```

Este paso no es obligatorio pero acelera el descubrimiento de las redes/interfaces. Sobre Debian se debe también agregar un paso extra y es que el usuario debe integrarse dentro del grupo *netdev* por una cuestión de permisos. Para hacerlo, hay que ejecutar (como *root* o si no con el comando *sudo* por delante) **adduser *usuario_actual* netdev** y hacer un *reboot* (o también reiniciar la red con */etc/init.d/networking restart* y hacer un logout-login -salir y entrar- para que el usuario actual se quede incluido en el grupo *netdev*).

20) Otras herramientas:

- **Nmap**: explorar y auditar con fines de seguridad una red.
- **Nessus**: evaluar la seguridad de una red de forma remota.
- **Wireshark** (ex-Ethereal): analizador de protocolos de red.
- **Snort**: sistema de detección de intrusos, IDS.
- **Netcat**: utilidad simple y potente para depurar y explorar una red.
- **TCPDump**: monitorización de redes y adquisición de información.
- **Hping2**: genera y envía paquetes de ICMP/UDP/TCP para analizar el funcionamiento de una red.

Ved también

Algunas de estas herramientas serán tratadas en el módulo de administración de seguridad en la asignatura *Administración avanzada de sistemas GNU-Linux*.

Actividades

1. Definid los siguientes escenarios de red:
a. Máquina aislada.
b. Pequeña red local (4 máquinas, 1 gateway).
c. 2 redes locales segmentadas (2 conjuntos de 2 máquinas, un router cada una y un gateway general).
d. 2 redes locales interconectadas (dos conjuntos de 2 máquinas + gateway cada una).
e. 2 máquinas conectadas a través de una red privada virtual. Indicar la ventajas/desventajas de cada configuración, para qué tipo de infraestructura son adecuadas y qué parámetros relevantes se necesitan.

2. Configurad la red de la opción a, b y d del punto.

Bibliografía

[Bro01] Scott Bronson. "VPN PPP-SSH". *The Linux Documentation Project*, 2001.

[Cis00] Cisco. "TCP/IP White Paper", 2000. http://www.cisco.com

[Com01] Douglas Comer. *TCP/IP Principios básicos, protocolos y arquitectura*. Prentice Hall, 2001.

[Dra99] Joshua Drake. "Linux Networking". *The Linux Documentation Project*, 1999.

[Gar98] Bdale Garbee (1998). *TCP/IP Tutorial*. N3EUA Inc.

[Gnu] Gnupg.org. *GnuPG Web Site*.http://www.gnupg.org/

[IET] IETF. "Repositorio de Request For Comment desarrollado por Internet Engineering Task Force (IETF) en el Network Information Center (NIC)".

[KD00] Olaf Kirch; Terry Dawson (2000). *Linux Network Administrator's Guide*.

O'Reilly Associates. Y como *e-book* (free) en Free Software Foundation, Inc. http://www.tldp.org/guides.html

[Law07] David Lawyer. "Linux Módem". *The Linux Documentation Project*, 2007.

[Log] LogCheck. http://logcheck.org/

[Mal96] Fred Mallett. *TCP/IP Tutorial*. FAME Computer Education, 1996.

[Mou01] Gerhard Mourani (2001). *Securing and Optimizing Linux: The Ultimate Solution*. Open Network Architecture, Inc.

[PPP00] Corwin Williams, Joshua Drake; Robert Hart (2000). "Linux PPP". *The Linux Documentation Project*.

[Ran05] David Ranch (2005). "Linux IP Masquerade" y John Tapsell. Masquerading Made Simple. *The Linux Documentation Project*.

[Rid00] Daniel López Ridruejo (2000). "The Linux Networking Overview". *The Linux Documentation Project*.

[Sec00] Andrés Seco (2000). "Diald". *The Linux Documentation Project*.

[Tri] Tripwire.com. *Tripwire Web Site*. http://www.tripwire.com/

[Vas00] Alavoor Vasudevan (2000). "Modem-Dialup-NT". *The Linux Documentation Project*.

[Wil02] Matthew D. Wilson (2002). "VPN". *The Linux Documentation Project*.

[Xin] Xinetd Web Site. http://www.xinetd.org/

Anexo

Controlando los servicios vinculados a red en FCx

Un aspecto importante de todos los servicios es cómo se ponen en marcha. Fcx (desde la FC6) incluye una serie de utilidades para gestionar los servicios -daemons- (incluidos los de red). Como ya se ha visto en el apartado de administración local, el *runlevel* es el modo de operación que especifica que daemons se ejecutarán. En FC podemos encontrar: runlevel 1 (monousuario), runlevel 2 (multiusuario), runlevel 3 (multiusuario con red), runlevel 5 (X11 más (runlevel 3). Típicamente se ejecuta el nivel 5 o 3 si no se necesitan interfaces gráficas. Para determinar qué nivel se está ejecutando, se puede utilizar */sbin/runlevel* y para saber qué nivel es el que se arranca por defecto *cat /etc/inittab* | grep :initdefault: que nos dará información como id:5:initdefault: (tambien se puede editar el */etc/inittab* para cambiar el valor por defecto).

Para visualizar los servicios que se están ejecutando, podemos utilizar /sbin/chkconfig –list y para gestionarlos podemos utilizar **system-config-services** en modo gráfico o *ntsysv* en la línea de comandos. Para habilitar servicios individuales podemos utilizar *chkconfig*. Por ejemplo, el siguiente comando habilita el servicio **crond** para los niveles 3 y 5: */sbin/chkconfig* ***--level 35 crond on.***

Independientemente de cómo se hayan puesto en marcha los servicios, se puede utilizar ***/sbin/service* –status-all** o individualmente ***/sbin/service crond status*** para saber cómo está cada servicio. Y también gestionarlo (start, stop, status, reload, restart); por ejemplo, ***service crond stop*** para pararlo o ***service crond restart*** para reiniciarlo.

Es importante **no deshabilitar los siguientes servicios** (a no ser que se sepa lo que se está haciendo): acpid, haldaemon, messagebus, klogd, network, syslogd. Los servicios más importantes vinculados a la red (aunque no se recogen todos, sí la mayoría de ellos en esta lista no exhaustiva) son:

- **NetworkManager, NetworkManagerDispatcher**: es un *daemon* que permite cambiar entre redes fácilmente (Wifi y Ethernet básicamente). Si sólo tiene una red no es necesario que se ejecute.

- **Avahi-daemon, avahi-dnsconfd**: es una implementación de *zeroconf* y es útil para detectar dispositivos y servicios sobre redes locales sin DNS (es lo mismo que *mDNS)*.

- **Bluetooth**, **hcid**, **hidd**, **sdpd**, **dund**, **pand**: Bluetooth red inalámbrica es para dispositivos portátiles (*NO ES* wifi, 802.11). Por ejemplo, teclados, mouse, teléfonos, altavoces/auriculares, etc.

- **Capi**, **isdn**: red basada en hardware *ISDN*[35].

(35)En español corresponde a RSDI (Red de Servicios Integrados).

- **Iptables**: es el servicio de *firewall* estándar de Linux. Es totalmente necesario por seguridad si se tiene conexión a red *(cable, DSL, T1)*.

- **Ip6tables**: es el servicio de *firewall* estándar de Linux pero para el protocolo y redes basadas en Ipv6.

- **Netplugd**: puede monitorizar la red y ejecutar comando cuando su estado cambie.

- **Netfs**: se utiliza para montar automáticamente sistemas de archivos a través de la red (NFS, Samba, etc.) durante el arranque.

- **Nfs**, **nfslock:** son los daemon estándar para compartir sistemas de archivos a través de la red en sistemas operativos estilo Unix/Linux/BSD.

- **Ntpd**: servidor de hora y fecha a través de la red.

- **Portmap:** es un servicio complementario para NFS (*file sharing*) y/o NIS (*authentication*).

- **Rpcgssd**, **rpcidmapd**, **rpcsvcgssd:** se utiliza para NFS v4 (nueva versión de NFS).

- **Sendmail**: este servicio permite gestionar los mails (MTA) o dar soporte a servicios como IMAP o POP3.

- **Smb**: este daemon permite compartir ficheros sobre sistemas *Windows*.

- **Sshd**: SSH permite a otros usuarios conectarse interactivamente de forma segura a la máquina local.

- **Yum-updatesd:** servicio de actualizaciones por red de FC.

- **Xinetd:** servicio alternativo de **inetd** que presenta un conjunto de características y mejoras, como por ejemplo lanzar múltiples servicios por el mismo puerto (este servicio puede no estar instalado por defecto).