Administración Linux

Departamento Sistemas Informáticos y Computación

Fernando Ferrer Garcia

Administración Linux

Departamento Sistemas Informáticos y Computación
Fernando Ferrer Garcia

Tabla de contenidos

Lista de figuras

Lista de tablas

1

Instalación de GNU/Linux

Tabla de contenidos

1.1. Introducción a GNU/Linux

En el año 1991, un estudiante de informática llamado Linus Torvalds comienza a programarse su propia versión de Unix. Linus publica en Internet no solamente los binarios, o sea los ejecutables en código máquina, sino también los fuentes correspondientes. El crecimiento de Internet hace que centenares de programadores de todo el mundo colaboren en el desarrollo de Linux: su motivación es resolver problemas individualmente, trabajando en equipo o bien, les gusta añadir nuevas funcionalidades para perfeccionar el sistema. La única condición es la de proporcionar el trabajo hecho y su código fuente otra vez a la comunidad de programadores. Este principio se manifiesta en la GNU General Public License (GPL).

Hoy en día existen muy buenas razones para optar por Linux ya que, el sistema ofrece estabilidad, seguridad y velocidad. Otro aspecto importante es su capacidad de conectividad en redes que ha sido decisiva para la conquista del mercado de servidores. Los gurús del Linux aprecian la disponibilidad del código fuente lo que proporciona al sistema operativo un alto nivel de independencia y flexibilidad. Debido a esta disponibilidad nadie está a merced de ningún productor de software sino que es posible hacer adaptaciones y extensiones según las necesidades. Tampoco hay que olvidar que el uso de Linux no exige la adquisición de licencias; da igual si se usa de forma particular o con propósitos comerciales.

Además de la gran cantidad de aplicaciones libres, cada vez hay mas aplicaciones de uso comercial para Linux. Productores de Bases de Datos como Oracle, Informix y Sybase al igual que de suites ofimáticas como Aplix, Corel o OpenOffice ofrecen sus productos para Linux.

1.1.1. ¿ Que es Linux ?

Lo que realmente se entiende bajo el término Linux es el Kernel, el núcleo del sistema operativo. Pero el kernel por sí solo no forma todavía ningún sistema operativo. Justamente para Unix existe una gran cantidad de software libre, lo que significa que estos están también disponibles para Linux. Es el conjunto de todo esto (kernel y utilidades) lo que realmente forma un sistema operativo.

En cuanto a las utilidades se trata generalmente de la versión GNU de los programas correspondientes de Unix, que la mayoría de las veces ofrecen mayor funcionalidad. El más conocido es el compilador GNU C/C++, considerado uno de los mejores compiladores del mundo. Tampoco se debe de olvidar todas aquellas utilidades que se pueden usar en la línea de comandos o en scripts: utilidades de tratamiento de textos o ficheros como sed, awk y perl (que es mucho mas que una utilidad), hasta editores como el vi o entornos de trabajo completos como Emacs.

Todo se complementa con el Servidor X Window para sistemas Unix de PC, Xfree86 (actualmente en la versión 4.3.0). Esta versión se ha portado de la distribución oficial X11R6.3 del consorcio X Consortium Inc., lo que proporciona total compatibilidad con este estándar.

Como ya se ha mencionado, existe para Unix una cantidad enorme de software libre, lo que permite a su vez componer una multitud de sistemas Linux. En este punto aparecen las distribuciones (SuSE, RedHat, Slackware, Debian, OpenLinux, Mandrake ...), las cuales contemplan la enorme oferta de software libre y eligen los programas más adecuados para distribuirlos en forma de CD o través de Internet, así que no hace falta comprar una distribución para tener un sistema GNU/Linux en nuestros servidores.

1.1.2. Razones para utilizar GNU/Linux

Una de las primeras razones que se pueden esgrimir para utilizar Linux, ya la hemos comentado: no son necesarias licencias. Aunque Linus Torlvalds mantiene la marca registrada Linux, el Kernel de Linux y la mayoría del software que le acompaña se distribuye bajo licencia GPL. Esto significa que se puede modificar el código fuente y vender los programas resultantes, pero los autores originales mantienen el copyright y tu debes proporcionar los fuentes de tus cambios.

Linux se ejecuta en más CPU's y plataformas diferentes que cualquier otro sistema operativo, ya que Linux viene con el código fuente del Kernel y es fácilmente portable. Si necesitas soporte, Linux representa una ventaja real, especialmente al comparar el coste de otros sistemas operativos. Además suele ser inmune a la cantidad de virus que afectan a otros sistemas operativos.

1.2. Instalación de un servidor linux

Toda la información que sigue a continuación referida a la instalación de un servidor GNU/Linux, hará referencia a la distribución CentOS-4 de Linux, que es la que se utilizará durante el curso. Los pasos a seguir no difieren en demasía para otras distribuciones, sobre todo en lo referente a los métodos de instalación (CD-ROM, ftp, http, NFS, disco duro). La elección de la distribución dependerá del usuario o administrador, siendo una de las mejores opciones CentOS/RedHat bajo mi personal punto de vista.

1.2.1. Requerimientos de Hardware

Para instalar Linux, primero es conveniente recoger toda la información referente al hardware disponible. La siguiente guía puede ser de gran ayuda:

1. ¿ Cuantos discos duros dispones ?
2. ¿ De que tamaño son ?
3. Si tienes mas de uno, ¿ Cual es el primario ?
4. ¿ Cuanta memoria RAM tiene tu equipo ?
5. ¿ Posee tu equipo un adaptador SCSI ?

6. ¿ Que tipo de ratón tienes y de cuantos botones ?

7. Si el ratón es serie, ¿ A que puerto está conectado ?

8. ¿ Cual es el fabricante y modelo de tu tarjeta de vídeo ?

9. ¿ Cuanta memoria posee tu tarjeta de vídeo ?

10. ¿ Que tipo de monitor tienes (fabricante y modelo) ?

11. ¿ Va a conectarse tu equipo a la red ? Entonces deberías de saber tu dirección IP, la máscara de red, la dirección del gateway, la dirección IP de tu servidor de nombres, tu nombre de dominio, tu nombre de host, tu modelo de tarjeta de red.

1.2.2. Métodos de Instalación

Puede instalar o actualizar Fedora Linux mediante varios métodos. Cada método funciona mejor en situaciones distintas, y tiene requerimientos distintos. Pero antes de describir cada método de instalación, veamos un matiz que puede afectar a algunos de ustedes.

Instalación desde CD-ROM

Si usted posee un CD-ROM con Red Hat Linux, y su ordenador tiene una unidad de CD-ROM debería considerar instalar por este procedimiento. Instalar directamente desde un CD-ROM es la aproximación más sencilla. Al instalar desde CD-ROM, los paquetes que usted seleccione serán copiados desde el CD-ROM, y serán instalados en su disco duro.

Como su propio nombre indica, necesitará un CD-ROM de Red Hat Linux, una unidad de CD-ROM soportada, y una manera de arrancar el programa de instalación.

Los sistemas Intel necesitarán usar un disquete de arranque (y el disquete con soporte PCMCIA si se usa un dispositivo PCMCIA durante la instalación). Hay un método alternativo para instalar desde CD-ROM que no usa disquetes, pero requiere que el sistema esté ejecutando DOS. El CD-ROM de Fedora Linux/Intel también puede ser el disco de arranque para los ordenadores nuevos que soporten CD-ROMs auto arrancables. No todos los ordenadores aceptan esta característica, así que si el suyo no puede arrancar desde CD-ROM, tendrá que utilizar un disquete de arranque (o aoutoboot desde DOS) para comenzar el proceso. Tenga en cuenta que puede necesitar cambiar la configuración de su BIOS para habilitar esta característica.

Instalación desde un servidor Ftp

Si no cuenta con un CD-ROM de Red Hat Linux o no posee una unidad de CD-ROM, pero tiene acceso a una red, puede que una instalación por FTP sea para usted. Cuando se instala vía FTP los paquetes de Red Hat Linux que seleccione son obtenidos (usando FTP) a través de la red, y son instalados en su disco duro.

Al realizar una instalación por FTP, necesitará acceso a una red basada en LAN; una conexión telefónica vía módem no funcionará. Si su Red de Área Local (Local Area Network) tiene acceso a Internet, puede usar uno de los muchos sitios FTP que hacen espejo de Red Hat Linux. Puede encontrar una lista de sitios en http://www.redhat.com/mirrors.html. Si su LAN carece de acceso a Internet, no todo está perdido. Si hay un ordenador en su LAN que acepte peticiones anónimas de FTP, simplemente ponga una copia de la distribución Red Hat Linux en ese sistema, y estará listo para empezar.

Su servidor de FTP debe ser capaz de manejar nombres largos de archivo. Para instalar por FTP, debe utilizar el disco de arranque específico a la instalación por red, y un disquete con soporte PCMCIA si va a usar un dispositivo PCMCIA durante la instalación. Necesitará tener configurado un servidor de nombres válido o deberá especificar la dirección IP del servidor de FTP que vaya a utilizar. También necesitará saber el path o camino del directorio de Red Hat Linux en el servidor de FTP.

Instalación desde un servidor HTTP

Para instalar por HTTP, debe utilizar el disco de arranque específico para la instalación por red, y un disquete con soporte PCMCIA si va a usar un dispositivo PCMCIA durante la instalación. Necesitará tener configurado un servidor de nombres válido o deberá especificar la dirección IP del servidor de HTTP que vaya a utilizar. También necesitará saber el path o camino del directorio de Red Hat Linux en el servidor de HTTP.

Instalación desde un servidor NFS

Para instalar por NFS, deberá montar el CD-ROM de Red Hat Linux en un ordenador que soporte el sistema de archivos ISO-9660 con extensiones Rock Ridge. La máquina también debe soportar NFS. Exporte el sistema de archivos del CD-ROM por NFS. Necesitará tener configurado un servidor de nombres válido o deberá especificar la dirección IP del servidor de NFS que vaya a utilizar. También necesitará saber el path o camino del CD-ROM exportado. Su servidor NFS debe soportar nombres largos de fichero. Para instalar por NFS, sólo necesitará un disquete de arranque.

Instalación desde disco duro

Si ninguno de los métodos de instalación le funciona, pero tiene medios para copiar los archivos de Red Hat Linux en el disco duro de su sistema, puede instalar desde su disco duro. En este método de instalación, los paquetes de Red Hat Linux que escoja serán leídos de una partición en un disco duro, y serán instalados en otra partición (o grupo de particiones).

El método de instalación desde disco duro requiere un poco de esfuerzo por adelantado de su parte, pues debe copiar todos los archivos necesarios en una partición antes de comenzar el programa de instalación de Red Hat Linux. Primero debe crear un directorio Fedora en el directorio raíz de su árbol de directorios. Todo lo que vaya a instalar debe estar colocado en ese directorio. A continuación, copie las imágenes ISO de los CD's en ese directorio. El sistema de instalación se encargará de acceder a los paquetes una vez hemos indicado la ruta a

los archivos de Fedora.

1.2.3. Creación de un disquete de arranque

Cuando no sea posible arrancar nuestra máquina directamente desde el CD-ROM, y sea el que sea el método elegido (ftp, nfs, http, cd-rom) para la instalación, deberemos crearnos un disquete de arranque desde un archivo imagen.

Un archivo imagen es un fichero que contiene una copia exacta (o imagen) del contenido de un disquete. Como el disquete contiene información del sistema de archivos, aparte de la información contenida en los ficheros, el archivo imagen no se podrá usar hasta que lo escribamos en un disquete. Para hacer esto, necesitará un disquete de 3,5 pulgadas de alta densidad (1.44 MB), y un ordenador con unidad de disquetes adecuada para este formato, capaz de ejecutar un programa DOS o la utilidad **dd**, que puede encontrar en la mayoría de los sistemas operativos del estilo de Linux.

Puede encontrar los ficheros imagen en los siguientes directorios de su CD de Red Hat Linux. Suponiendo que el CD-ROM se encuentra en la unidad D: bajo DOS, habrá que acceder al directorio d:\images.

Una vez que ha seleccionado el fichero imagen apropiado, ha llegado el momento de transferirlo a un disquete. Esto se puede hacer en un sistema DOS, o en un sistema en el que se encuentre funcionando un sistema operativo de tipo Linux.

Para preparar un disquete bajo MS-DOS, emplee la utilidad rawrite que incluímos en el CD de Red Hat Linux en el directorio dosutils. Primero etiquete un disquete formateado con el nombre adecuado. Introdúzcalo en la unidad de disquetes, y emplee las siguientes órdenes en su computadora. Asumimos que su unidad de CD es D::

```
D:> cd dosutils
D:> rawrite
Enter disk image source file name: ..\images\boot.img
Enter target diskette drive: a:
Please insert a formatted diskette into drive A: and press ENTER
```

rawrite le preguntará primero por el nombre del archivo imagen. Introduzca el nombre completo, incluyendo el directorio, del archivo que desea escribir en el disquete, por ejemplo: ..\images\boot.img. A continuación, rawrite pregunta por la unidad de disquete a donde transferir el fichero imagen. Por último, rawrite le pide que confirme que hay un disquete formateado en la unidad seleccionada. Una vez que haya pulsado Intro para confirmar, rawrite copia el fichero imagen al disquete. Si precisa preparar otro disquete, etiquételo y utilice rawrite de nuevo, indicando el achivo imagen apropiado.

Para preparar un disquete de instalación bajo Linux, u otro sistema operativo de su mismo tipo, precisa de permiso de escritura para el dispositivo asociado a la unidad de disquetes de 3.5" (/dev/fd0 bajo Linux). Primero etiquete un disquete formateado y en blanco de manera

apropiada (p.ej. «disco de arranque», «disco suplementario», etc...). Introdúzcalo en la unidad de disquetes, pero no utilice la orden mount con él. Cuando haya montado el CD, cambie al directorio que contenga el archivo imagen deseado y emplee la siguiente orden en su computadora (cambiando el nombre del archivo imagen según lo precise:

```
grooucho@fferrer$ cd /mnt/cdrom/images
grooucho@fferrer$ dd if=boot.img of=/dev/fd0
```

1.2.4. Clases de Instalación

Fedora Linux 9 define cinco clases o tipos de instalación diferentes: **Escritorio Personal**. **Estación de trabajo**. **Servidor**. **Personalizada**. **Actualización**. Los tres primeros tipos de instalación simplifican en gran medida el proceso de instalación, ya que el automáticamente se encargará de particionar el disco e instalar los paquetes apropiados,perdiendo sin embargo gran flexibilidad a la hora de configurar el sistema. Por esta razón recomendamos la opción "*Personalizada*", ya que te permite elegir que servicios quieres añadir a tu sistema y como quieres que se particione tu disco.

1.2.5. Particionando el disco duro

Se recomienda encarecidamente que antes de particionar un disco duro que contenga otras particiones (otros sistemas operativos) se haga una copia de seguridad de los datos importantes. La utilidad básica de cualquier distribución GNU/Linux utilizada para particionar discos se llama fdisk. Pero Fedora Linux 9 en su proceso de instalación utiliza un programa mas amigable llamado Disk Druid. Con Disk Druid podremos añadir nuevas particiones (Add), editar una partición existente (Edit), borrar una partición existente (Delete), o resetear la partición a su estado original (Reset).

De todas formas vamos a analizar algunas situaciones con las que nos podemos encontrar a la hora de particionar nuestro disco duro.

Linux y otros sistemas operativos

Para poder instalar CentOS-4 Linux, debe hacerle sitio en su disco duro. Este espacio en disco debe estar separado del que utilizan otros sistemas operativos que pueda tener instalados en su ordenador, como Windows, OS/2, o incluso otra versión de Linux. Esto se consigue dedicando una o más particiones a Fedora Linux.

Que las particiones de Fedora Linux vayan a compartir el disco duro con particiones usadas por otros sistemas operativos, la mayoría de las veces, no le supondrá ningún problema. Aún así, hay ciertas combinaciones de Linux con otros sistemas operativos que requieren precauciones adicionales. Hay información sobre cómo crear particiones compatibles con otros sistemas operativos en varios COMOs y Mini-COMOs (HOWTOs y Mini-HOWTOs), incluídos en el CD de Red Hat Linux en los directorios doc/HOWTO y doc/HOWTO/mini. En particular, los Mini-COMOs cuyos nombres comienzan con Linux+ son bastante útiles.

Si Fedora Linux/Intel va a coexistir en su sistema con OS/2, debe crear las particiones de su disco duro con el software de particionamiento de OS/2--de otro modo, OS/2 puede no reconocer las particiones. Durante la instalación, no cree particiones nuevas, pero establezca correctamente los tipos de partición de sus particiones Linux usando fdisk para Linux.

Particiones y puntos de montaje

Hay un área que los neófitos en Linux encuentran complicada y es la forma en que el sistema operativo Linux accede y usa las particiones. En DOS/Windows, es relativamente fácil. Si tiene más de una partición, cada una obtiene una ``letra de unidad''. Se usará dicha letra de unidad para referirse a los archivos o directorios de una partición dada.

Esto es completamente distinto a cómo Linux maneja las particiones y, a los efectos, el almacenamiento en disco en general. La diferencia principal es que cada partición se integra en el sistema de almacenamiento necesario para formar parte de un sólo juego de archivos y directorios. Esto se consigue asociando una partición con un directorio mediante un proceso conocido como montaje. Montar una partición significa disponer de su capacidad de almacenamiento comenzando en el directorio especificado (conocido como punto de montaje).

Por ejemplo, si la partición /dev/hda5 estuviera montada en /usr, significaría que todos los archivos y directorios bajo /usr estarían físicamente alojados en /dev/hda5. Por lo tanto, el archivo */usr/doc/FAQ/txt/Linux-FAQ* estaría almacenado en /dev/hda5, mientras que el archivo */etc/X11/gdm/Sessions/Gnome* no lo estaría. Continuando con nuestro ejemplo, también es posible que uno o más directorios bajo */usr* fueran puntos de montaje para otras particiones. Como ejemplo, una partición (digamos /dev/hda7) estaría montada en */usr/local*, queriendo decir que, por ejemplo, */usr/local/man/whatis* residiría en /dev/hda7 y no en /dev/hda5.

Número de particiones

En este punto del proceso de preparación para instalar Fedora Linux, necesitará considerar el número y el tamaño de las particiones que utilizará su nuevo sistema operativo. Se recomien-

da, a no ser que tenga una razón para no hacerlo, crear las siguientes particiones como mínimo.

Una partición de intercambio (*swap*). Las particiones de intercambio se usan como apoyo a la memoria virtual. Si su ordenador tiene 16 MB de RAM o menos, debería crear una partición para el intercambio. Incluso teniendo suficiente memoria, se sigue recomendando tener una partición swap. El tamaño mínimo debería ser igual a la RAM presente en su ordenador.

Una partición /boot. La partición montada en /boot contiene el kernel del sistema operativo, así como los archivos usados durante el arranque. Debido a las limitaciones de la mayoría de las BIOS de los PCs, no es mala idea crear una pequeña partición para alojar estos archivos. Esta partición no debería ser mayor de 16MB.

La partición raíz o partición root. La partición raíz es donde reside / (el directorio raíz). En este perfil de particiones, todos los archivos (excepto los alojados en /boot) se encuentran en la partición raíz. Por ello, interesa maximizar el tamaño de la partición raíz. Una partición raíz de unos 1500 MB le proporcionará el equivalente a una instalación de tipo workstation (con muy poco espacio libre, mientras que una partición raíz de 4 GB le permitirá instalar todos los paquetes.

De todas formas, es posible crear una estructura de particiones diferentes para adecuarla a las funciones que realice nuestro servidor. No sería mala idea colocar los directorios /tmp y /home en particiones separadas de la partición raíz, ya que si los usuarios van a acceder al servidor, esta división prevendrá que estos puedan llenar cualquier sistema de ficheros crítico. Tampoco sería mala idea colocar /var y /usr en particiones separadas, por las mismas razones esgrimidas anteriormente.

Por último comentar que será a través de la herramienta **Disk Druid**, donde podremos definir el número y el tipo de las particiones que requerirá nuestro sistema. Hay que comentar que el tipo de las particiones que utiliza Linux es el ext3 (por lo menos la particiones del sistema deberán de ser de este tipo), lo cual no le impide que pueda ser capaz de leer o crear otro tipo de particiones.

Elección del gestor de arranque

Para poder arrancar el sistema sin la necesidad de un disquete de boot, normalmente se utiliza un cargador de sistemas operativos. Este cargador es un software que se ejecuta cuando la máquina arranca y es el responsable de cargar y transferir el control al kernel. El kernel a su vez, inicializa el resto del sistema operativo. El proceso de instalación de Fedora, proporciona dos tipos de cargadores a elegir, GRUB y LILO.

GRUB (GRand Unified Bootloader), el cargador por defecto, es el mas poderoso. Puede cargar una gran variedad de sistemas operativos libres, así como sistemas operativos propietarios utilizando la técnica de *chain-loading*.

LILO (LInux LOader) es también un cargador para linux muy eficaz. No depende de un sistema de ficheros específico y puede arrancar/cargar imágenes del kernel linux desde disquete o disco duro, así como otros sistemas operativos.

1.2.6. Instalación de paquetes

Después de configurar las particiones y seleccionarlas para formatearlas, se está en disposición de seleccionar los paquetes para su instalación. Puede seleccionar componentes, que agrupan paquetes por su función, paquetes individuales, o una combinación de ambos.

Los componentes agrupan paquetes según la funcionalidad que proporcionan. Por ejemplo, Desarrollo C [C Development], Estación de Trabajo en Red [Networked Workstation], o Servidor Web [Web Server]. Seleccione cada componente que desee instalar y presione Espacio. Si selecciona Todo [Everything] (puede ser encontrado al final de la lista de componentes) se instalan todos los paquetes incluidos en Red Hat Linux. Si selecciona todos los paquetes, necesitará cerca de 1Gb de espacio de disco libre.

Después de seleccionar los componentes que desea instalar, puede querer seleccionar o deseleccionar paquetes individuales. El programa de instalación presenta una lista de los grupos de paquetes disponibles; utilizando las flechas, seleccione un grupo para examinar, y presione Intro o Espacio. El programa de instalación presenta una lista de los paquetes de ese grupo, que debe seleccionar o deseleccionar utilizando las flechas para resaltar un paquete, y presionando Espacio. Algunos paquetes (tales como el núcleo y ciertas librerías) son necesarios en todos los sistemas Red Hat Linux y no están disponibles para ser seleccionados o deseleccionados.

Muchos de los paquetes software, para trabajar correctamente, dependerán de otros paquetes software, o librerías que deben ser instaladas en su sistema. Por ejemplo, muchas de las herramientas gráficas de administración de sistema de Red Hat requieren los paquetes python y pythonlib. Para asegurar que su sistema tenga todos los paquetes que necesite para ser completamente funcional, Red Hat Linux comprueba las dependencias de estos paquetes cada vez que instala o elimina paquetes software. Después de que haya acabado de seleccionar paquetes para instalar, el programa de instalación comprueba la lista de dependencias de los paquetes seleccionados. Si cualquier paquete necesita otro paquete que no ha seleccionado para instalar, el programa presenta una lista de estas dependencias sin resolver y le da la oportunidad de resolverlas. Si simplemente presiona Aceptar [Ok], el programa las resolverá automáticamente añadiendo todos los paquetes requeridos por la lista de paquetes seleccionados.

Después de haber resuelto todas las dependencias de los paquetes, el programa de instalación presenta un cuadro de diálogo indicándonos que se va a escribir el fichero / tmp/install.log con un registro de todos los paquetes instalados en su Red Hat Linux. Seleccione la opción Aceptar [Ok] y presione Espacio para continuar. En este punto, el programa de instalación formateará todas las particiones que haya seleccionado para formatear. Este proceso puede llevar varios minutos, (e incluso será más largo si le indicó al programa de instalación que comprobara los bloques dañados). Una vez formateadas las particiones, el programa de instalación empieza a instalar paquetes.

2

Configuración Post-Instalación de GNU/Linux

Tabla de contenidos

2.1. Configuración del Proceso de arranque

Una de las características más importantes de Linux es el método altamente configurable que se utiliza para el inicio del sistema operativo. El administrador es libre de configurar muchos aspectos del proceso de arranque, incluyendo qué programas se lanzarán en el momento del arranque. De forma parecida, la parada del sistema finaliza los procesos de forma organizada y configurable, aunque la personalización de este proceso casi nunca es necesaria. Entender el funcionamiento del proceso de arranque y parada no solo le permite personalizarlo, sino que también facilita resolver problemas relacionados con el inicio y el cierre del sistema.

2.1.1. La Secuencia de Arranque

Cuando un ordenador arranca, el procesador busca al final de la memoria del sistema el programa de la **BIOS** (*Basic Input/Output System*) y lo ejecuta. La BIOS controla no sólo el primer paso del proceso de arranque, sino que también proporciona una interfaz de bajo nivel para dispositivos periféricos.

Una vez que se haya cargado, la BIOS chequea los periféricos y localiza un dispositivo con el que arrancar el sistema. Habitualmente, en primer lugar comprueba cualquier disquete y unidades de CD-ROM presente por los medios de arranque, y a continuación si esto falla, echa un vistazo a las unidades de disco duro del sistema. El orden de las unidades necesario para arrancar puede ser controlado por la BIOS. La BIOS carga en memoria cualquier programa que resida en el primer sector de este dispositivo, llamado **Master Boot Record** o *MBR*. El MBR sólo tiene 512 bytes de tamaño y contiene las instrucciones de código de máquina para el arranque del equipo, llamado gestor de arranque así como también la tabla de particiones. Una vez que la BIOS haya encontrado y cargado el gestor de arranque en memoria, le deja el control del proceso de arranque a éste.

Los gestores de arranque de Linux para la plataforma x86 se dividen en dos etapas. La primera es un pequeño código binario que se encuentra en el MBR. Su única función es la de localizar el gestor de arranque de la segunda etapa y cargar la primera parte de éste en memoria. GRUB es uno de los gestores de arranque más modernos, siendo capaz de leer particiones casi de cualquier tipo, pudiendo cargar su archivo de configuración — **/boot/grub/grub.conf** — en el momento de arranque desde cualquiera de ellas

Una vez que la segunda etapa del gestor de arranque está en memoria, presenta al usuario una pantalla gráfica mostrando los diferentes sistemas operativos o kernels que puede arrancar. En esta pantalla el usuario puede usar las flechas direccionales para escoger el sistema operativo o kernel con el que desea arrancar y presionar la tecla [**Intro**]. Si no se presiona ninguna tecla, el gestor de arranque carga la entrada predeterminada después de un período de tiempo de espera (también configurable).

Una vez que el gestor de arranque de la segunda etapa haya determinado qué kernel arrancar, localizará el binario del kernel correspondiente en el directorio **/boot/**. La llamada al kernel sigue el siguiente formato — */boot/vmlinuz-<kernel-version>* (donde

<*kernel-version*> corresponde a la versión del kernel especificada en las configuraciones del gestor de arranque). El gestor de arranque cargará una imagen inicial de RAM apropiada (initial RAM disk), conocida como **initrd**, en la memoria. El initrd es usado por el kernel para cargar controladores necesarios para arrancar el sistema. Esto es muy importante si posee unidades de disco duro SCSI o si está usando el sistema de ficheros ext3

Cuando el kernel se carga, inmediatamente se inicializa y configura la memoria del ordenador y los diferentes dispositivos hardware conectados al sistema, incluyendo procesadores, subsistemas de entrada/salida y dispositivos de almacenamiento. A continuación buscará la imagen initrd en una ubicación predeterminada en memoria, la descomprimirá, la montará y cargará todos los controladores necesarios. A continuación inicializa los dispositivos virtuales relacionados con el sistema de ficheros, tal como LVM o software RAID antes de desmontar la imagen del disco initrd y liberar toda la memoria que la imagen del disco ocupó anteriormente. El kernel montará la **partición raíz** (*root*) como sólo lectura y liberará cualquier memoria no utilizada. Llegados a este punto, el kernel está cargado en memoria y operativo. Sin embargo, pocas cosas interesantes se pueden hacer, ya que no existe una forma de interactuar con el kernel. Para configurar el entorno de usuario que interactue con el kernel, este inicia el programa **/sbin/init**(también llamado init) que coordina el resto del proceso de arranque y configura el ambiente del usuario..

Cuando el comando **init** arranca, se convierte en el proceso padre de todos los procesos que comienzan automáticamente en el sistema. Init lee el fichero **/etc/inittab** que describe cómo el sistema debería configurarse en cada *nivel de ejecución* (ver sección Sección 2.1.2, “Niveles de ejecución”). Básicamente, antes de establecer el nivel de ejecución, init ejecuta el script */etc/rc.d/rc.sysinit*, que establece la variable PATH, activa el swap, controla los sistemas de fichero y se encarga de todo lo que el sistema necesita tener hecho al momento de la inicialización. A continuación procede a ejecutar todos los servicios que estén definidos en el nivel de ejecución predeterminado (ver sección Sección 2.1.2, “Niveles de ejecución”)

2.1.2. Niveles de ejecución

Los niveles de ejecución son un estado, o modo, en el que entra el sistema en el proceso de arranque y que define los servicios que serán arrancados por la máquina. Linux está programado para ejecutarse en un determinado nivel de ejecución. El número de niveles y sus nombres están predeterminados. En cambio, las acciones a realizar en cada nivel son configurables por el superusuario tal como se explica más tarde en este documento.

La configuración de niveles en CentOS Linux se presenta en la siguiente tabla:

Tabla 2.1. Niveles de ejecución

0	Halt	Este nivel detiene el sistema
1	Single User	Modo de administración. El sistema crea un shell con los privilegios del superusuario sin solicitar nombre de usuario o contraseña.
2	Multiuser	Modo de funcionamiento normal sin algunos servicios de red.
3	Multiuser + network	Como el modo 2 pero con todos los servicios de red activos, NFS por ejemplo.
4		Generalmente no utilizado
5	Modo gráfico multiusuario completo	Con una pantalla de inicio de sesión basada en X
6	Reboot	Se reinicia el sistema.
s,S	Emergency single user	Igual al nivel 1 pero sin acceder a los ficheros de configuración de inicio.

Bajo esta perspectiva, un sistema Linux no se arranca o detiene, sino que simplemente se cambia su nivel de ejecución. Algunas consideraciones importantes sobre los niveles son:

- Durante un arranque normal, el sistema se coloca en el nivel 3 (multiusuario con red) o en el nivel 5 (análogo al 3 pero con el sistema de ventanas activo desde el inicio).
- **shutdown -h now** cambia el nivel actual al nivel 0 (halt).
- **shutdown -r now** cambia el nivel actual al nivel 6 (reboot).
- **/sbin/init nivel** cambia al nivel especificado
- **/sbin/runlevel** indica el *nivel* de ejecución previo y el actual.
- Desde el cargador puede expresarse el nivel de ejecución deseado pasándole como parámetro al kernel el nivel de ejecución.

El nivel de ejecución por defecto para el sistema está definido en el fichero `/etc/inittab`. Para saber el nivel de ejecución por defecto de un sistema, busque una línea similar a la que se muestra abajo cerca de la parte superior de `/etc/inittab`:

```
id:5:initdefault:
```

El programa **init** inicia todas las entradas de `/etc/inittab` que se correspondan con el nivel de ejecución por defecto. Un listado con las entradas más relevantes que se ejecutarán en el nivel 5 se muestra a continuación:

```
l5:5:wait:/etc/rc.d/rc 5

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1 2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3 4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5 6:2345:respawn:/sbin/mingetty tty6
```

Cuando se arranca el nivel de ejecución 5, el programa **init** ejecuta el script **/etc/rc.d/rc 5**. Este script consulta el directorio `/etc/rc.d/rc5.d/` para determinar qué procesos iniciar o parar.

De forma general, existirá un directorio /etc/rc.d/rc<x>.d/ , por cada nivel de ejecución definido por el sistema, donde se encuentran los servicios que deberán ser lanzados y parados en ese nivel de ejecución.

Realmente, **/etc/rc.d/rc** cuando entra en un determinado nivel de ejecución realiza las siguientes acciones:

1. Ejecuta, por orden de nombre, todos los scripts que comienzan por **K** en el directorio correspondiente al nivel, utilizando como argumento para dicho script la opción **stop**.

2. Ejecuta, por orden de nombre, todos los scripts que comienzan por **S** en el directorio correspondiente al nivel, utilizando como argumento para dicho script la opción **start**.

A título de ejemplo, a continuación se muestra un listado del directorio que corresponde al nivel multiusuario con red (`/etc/rc.d/rc3.d`).

```
ls -l rc3.d/

total 0
lrwxrwxrwx   1 root root    13 Apr 1 1998 K15gpm -> ../init.d/gpm
lrwxrwxrwx   1 root root    13 Apr 1 1998 K60lpd -> ../init.d/lpd
lrwxrwxrwx   1 root root    15 Apr 1 1998 K95nfsfs -> ../init.d/nfsfs
lrwxrwxrwx   1 root root    17 Apr 1 1998 S01kerneld -> ../init.d/kerneld
lrwxrwxrwx   1 root root    17 Apr 1 1998 S10network -> ../init.d/network
lrwxrwxrwx   1 root root    16 Apr 1 1998 S20random -> ../init.d/random
lrwxrwxrwx   1 root root    16 Apr 1 1998 S30syslog -> ../init.d/syslog
lrwxrwxrwx   1 root root    13 Apr 1 1998 S40atd -> ../init.d/atd
lrwxrwxrwx   1 root root    15 Apr 1 1998 S40crond -> ../init.d/crond
lrwxrwxrwx   1 root root    18 Apr 1 1998 S75keytable -> ../init.d/keytable
lrwxrwxrwx   1 root root    11 Apr 1 1998 S99local -> ../rc.local
```

Como puede apreciar, ninguno de los scripts que inician y apagan los servicios están localizados en el directorio `/etc/rc.d/rc3.d/`. Casi todos los ficheros en `/`

`etc/rc.d/rc3.d/` son enlaces simbólicos apuntando a los scripts localizados en el directorio `/etc/rc.d/init.d/`. Los enlaces simbólicos se usan en cada uno de los directorios `rc` de manera que los niveles de ejecución puedan ser reconfigurados al crear, modificar y eliminar los enlaces simbólicos sin que afecte a los scripts actuales a los que se refiere.

El nombre de cada enlace simbólico empieza con K o S. Como ya habíamos comentado, los scripts que empiezan por K son procesos candidatos a ser parados en ese nivel de ejecución, mientras que aquellos que empiezan por S son procesos candidatos a ser iniciados.

El administrador puede configurar las acciones que deben realizarse al entrar en un determinado nivel de ejecución. A modo de resumen, los directorios y ficheros relevantes para configurar el proceso de arranque se detallan a continuación:

/etc/inittab	Fichero base de configuración del arranque de la máquina.
/etc/rc.d	En él residen todos los scripts de inicialización.
/etc/rc.d/rc.sysinit	Script de inicialización del ordenador, independiente del nivel.
/etc/rc.d/rc<x>.d	Existe un directorio por cada nivel de ejecución, que contiene *enlaces simbólicos* a los scripts que configuran la entrada a este nivel.
/etc/rc.d/init.d	Aquí residen todos los scripts reales que pueden ser ejecutados cuando se entra en un nivel de ejecución.

Hay que tener en consideración que los scripts que residen en el directorio `/etc/rc.d/init.d` pueden utilizarse directamente, lo que permite iniciar o detener servicios de forma manual. Por ejemplo, los siguientes mandatos detienen el subsistema de red y lo vuelven a iniciar:

```
# /etc/rc.d/init.d/network stop
# /etc/rc.d/init.d/network start
```

2.1.3. Configuración de los niveles de ejecución

Como ya se ha dicho, el administrador tiene la potestad de variar el proceso de arranque de un sistema Linux, bien simplemente cambiando el nivel de ejecución al editar el fichero `/etc/inittab` o pasándole un parámetro al kernel indicando el nivel de ejecución deseado.

El sistema Linux, según la distribución elegida, vendrá con una configuración predeterminada de servicios que se deben lanzar en el proceso de arranque del sistema. De nuevo el administrador puede variar ese comportamiento. Si hemos seguido con atención la sección anterior, la forma más directa de hacer que un determinado servicio no se lance en un nivel de ejecución, sería borrar el enlace simbólico que exista en el directorio predeterminado del nivel de ejecución (`/etc/rc.d/rc<x>.d`). Si queremos volver a arrancar en el proceso de

inicio el servicio, crearemos el enlace de nuevo y listo.

Si por el contrario, nuestras necesidades pasan por añadir al proceso de arranque un nuevo servicio, los pasos necesarios para integrarlo serían los siguientes:

1. Crear un script en el directorio /etc/rc.d/init.d, cuyo esqueleto sea el siguiente:

```
#! /bin/bash
#
# miservicio  Start/Stop miservicio.
#
# chkconfig: 2345 90 60
# description:
# Source function library.
. /etc/init.d/functions
prog=/usr/sbin/miservicio
start() {
     echo -n "Iniciando $prog:"
     daemon miservicio
     RETVAL=$?
     echo [ $RETVAL -eq 0 ] && touch /var/lock/subsys/miservicio
     return $RETVAL
}
stop() {
    echo -n "Parando $prog: "
    killproc miservicio
    RETVAL=$?
    echo [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/miservicio
    return $RETVAL
}
case "$1" in
   start) start
          ;;
   stop) stop
          ;;
esac
```

2. Crear los enlaces simbólicos necesarios para parar y arrancar el servicio en el directorio que represente el nivel de ejecución predeterminado:

```
# cd /etc/rc.d/rc5.d
# ln -s /etc/rc.d/init.d/miservicio S90miservico
```

Para facilitar la tarea al administrador, CentOS posee un par de herramientas que ayudan en todo este proceso.

chkconfig

El comando **chkconfig** permite añadir y eliminar servicios en los niveles de ejecución, así como consultar la configuración de cada servicio. La sintaxis de este mandato es la siguiente:

```
chkconfig --list [name] chkconfig [--level levels] name <on|off|reset>
```

Utilizado con la opción *--list*, este mandato visualiza la configuración de todos los servicios o de un nivel concreto. Las acciones *on* y *off* activan y desactivan respectivamente un servicio en los niveles especificados. La acción *reset* reestablece los valores predeterminados para este servicio.

redhat-config-services

Desde la versión RedHat 8, el sistema incorpora una serie de utilidades gráficas para poder configurar las diferentes opciones del sistema. En CentOS, las herramientas que aparecen bajo el nombre **redhat-config-<servicio>** son las encargadas de la configuración.

El comando redhat-config-services permite al administrador modificar los servicios que se lanzarán en los diferentes niveles de ejucución.

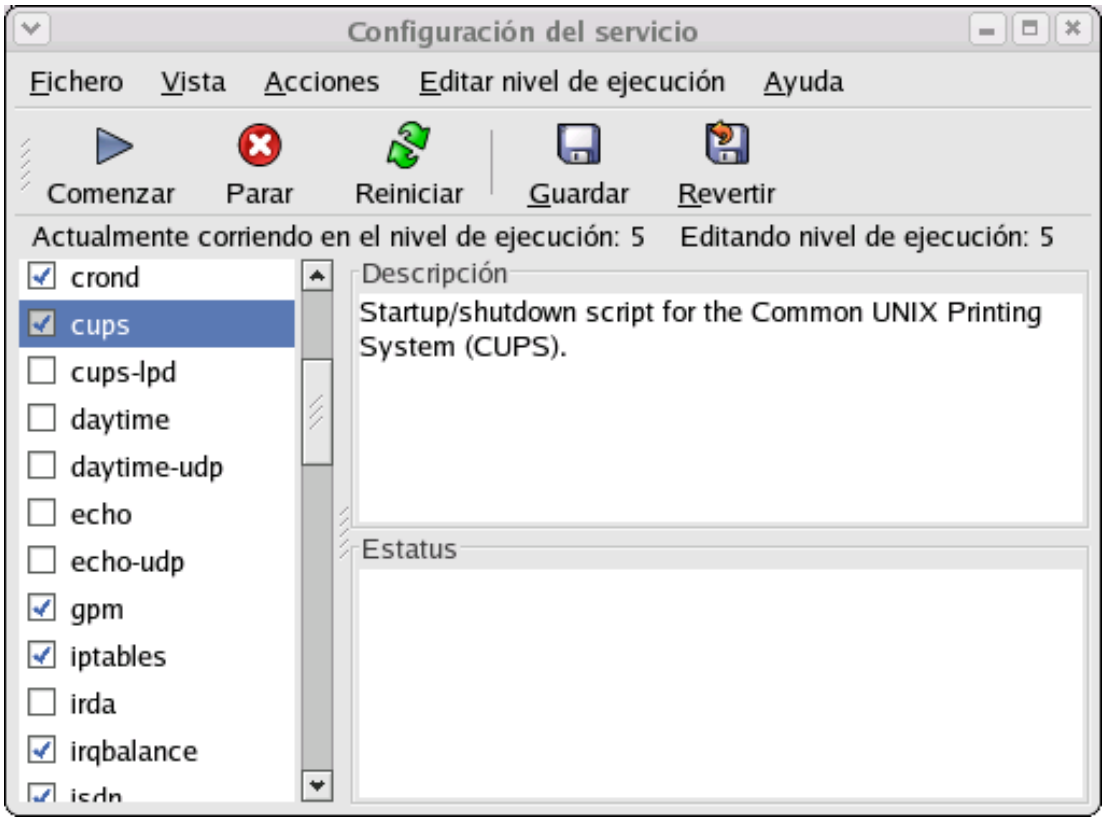

Figura 2.1. Herramienta redhat-config-services

Esta herramienta nos permite Arrancar, Parar y Reiniciar un servicio, así como definir en que niveles se ejecutará por defecto.

ntsysv

La utilidad basada en la librería ncurses **/usr/sbin/ntsysv** proporciona una interfaz interactiva en modo texto, más fácil de usar que chkconfig. No es tan versátil pero permite definir que servicios serán arrancados en el nivel de ejecución por defecto.

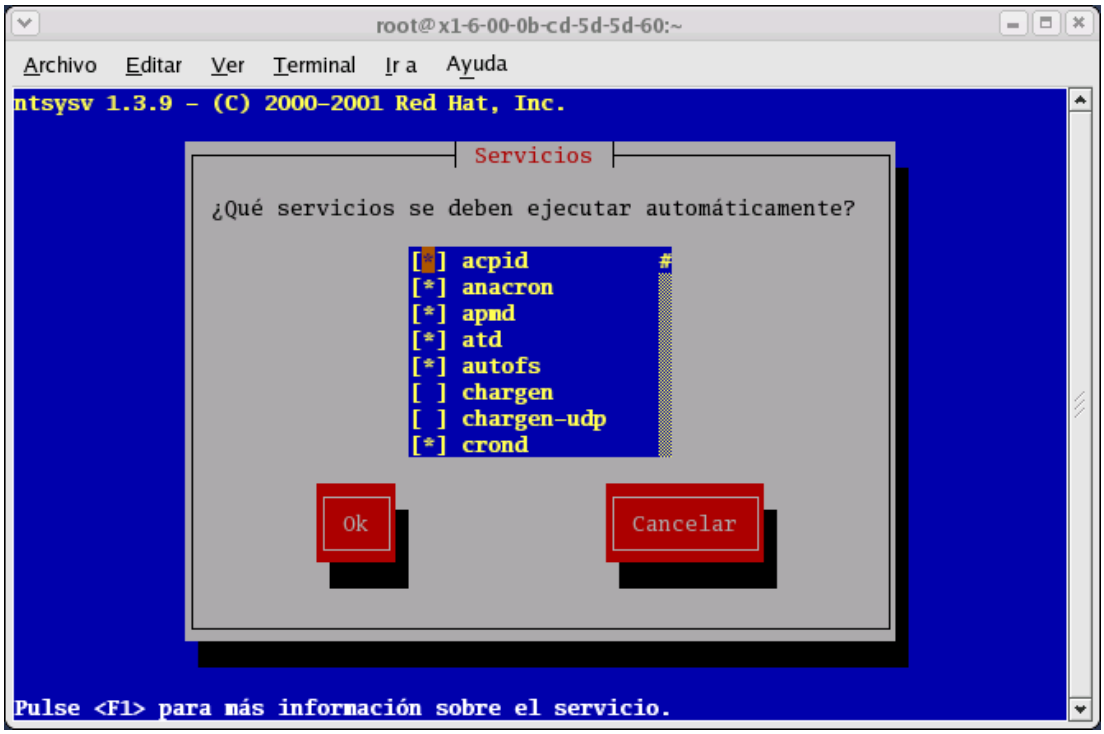

Figura 2.2. Herramienta ntsysv

2.1.4. Configuración del gestor de arranque GRUB

GNU *GRand Unified Boot loader* o GRUB es un programa que permite al usuario seleccionar qué sistema operativo instalado deseamos arrancar en el momento de arranque del sistema. Permite también que el usuario pase argumentos al kernel.

GRUB posee una serie de características que lo convierten en el método favorito respecto al resto de gestores de arranque disponibles para la arquitectura x86. A continuación exponemos una lista de las características más importantes:

1. GRUB proporciona un entorno verdadero basado en comandos, lo cual supone disponer de un pre-sistema operativo en el momento del arranque. Esto proporciona la máxima flexibilidad en la carga de los sistemas operativos que admitan determinadas opciones.

2. GRUB soporta el modo Direccionamiento Lógico de Bloques (LBA). El modo LBA permite la conversión de direccionamiento utilizada para buscar archivos en la unidad de disco duro del firmware y se utiliza en muchos discos IDE y en todos los discos duros SCSI. Antes de LBA, los gestores de arranque encontraban la limitación del cilindro 1024 de la BIOS, donde la BIOS no podía encontrar un archivo después de ese cabezal de cilindro del disco. El soporte LBA permite que GRUB arranque los sistemas operativos desde las particiones más allá del límite de 1024 cilindros, siempre y cuando la BIOS del sistema soporte el modo LBA

3. GRUB puede leer casi todo tipo de particiones. Esto permite que GRUB acceda a su archivo de configuración, `/boot/grub/grub.conf`, cada vez que el sistema arranca, eliminando la necesidad que tiene el usuario de escribir una nueva versión de la primera etapa del gestor de arranque al MBR en caso de que se produzcan cambios de la configuración. El único caso en el que el usuario necesitaría reinstalar GRUB en el MBR es en caso de que la localización física de la partición `/boot/` se traslade en el disco.

Instalación de GRUB

Si decidió no instalar GRUB durante el proceso de instalación, se puede hacer después. Una vez instalado, se convierte en el gestor de arranque por defecto. Para instalar el paquete que contiene GRUB en *CentOS*, ejecute el siguiente comando:

```
# yum install grub
```

Una vez que el paquete GRUB esté instalado, abra un intérprete de comandos de la shell y ejecute el comando **/sbin/grub-install <location>**, donde <location> es la ubicación en la que la Etapa 1 de GRUB debería ser instalado. Por ejemplo, el comando siguiente instala GRUB al MBR del dispositivo maestro IDE en el bus IDE primario:

```
# /sbin/grub-install /dev/hda
```

La próxima vez que arranque el sistema, el menú del gestor de arranque gráfico de GRUB aparecerá antes de que el kernel se cargue en memoria.

La Interfaz de usuario de GRUB

GRUB posee una interfaz de menú que permite escoger entradas que han sido definidas en el fichero de configuración de GRUB, así como acceder a una línea de comando muy flexible para ejecutar las acciones de arranque que deseemos.

GRUB busca su fichero de configuración (`/boot/grub/grub.conf`) tan pronto es cargado. Si lo encuentra, la interfaz de menú se activa, utilizando las entradas encontradas en el fichero. Si se elije la opción de menú línea de comandos o no se encuentra el fichero de configuración, entonces GRUB entra la interfaz de línea de comandos.

Interfaz de línea de comandos

La interfaz de línea de comandos proporciona al usuario un `prompt` parecido a una shell de UNIX o DOS. Cada comando introducido aquí es ejecutado inmediatamente después de presionar la tecla [**Enter**].

Los comandos disponibles en esta sección son un subconjunto de los que pueden aparecer en el fichero de configuración de GRUB. A continuación se muestran los más importantes:

- **boot**. Arranca el sistema operativo o gestor de encadenamiento que se ha cargado.
- **chainloader** `</path/to/file>`. Carga el archivo especificado como gestor de encadenamiento. Si el archivo está ubicado en el primer sector de la partición especificada, puede utilizar la notación de lista de bloques, +1, en vez del nombre del archivo.
- **initrd** `</path/to/initrd>`. Le permite especificar un disco RAM inicial para utilizarlo al arrancar. Es necesario un initrd cuando el kernel necesita ciertos módulos para poder arrancar adecuadamente.
- **kernel** `</path/to/kernel>`<opcion-1> <opcion-N> ... Especifica el archivo del kernel a cargar cuando se cargue el sistema operativo. Se sustituye </path/to/kernel> con una ruta absoluta desde la partición especificada por el comando root. Reemplace <opcion-1> con las opciones para el kernel de Linux, tales como root=/dev/hda5 para especificar el dispositivo en el que se ubica la partición root para el sistema.
- **root** (<device-type><device-number>,<partition>). Configura la partición raíz para GRUB, tal como (hd0,0) y monta la partición.
- **rootnoverify** (<device-type><device-number>,<partition>). Configura la partición raíz para GRUB, tal como el comando root pero no monta la partición.
- **makeactive**. Define la actual partición raíz (configurada con el comando root[noverify]) como la partición activa.
- **hide** <partition>. Oculta la partición especificada por la opción <partition>. Este comando es útil cuando se pretende arrancar un sistema operativo como Windows donde existen múltiples particiones FAT o NTFS en el mismo disco.

Interfaz de menú de GRUB

Esta es la interfaz por defecto cuando se configura GRUB desde el programa de instalación. En esta interfaz hay un menú de sistemas operativos o kernels preconfigurados en forma de lista ordenada por nombre. Puede utilizar las teclas de flecha para seleccionar una opción en lugar de la selección por defecto y pulsar la tecla [**Enter**] para arrancar el sistema.

El archivo de configuración de la interfaz de menú de GRUB es `/boot/grub/grub.conf`. Los comandos para configurar las preferencias globales para la interfaz de menú están ubicados al inicio del archivo, seguido de las diferentes entradas para cada sistema operativo o kernels listados en el menú.

El siguiente es un ejemplo de archivo de configuración de menú de GRUB muy básico diseñado para arrancar bien Fedora Core Linux o Microsoft Windows 2000:

```
default=1
timeout=10
splashimage=(hd0,5)/boot/grub/splash.xpm.gz
password --md5 $1$lcF0V/$zaN1LNyAr5TA6NG/4KP1N/
title CentOS
        root (hd0,5)
        kernel /boot/vmlinuz ro root=/dev/hda6
        initrd /boot/initrd.img
title Windows 2000
     unhide (hd0,0)
     hide (hd0,1)
     hide (hd0,2)
     rootnoverify        (hd0,0)
     makeactive
     chainloader +1
```

Los siguientes comandos son exclusivos de la interfaz de menú:

- **default**=<valor>. Entrada que será ejecutada por defecto sino hay intervención del usuario.
- **timeout**=<valor>. Tiempo de espera sino hay intervención del usuario.
- **splashimage**=`<path-to-image>` — Especifica la ubicación de la imagen de pantalla *splash* que se utilizará al arrancar.
- **password**=<contraseña>. Será necesario conocer la contraseña si queremos modificar las opciones de las diferentes entradas del menú.

2.2. Configuración del Sistema

Una vez configurado el proceso de arranque de GNU/Linux - nivel de ejecución predeterminado, servicios por defecto, gestor de arranque - el administrador de sistemas debe de prestar atención a otros aspectos más particulares de la máquina. Dos tareas esenciales que preparar una vez el sistema ha sido arrancado son la configuración de la red y el entorno gráfico.

2.2.1. /etc/sysconfig

Existen algunos ficheros de configuración específicos de Fedora Core 1 en el directorio `/`

`etc/sysconfig/`. Este directorio almacena una variedad información de la configuración. Muchos scripts que se ejecutan al iniciar el sistema, usan los archivos de este directorio.

Por ejemplo, la red se configura a base de ejecutar scripts que se encuentran en este directorio. El sentido básico de este directorio es mantener información de configuración que leen los diferentes servicios del sistema antes de ejecutarse.

2.2.2. Configuración de Red

Si hemos instalado Linux a través de la red, ya sea vía NFS o FTP, seguramente hemos configurado la información necesaria para el protocolo TCP/IP que es el que utiliza Linux por defecto para formar parte de una red.

Básicamente la información que se necesita para configurar una tarjeta de red adecuadamente y que tengamos acceso a la red es una dirección IP y una máscara de red. Luego deberíamos saber cual es el servidor DNS (que resolverá nombres en direcciones IP) y si queremos tener acceso a Internet, el nombre del gateway o pasarela.

Ifconfig es la utilidad que permite configurar manualmente nuestra tarjeta de red. Pero hay que tener en mente que cuando se configura nuestro dispositivo de red manualmente, estas propiedades no permanecerán ante un reinicio del sistema, por tanto habrá que habilitar algún mecanismo para que esto no sea un problema (estos y otros aspectos se verán en el punto Arranque y Parada del sistema).

Para asignar al interfaz de red (representada para el sistema como el dispositivo eth0) la dirección IP 158.42.48.111 usaremos el siguiente comando:

```
[root@mis01]# ifconfig eth0 158.42.48.111 netmask 255.255.0.0
```

Para ver todas las tarjetas que tenemos y la información relativa a ellas podremos usar el comando:

```
[root@mis01]# ifconfig

eth0      Link encap:Ethernet  HWaddr 00:60:08:65:33:97
          inet addr:158.42.48.111  Bcast:158.42.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:15700406 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17629 errors:0 dropped:0 overruns:0 carrier:0
          collisions:495 txqueuelen:100
          Interrupt:9 Base address:0xb800

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:53 errors:0 dropped:0 overruns:0 frame:0
          TX packets:53 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

Para asignar la pasarela por defecto:

```
[root@mis01]# route add default gw 158.42.1.10
```

Como se puede observar, en Fedora Core Linux, todas las comunicaciones de red se producen entre interfaces de software configuradas y dispositivos de red físicos conectados al sistema. Si queremos que nuestra configuración de red se mantenga en cada arranque habrá que definir la información de TCP/IP en los diferentes ficheros que utiliza Fedora Core para levantar la red.

Ficheros de configuración de red

Los principales ficheros de configuración de la red son los siguientes:

- `/etc/hosts`. El principal propóposito de este archivo es resolver los nombres de hosts en su correspondiente dirección IP. Se puede usar para resolver nombres de hosts en pequeñas redes sin servidor DNS. Sin tener en cuenta el tipo de red en que se encuentre el ordenador, este archivo debe contener un línea que especifica la dirección IP del dispositivo loopback (127.0.0.1) como por ejemplo localhost.localdomain.

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1               localhost.localdomain localhost
10.0.0.1                mis0200.dsic.upv.es mis0200
```

- `/etc/resolv.conf`. Este archivo especifica las direcciones IP de los servidores DNS y el dominio de búsqueda. Los scripts de inicialización de la red definen este archivo, aunque es fácilmente editable:

```
search dsic.upv.es
nameserver 158.42.184.2
```

- `/etc/sysconfig/network`. Especifica la información de routing y del host para todas las interfaces de red.

- /`etc/sysconfig/network-scripts/ifcfg-<interface-name>.` Para cada interfaz de red existe un script de configuración de interfaz correspondiente. Cada uno de estos archivos proporcionan información específica para una interfaz de red determinada.

  A continuación se muestra un ejemplo de un archivo `ifcfg-eth0` para un sistema que usa una dirección IP fija:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETWORK=10.0.0.0
```

```
NETMASK=255.255.255.0
IPADDR=10.0.0.1
USERCTL=no
```

El fichero `ifcfg-eth0` para una interfaz que use DHCP:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

- `/etc/modules.conf`. Este fichero mantiene los módulos del kernel que serán cargados en el arranque. El driver de la tarjeta de red debría aparecer aquí.

```
alias eth0 e100
```

La **Herramienta de administración de redes** (**system-config-network**) es una forma fácil de hacer los cambios a los diferentes archivos de configuración.

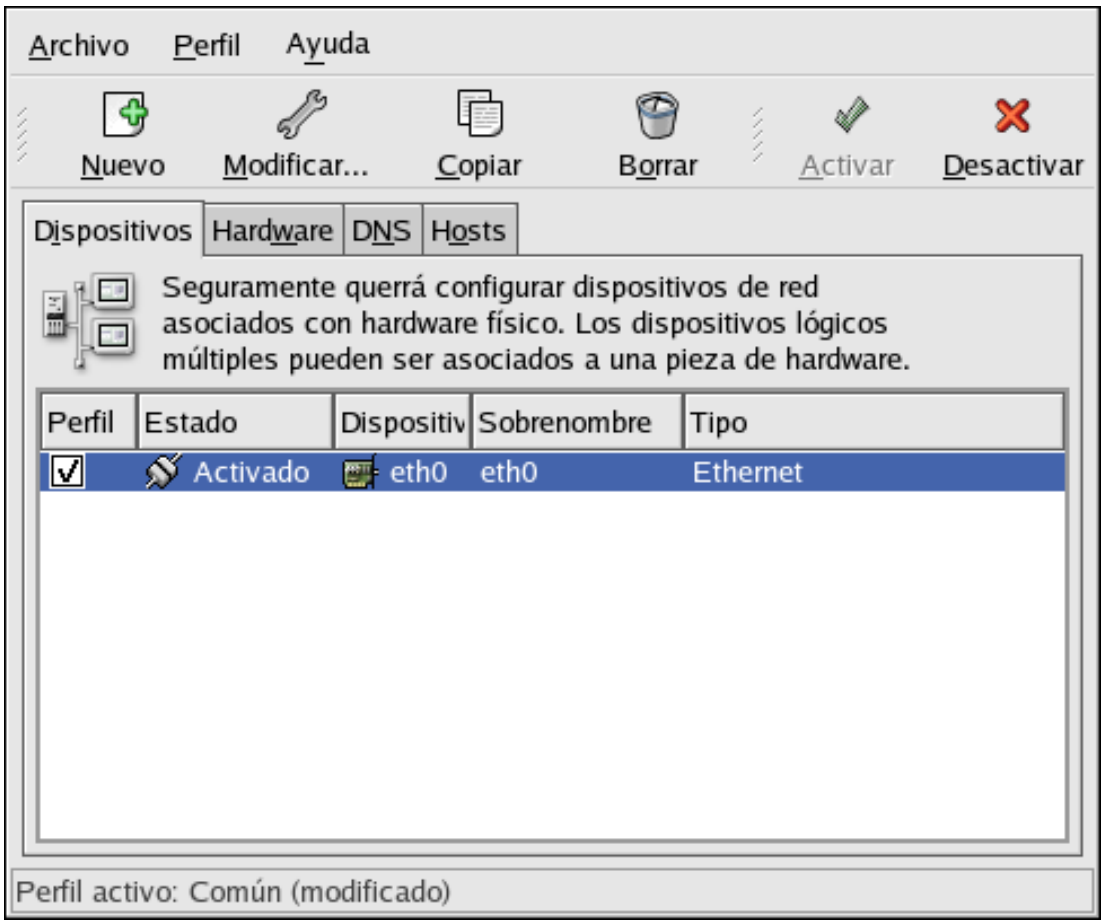

Figura 2.3. Herramienta de administración de red

2.2.3. Configuración del entorno gráfico

La mayoría de distribuciones de Linux intentan configurar la interfaz gráfica, es decir, poder hacer uso de nuestra tarjeta de vídeo y nuestra pantalla, en el mismo proceso de instalación. Pero hay que saber de que utilidades disponemos para configurar el servidor X, si por ejemplo cambiamos de tarjeta de vídeo o queremos mejorar su rendimiento.

system-config-display, es una herramienta propia de los sistemas basados en RedHat que prueba su sistema para intentar determinar el tipo de tarjeta de vídeo que posee.

Figura 2.4. system-config-display

Si no lo consigue, system-config-display presentará una lista con todas las tarjetas de vídeo. Se selecciona una tarjeta de vídeo de la lista y pulse **Intro**. Si su tarjeta de vídeo no aparece en la lista puede deberse a que no esté soportada por XFree86. No obstante, si posee las características técnicas de su tarjeta, debe seleccionar Tarjeta no incluida en la lista e intentar configurarla estableciendo el chipset de vídeo de su tarjeta con uno de los servidores-X disponibles.

2.2.3. Configuración del entorno gráfico

Una vez seleccionada su tarjeta de vídeo, el programa de instalación instalará el servidor XFree86 apropiado, y system-config-display mostrará una lista de monitores. Si su monitor aparece en la lista, seleccionelo y pulse Intro. En otro caso, seleccione Personalizado. Si selecciona Personalizado, Xconfigurator le indica que seleccione el rango de sincronismo horizontal y el rango de sincronismo vertical de su monitor (estos valores, generalmente están disponibles en la documentación que acompaña a su monitor, o mediante su vendedor o fabricante).

A continuación, system-config-display le solicita la cantidad de memoria de vídeo instalada en su tarjeta de vídeo. Si no está seguro, por favor consulte la documentación que acompaña a su tarjeta de vídeo. No se dañará su tarjeta por elegir más memoria de la que está disponible, pero el servidor XFree86 puede que no se inicie correctamente si lo hace.

Si la tarjeta de vídeo seleccionada tiene un chip de reloj de vídeo, redhat-config-xfree86 mostrará una lista de los chips de reloj. La opción recomendada es Sin establecer chip de reloj, desde que XFree86 puede detectar el chip de reloj apropiado en la mayoría de los casos.

A continuación, system-config-display le indica que seleccione los modos de vídeo que desea usar. redhat-config-xfree86 escribirá entonces un fichero de configuración conteniendo todas sus elecciones en /etc/X11/xorg.conf. Finalmente, verá una pantalla en la que se le da la opción de ejecutar el sistema de ventanas X cuando rearranque.

Pero system-config-display no es la única utilidad que dispone para configurar su entorno gráfico. Cualquier sistema Linux viene con dos utilidades, una en modo texto llamada xf86config que le irá guiando en proceso descrito anteriormente, y otra XFSetup en modo gráfico ya que es capaz de arrancar un servidor VGA, y que a base de menús le permite configurar todos los dispositivos necesarios para poder lanzar el entorno de ventanas.

Todas estas utilidades se encargarán de establecer los parámetros adecuados en el fichero de configuración del servidor X (/etc/X11/xorg.conf).

3

Administración del sistema de ficheros

Tabla de contenidos

3.1. Introducción

Un sistema de ficheros son los métodos y las estructuras de datos que emplea el sistema operativo (en nuestro caso, Linux) para organizar los ficheros en disco. El término Sistema de Ficheros se utiliza tanto para referirse a una partición o a un disco. Pero hablando con propiedad, muchos programas no trabajarán satisfactoriamente con una partición que no contenga un sistema de ficheros.

Linux soporta varios tipos de sistemas de ficheros. Entre los más importantes podemos destacar los siguientes:

- *MINIX*: el más antiguo, presume de ser el más seguro, pero es bastante limitado en las características que proporciona. Un sistema de ficheros de este tipo solo puede tener 64 MB.
- *EXT2*: es el sistema de ficheros nativo de Linux. Está diseñado para ser compatible con versiones anteriores, así que las nuevas versiones del código del sistema de ficheros no requerirán rehacer los sistemas de ficheros existentes.
- *EXT3*: es una modificación del ext2 para añadirle funcionalidades de journaling.
- *VFAT*: este tipo permite utilizar sistemas de ficheros de Windows (FAT, FAT32), y actualmente está soportado el sistema de ficheros de Windows NT, pero solo fiable en sololectura.
- *Iso9660*: es el sistema de ficheros estándar para CD-ROM.
- *NFS*: un sistema de ficheros en red que permite compartir sistemas de ficheros entre diferentes máquinas conectadas en red y tratarlos de forma local.
- *HPFS*: es el tipo de sistema de ficheros de OS/2
- *SYSV*: es el tipo de sistema de ficheros de SystemV/386, Coherent y Xenix.

Existe también un sistema de ficheros especial denominado proc, y que es accesible via el directorio /proc, el cual no es realmente un sistema de ficheros. El sistema de ficheros /proc permite acceder fácilmente a ciertas estructuras de datos del kernel, como es la lista de procesos. Convierte estas estructuras de datos en algo parecido a un sistema de ficheros y por tanto da la posibilidad de manipularlas con las herramientas habituales de manipulación de ficheros. Hay que tener en cuenta que aunque se le denomine sistema de ficheros, ninguna parte del sistema de ficheros /proc toca el disco. El existe únicamente en la imaginación del kernel.

3.2. Creación de un sistema de ficheros

Los sistemas de ficheros se crean con el comando **mkfs**. Actualmente, en Linux, existe un programa separado por cada tipo de sistema de ficheros. De esta forma *mkfs* es solamente un 'front-end' que ejecuta el programa apropiado dependiendo del tipo de sistema de ficheros deseado. El tipo de sistema de ficheros se define con la opción -t de mkfs. Por tanto se puede suponer que los programas invocados por mkfs pueden soportar diferentes opciones. Lo mejor es echar un vistazo a la página de manual.

Para crear un sistema de ficheros del tipo ext2 en disquete, podríamos seguir los siguientes pasos:

```
fdformat -n /dev/fd0H1440
badblocks /dev/fd0H1440 1440 > listadebloquesmalos
mkfs -t ext2 -l listadebloquesmalos /dev/fd0H1440
```

En primer lugar formateamos el disquete con la opción -n que previene que chequee bloques malos, para en un segundo paso generar una lista de bloques defectuosos con el comando badblocks. Esa lista se la pasaremos al comando mkfs que creará el sistema de ficheros en el disquete.

3.3. Montaje de sistemas de ficheros

Antes de poder utilizar un sistema de ficheros, este debe de ser montado. A diferencia de otros sistemas operativos (léase Windows) y como cualquier UNIX, Linux emplea una jerarquía (árbol) de directorios único. Por tanto la operación de montaje hará que los contenidos de un sistema de ficheros nuevo parezcan los contenidos de un subdirectorio existente de algún sistema de ficheros ya montado.

Imaginemos dos sistemas de ficheros separados, cada uno con su propio directorio raiz. Cuando el segundo sistema de ficheros se monte bajo el directorio /home en el primer sistema de ficheros, se obtendrá un único árbol de directorios.

El comando que nos permitirá realizar esta operación se llama mount:

```
[root@mis01]# mount -t ext3 /dev/hda2 /home
```

La opción -t le indica al comando mount el tipo de sistema de ficheros que se va a montar, el cual deberá estar soportado por el kernel. Los otros dos argumentos son el fichero dispositivo correspondiente a la partición que contiene el sistema de ficheros, y el directorio sobre el que será montado. El directorio sobre el que se monta no debe de estar vacío, pero si que debe existir. Cualquier fichero que haya en ese directorio será inaccesible mientras el sistema de ficheros esté montado.

Si por ejemplo no deseáramos que nadie pudiera escribir en el sistema de ficheros se po-

dría haber usado la opción -r de mount para indicar que es un sistema de ficheros de solo lectura. Esto obligará al kernel a detener cualquier intento de escritura sobre el sistema de ficheros y también detendrá cualquier actualización de los tiempos de acceso de los inodos.

En este momento alguien puede preguntarse como es posible que el sistema de ficheros raiz se monte, ya que no puede ser montado en otro sistema de ficheros. Entonces si es sistema de ficheros raíz no puede ser montado, el sistema no arrancará nunca. La respuesta está en que ese sistema de ficheros que se monta como root está compilado en kernel o definido utilizando LILO o el comando rdev.

Cuando un sistema de ficheros no se necesita, se puede desmontar utilizando la orden umount, la cual toma como argumento el fichero dispositivo que define la partición que alberga al sistema de ficheros o el nombre del directorio sobre el que se ha montado.

Un ejemplo claro son los disquetes, que no deberían ser extraídos sin previamente haber desmontado el sistema de ficheros. Ya que debido al caché de disco los datos no son necesariamente escritos hasta que se desmonta.

3.3.1. /etc/fstab

En la mayoría de sistemas existen otras particiones además de la raíz que son necesarias se monten en el arranque. Estas serán especificadas en el fichero /etc/fstab que contiene todas las informaciones que conciernen el montaje de sus particiones.

```
#/etc/fstab
# Dispositivo    Direct     type        options      frequence  passe
/dev/hda2        /          ext3        defaults        5        1
/dev/hdb2        /usr2      ext3        defaults        5        2
/dev/sda2        /usr3      ext3        defaults        10       2
/dev/hda1        /dos       msdos       defaults        0        0
/dev/hdb1        /dos2      msdos       defaults        0        0
none             /proc      proc        defaults        0        0
/dev/hda3        none       swap        defaults        0        0
/usr2/swapfile   /usr2      swap        defaults        0        0
```

Cada una de las columnas tiene el significado que comentamos a continuación.

- device (dispositivo) de la partición. En el caso de un archivo de swap, es el nombre del archivo.
- directorio de montaje de la partición.
- tipo de la partición.
- opciones (sólo lectura, etc)
- frecuencia: correspondiente al número de días entre dos tratamientos del archivo por la orden dump. Esta orden existe solamente para ext2fs (es una migración de la versión

4.4BSD) pero no está aún incluida en todas las distribuciones.

- orden de test de las particiones (fsck). Si pone 0 la verificación automática no es efectuada al arrancar. Las particiones situadas sobre un mismo disco serán verificadas de manera secuencial, pero si están situadas en dos discos diferentes se hará en paralelo.

Existen mas opciones que la de por defecto a la hora de montar un dispositivo. Para autorizar a un usuario a montar un volumen, tiene que crear una linea que contenga la opción "user" Ejemplo (caso de un CD-ROM SCSI) :

```
/dev/cdrom      /mnt/cdrom   iso9660   user,exec,dev,nosuid,ro,noauto
```

Cualquier usuario podrá, a partir de ese momento, montar y desmontar un CD (utilizar mount /mnt/cdrom, umount /mnt/cdrom) La página de man de mount, sección 8 (man 8 mount) explica el significado de las opciones posibles.

3.4. Sistema de cuotas

El almacenamiento en disco se puede restringir mediante la implementación de cuotas de disco. Las cuotas se pueden configurar para usuarios individuales o para grupos. Este tipo de flexibilidad hace posible darle a cada usuario una pequeña porción del disco para que maneje sus archivos personales (tales como correo o informes), mientras que se le permite tener más espacio para manejar los proyectos en los que estén trabajando.

Además, se puede configurar las cuotas no sólo para que controlen el número de bloques de disco sino también el número de inodos. Debido a que los inodos son usados para almacenar información relacionada a los archivos, esto permite controlar el número de archivos que pueden ser creados.

El soporte de cuotas de disco ha sido integrado en el kernel Linux desde la versión 1.3.46. Se necesita utilizar un kernel posterior para poder beneficiarse de las cuotas. El paquete software necesario que permite gestionar las cuotas es quota. Además necesitamos tener esa opción compilada en el kernel respondiendo afirmativamente a la opción Quota support. Con esto conseguiremos limitar el espacio de disco consumido por usuario o por un grupo de usuarios.

3.4.1. Activación de las cuotas

Existen dos tipos de cuotas: las cuotas asociadas a los usuarios y las cuotas asociadas a los grupos de usuarios. Las primeras definen el número máximo de archivos y de bloques de disco asociados a cada usuario, las segundas definen el número máximo de archivos asociados a cada grupo de usuarios. Los dos tipos de cuotas pueden ser activados separadamente.

Para activar las cuotas para los usuarios es necesario indicar la opción usrquota para los sistemas de archivos referidos en /etc/fstab. Las cuotas que conciernen a los grupos son regu-

ladas por la opción grpquota. Los archivos de definición de cuotas se llaman respectivamente quota.user y quota.group y están situados en la raíz de cada sistema de archivos involucrado.

Es posible modificar los nombres de los archivos de gestión de cuotas utilizando la sintáxis siguiente:

```
usrquota=nombredearchivo
grpquota=nombredearchivo
```

He aquí un ejemplo del archivo /etc/fstab:

```
/dev/hda2    /         ext3   defaults,rw 0 1
/dev/hdb2    /home     ext3   defaults,rw,usrquota,grpquota 0 1
/dev/sda1    /usr/src  ext3   dzfaults,rw,usrquota 0 1
```

La activación de las cuotas es lanzada por la orden quotaon. Para activarlas automáticamente a la inicialización del sistema, se debe agregar al archivo de inicialización (/etc/rc.d) las líneas:

```
quotaon -avug
```

Para crear los archivos de cuotas (aquota.user y aquota.group) en el sistema de archivos, se usa la opción -c del comando quotacheck.

```
quotacheck -acug /home
```

Puede ser igualmente necesario verificar la coherencia de la información sobre gestión de cuotas después de bloqueos repentinos. Para esto se utiliza la orden quotacheck:

```
quotacheck -avug
```

Si las utilidades han sido compiladas con-DEXTc2DIRECT, la orden quotacheck debe ejecutarse relativamente rápido, en caso contrario puede ser muy lento, pues debe explorar todos los directorios del disco. Es aconsejable de todas maneras ejecutar quotacheck durante la inicialización del sistema antes de la activación de las cuotas con quotaon.

3.4.2. Cuotas de usuario

La orden edquota es utilizada para asignar una cuota a un usuario o a un grupo de usuarios. Su sintáxis es edquota -u usuario o edquota -g grupo. Esta orden lanza un editor de texto que contiene la definición de cuotas asignadas al usuario o al grupo y toma en cuenta el nuevo valor cuando el archivo es reescrito.

Para cada usuario o grupo existen dos limitaciones: el número de archivos y el número de bloques disco (expresados en bloques de 1024 octetos). Para cada uno existen dos límites:

- El límite "*suave*": cuando este límite es alcanzado o superado un mensaje advierte al usuario cada vez que un nuevo bloque o archivo es escrito.
- El límite "*duro*": cuando este límite es alcanzado el usuario no puede escribir nuevos archivos o nuevos bloques.

El límite "suave" se transforma en límite "duro" cuando ha sido alcanzado o superado transcurrido cierto tiempo (7 días por defecto).

Se puede definir una plantilla de usuario al cual se le asignan las cuotas y utilizarla para adjudicar cuotas a los demás usuarios del sistema.

```
edquota -p usuarioplantilla -u usuario
```

Todo usuario puede obtener el estado de la cuota que le ha sido asignada (limites como el número de archivos y de bloques que le han sido atribuidos) gracias a la orden quota.

El superusuario puede obtener las mismas informaciones sobre cualquier usuario o grupo con la misma orden : quota -u usuario o quota -g grupo. Además es posible utilizar la orden repquota para obtener una lista de cuotas asociadas a uno o varios sistemas de archivos.

3.5. Memoria virtual: swap

Ya se vio a la hora de instalar el sistema RedHat Linux que se nos preguntaba por la creación de un espacio en disco que sirviera como intercambio cuando el kernel necesita descargar procesos a disco ya que no dispone de más memoria. Entonces, en ese momento creamos una partición de swap del doble de la memoria disponible. La creación de la memoria virtual puede ser a través de una partición expresamente para eso o a través de un fichero.

3.5.1. Sistema de ficheros de swap

Para añadir una partición de swap de 8 Mb , se tiene que crear primero la partición gracias al programa fdisk, y luego:

```
mkswap /dev/hda3 8192
```

Para que esto se tenga en cuanta cada vez que arranque el sistema habría que añadir la línea

```
/dev/hda3     swap     swap defaults 0 0 en el archivo /etc/fstab
```

Finalmente, para que funcione, re-arranque la máquina o ejecute swapon -a.

3.5.2. Creación de un fichero de swap

Para el archivo, es más complicado. Debe hacer :

```
dd if=/dev/zero of=/usr2/swapfile bs=1024 count=8192
mkswap /usr2/swapfile 8192
```

Y agregar en el archivo /etc/fstab la línea

```
/usr2/swapfile   /usr2 swap   defaults 0 0
```

Hay que tener en cuenta que se debe poner esta línea después del montaje de la partición /usr2. Sino, no funcionará nunca.

De ahora en adelante, para salir de Linux estará obligado a hacer :

```
swapoff -a
reboot
```

En efecto, si no se desactiva el archivo de intercambio, Linux no podrá desmontar la partición y por consiguiente hará un fsck sobre ella cada vez que arranque la máquina. Aunque eso ya está contemplado en los scripts de arranque del sistema.

3.6. Copias de seguridad en linux

Como todo sistema UNIX, Linux provee herramientas estándar para realizar las copias de seguridad de los discos.

Dos tipos de herramientas principales son actualmente utilizadas.

3.6.1. dump y restore

Las herramientas dump y restore fueron puestas bajo Linux por Remy Card. El paquete necesario en RedHat Linux es dump-0.4b21-3.

Una vez que las fuentes han sido compiladas e instaladas, la utilización de dump y restore es relativamente simple. Para realizar la copia de seguridad de una partición /dev/sda1 sobre /dev/rmt0, es suficiente hacer:

```
# dump 0sfu  3600  /dev/rmt0  /dev/sda1
# dump 0sfu mis02:/dev/rmt0  /dev/sda1
```

La segunda orden permite copia de seguridad de un disco sobre un dispositivo remoto (por ejemplo situado aquí sobre la máquina "mis02"). Las opciones de dump pueden parecer

complejas. A continuación damos una corta descripción:

- 0 a 9 : nivel de copia de seguridad. 0 corresponde a una copia de seguridad completa, mientras que los otros niveles n corresponden a la copia de seguridad de archivos que fueron modificados desde la enésima copia de seguridad;
- s : tamaño de la cinta en pies;
- f : archivo. Puede estar compuesto de máquina:archivo;
- u : escritura de la fecha y del nivel de copia de seguridad en el archivo /etc/dumpdates.

Existen otras opciones. Para mayor información, consultar las páginas del manual. Existen dos maneras de efectuar una restauración : en línea de ordenes o en modo llamado "interactivo". El segundo modo es más simple para las restauraciones parciales. El primero es sobre todo utilizado para las restauraciones completas.

Para restaurar la cinta en modo interactivo es suficiente hacer:

```
# restore -if /dev/rmt0
# restore -if mis02:/dev/rmt0
```

En este caso, un mini-intérprete de órdenes es ejecutado. Utilice la orden help para más detalles.

Para restaurar completamente una cinta:

```
# restore rf /dev/rmt0
```

Para la utilización de dump y restore a través de una red (copia de seguridad sobre dispositivos remotos), debe utilizar los archivos .rhosts. En el siguiente ejemplo de copia de seguridad, la máquina "mis01" debe tener:

```
# cat ~root/.rhosts
fferrer
```

Cuidado de todas formas con los fallos de seguridad engendrados por los ficheros .rhosts. El uso de dispositivos remotos necesita igualmente de la presencia del programa rmt en la máquina que maneja los dispositivos de copia de seguridad. Este programa está incluido en la distribución fuente de dump para Linux.

3.6.2. tar

A diferencia de dump o restore, tar permite copia de seguridad de los archivos deseados, excluir ciertos repertorios, etc. Es necesario notar que el tar utilizado bajo Linux es el tar GNU. Este posee ciertas opciones particulares.

Para conocer todas las opciones posibles, te aconsejo hacer tar --help. Una utilización simple de tar puede ilustrarse con la copia de seguridad de una partición de usuarios:

```
# tar cvf /dev/rmt0 /users | mail backup-user
```

La lista de archivos será así enviada al usuario backup-user. Ciertos sitios utilizan exclusivamente tar para efectuar sus copia de seguridad, cada cual escoge lo mas apropiado.

4

Protección Local

Tabla de contenidos

4.1. La tabla de usuarios

Los usuarios de un sistema Unix son registrados en el fichero `/etc/passwd`. Cada línea corresponde a un usuario y describe los atributos del mismo. Los atributos son separados por el carácter ':'.

Por ejemplo, dada la siguiente línea del fichero `/etc/passwd`:

```
usu1:$1$ZtoHwEykKW/eCLHzSUigg5KAEyxa51:1002:2000:Usuario 1:/home/usu1:/bin/bash
```

la tabla a continuación describe cada elemento de la misma:

Tabla 4.1. Tabla de usuarios

Elemento	**Significado**
usu1	Nombre de usuario (*login name*)
1ZtoHwEykKW/eCLH zSUigg5KAEyxa51	Contraseña cifrada
1002	Identificador del usuario (UID)
2000	Indentificador del grupo primario al que pertenece el usuario (GID)
Usuario 1	Descripción del usuario
`/home/usu1`	Directorio de conexión inicial del usuario.
`/bin/bash`	Programa intérprete de órdenes (shell)

Existen varios usuarios especiales predefinidos en el sistema, normalmente aquellos cuyo UID es inferior a 500. Entre estos usuarios cabe destacar a `root`, cuyo UID es igual a 0. Este es el administrador del sistema, conocido generalmente como el *superusuario*. Otros usuarios especiales sirven para asociar niveles de acceso a ciertos servicios del sistema, como por ejemplo los usuarios `mail` o `news`. Otro caso destacable es el usuario `nobody`, normalmente empleado para representar conexiones de red realizadas por usuarios desconocidos o anómimos. Todos estos usuarios se crean durante la instalación del sistema o cuando se instalan ciertos paquetes y, en principio, no deben modificarse nunca.

4.2. La extension de la tabla de usuarios

Además de la tabla de usuarios descrita en el apartado anterior, en la mayoría de sistemas Unix actuales se utiliza otra tabla, contenida en el fichero `/etc/shadow`, en la que se almacena información adicional para los usuarios.

Cada línea de este fichero se corresponde con una línea del fichero `/etc/passwd` y la

información que contiene es la siguiente (de acuerdo con el ejemplo anterior):

```
usu1:$1$ZtoHwEykKW/eCLHzSUigg5KAEyxa51:10989:0:99999:7:-1:-1:134538436
```

Tabla 4.2. Extensión de la tabla de usuarios

Elemento	Significado
usu1	Nombre de usuario (*login name*)
1ZtoHwEykKW/eCLH zSUigg5KAEyxa51	Contraseña cifrada
10989:0:99999:7: -1:-1:134538436	Información de caducidad de la contraseña

Cuando el sistema emplea esta tabla, la contraseña que debería estar almacenada en el fichero `/etc/passwd` se sustituye por una 'x'. Dado que el fichero `/etc/passwd` es legible por todos los usuarios mientras que `/etc/shadow` sólo es legible por el administrador, mediante este cambio se consigue que ningún usuario tenga acceso a las contraseñas cifradas.

La información de caducidad de la contraseña está expresada en días, siendo su utilización e interpretación compleja en algunos casos. Por este motivo, para modificar y consultar esta información no se recomienda en general trabajar directamente con el fichero `/etc/shadow`, sino utilizar herramientas administrativas específicas, ya sea en forma de mandatos (**passwd** y **chage**) o mediante aplicaciones gráficas.

4.3. La tabla de grupos

Los grupos de un sistema Unix son registrados en un fichero denominado `/etc/group`. Cada línea corresponde a un grupo y describe los atributos del mismo. Los atributos son separados por el carácter `:' y su significado se describe mediante el siguiente ejemplo:

```
proy1::65534:usu1,usu2,usu3
```

Tabla 4.3. Extensión de la tabla de usuarios

Elemento	Significado
proy1	Nombre del grupo
	Contraseña cifrada (vacía en el ejemplo). De conocerla, un usuario podría cambiar su grupo primario. Sin embargo, esta funcionalidad está en desuso
65534	Identificador del grupo (GID)
usu1,usu2,usu3	Lista de usuarios que pertenecen a este grupo (separados por comas)

Como se deduce del ejemplo, un usuario puede figurar en la lista de distintos grupos, es decir, puede pertenecer a *varios* grupos. De todos esos grupos, aquél cuyo GID figura en la entrada correspondiente al usuario en el fichero `/etc/passwd` se considera el *grupo primario* de dicho usuario. El resto de grupos se denominan *grupos suplementarios*.

Al igual que sucede con los usuarios, existen ciertos grupos que son especiales, en concreto aquellos cuyo GID es menor que 500. De forma análoga a los usuarios especiales, representan servicios, usuarios anónimos, etc. No deben, por tanto, ser modificados.

4.4. Procedimientos para la creación de grupos y usuarios

En la mayoría de sistemas Unix System V pueden emplearse los siguientes mandatos para la gestión de usuarios y grupos:

Tabla 4.4. Extensión de la tabla de usuarios

Mandato	Uso
useradd	Añadir un usuario
userdel	Eliminar un usuario
usermod	Modificar los atributos de un usuario
groupadd	Añadir un grupo
groupdel	Eliminar un grupo
groupmod	Modificar los atributos de un grupo
passwd	Cambiar la contraseña de un usuario
chage	Gestionar la caducidad de la contraseña

Estos mandatos son especialmente útiles cuando la gestión de usuarios debe realizarse

desde \e{scripts} del \e{shell}, por ejemplo, cuando debe crearse una gran cantidad de usuarios de forma automática. La sintaxis y opciones más usuales de estos mandatos se citan en el Sección 4.13, “Sintaxis y funcionamiento de los mandatos Unix relacionados” al final de este capítulo.

Adicionalmente, cada vez es más frecuente la disponibilidad de herramientas gráficas que facilitan la labor del administrador, tal como la utilidad Gestor de usuarios (o "User Manager") de RedHat.

4.5. Atributos de protección de los procesos

Los atributos de un proceso que intervienen en el mecanismo de protección son:

Identificadores del usuario propietario del proceso	En realidad, cada proceso contiene dos versiones del identificador de usuario, la denominada versión *real* (rUID) y la versión *efectiva* (eUID). La versión real siempre corresponde al usuario que creó el proceso. La versión efectiva corresponde al usuario bajo el cual se comporta el proceso y es el utilizado en el mecanismo de protección. Ambos identificadores suelen ser iguales, salvo que intervenga el denominado bit `SETUID` en el fichero ejecutable que está ejecutando el proceso, tal como se describe más adelante en este documento (ver Sección 4.9, “Los bits SETUID y SETGID en ficheros ejecutables ”).
Identificadores del grupo propietario del proceso	En este caso, el proceso también contiene una versión real (rGID) y otra efectiva (eGID) del identificador. La versión real corresponde al grupo primario al que pertenece el usuario que creó el proceso. La versión efectiva corresponde al grupo bajo el cual se comporta el proceso y es el utilizado en el mecanismo de protección. Ambos identificadores suelen ser iguales, salvo que intervenga el denominado bit `SETGID` en el fichero ejecutable que está ejecutando el proceso, tal como se describe más adelante en este documento (ver Sección 4.9, “Los bits SETUID y SETGID en ficheros ejecutables ”).
Lista de grupos suplementarios	Es la lista formada por los grupos suplementarios del usuario que creó el proceso.

Estos atributos son asignados al proceso en el momento de su creación, y heredados de su proceso padre. Cada usuario conectado a un sistema Unix posee un proceso de atención, el cual puede ser un entorno de ventanas o un intérprete de órdenes (*shell*). En cualquiera de ambos casos, este proceso es el padre de todos los procesos que el usuario genera. Este proceso shell recibe sus atributos no por herencia, sino en el momento en el que un usuario se

acredita al sistema mediante su nombre de usuario y contraseña asociada. Los atributos rUID y rGID se extraen de la tabla de usuarios `/etc/passwd`. La lista de grupos suplementarios es confeccionada a partir de la tabla de grupos `/etc/group`. Los atributos eUID y eGID son asignados a los valores respectivos de rUID y rGID. Un mandato interesante al respecto es **id**, el cual muestra todos estos atibutos.

4.6. Atributos de protección de los ficheros

Los atributos de un fichero que intervienen en el mecanismo de protección son:

`OwnerUID` Identificador del usuario propietario del fichero.

`OwnerGID` Identificador del grupo propietario del fichero.

Bits de permiso Un total de 12 bits que expresan las operaciones del fichero que son permitidas en función del proceso que acceda a este fichero. La Tabla 4.5, “Significado de los bits de permiso en Unix” a continuación muestra el significado de cada uno de estos bits.

Tabla 4.5. Significado de los bits de permiso en Unix

Bit	**Significado**
11	`SETUID`
10	`SETGID`
9	`Sticky`
8	Lectura para el propietario.
7	Escritura para el propietario.
6	Ejecución para el propietario
5	Lectura para el grupo.
4	Escritura para el grupo.
3	Ejecución para el grupo.
2	Lectura para el resto de usuarios.
1	Escritura para el resto de usuarios.
0	Ejecución para el resto de usuarios.

El significado de los bits de lectura, escritura y ejecución es diferente en función del tipo de archivo que los defina. Para ficheros regulares, su significado es el esperable (permiten leer, modificar y ejecutar el fichero respectivamente). Evidentemente, el bit de ejecución sólo tiene sentido si el fichero es un ejecutable o contiene un *shellscript*.

En un directorio, el significado de estos tres bits es el siguiente:

Lectura	Puede listarse el contenido del directorio.
Escritura	Permite la creación, eliminación o renombrado de ficheros o directorios dentro del directorio al cual se aplica este bit.
Ejecución	Permite la utilización del directorio al cual se aplica este bit para formar parte de un nombre de ruta, es decir, puede utilizarse el directorio para nombrar un fichero.

De lo anteriormente expuesto, puede observarse que no existe un bit de permiso específico para el borrado de un fichero/directorio. Dicho permiso es controlado realmente desde el directorio donde reside el fichero que quiere eliminarse. En algunos sistemas Unix (como por ejemplo RedHat Linux), el bit `sticky` es utilizado para modificar la regla de eliminación de ficheros: si se activa en un directorio, un usuario puede borrar un fichero en él sólo si es el propietario de dicho fichero. Los bits `SETUID` y `SETGID` son tratados en apartados posteriores.

El `ownerUID` puede modificarse con el mandato **chown**. El `ownerGID` puede modificarse con el mandato **chgrp**. Los bits de permiso pueden modificarse con el mandato **chmod**. Todos estos mandatos se describen en Sección 4.13, “Sintaxis y funcionamiento de los mandatos Unix relacionados”. .

4.7. Las reglas de protección básicas

Las reglas de protección básicas se activan cuando un proceso notifica al sistema que desea utilizar un determinado fichero. El proceso también notifica al sistema qué tipo de operaciones quiere realizar: lectura, escritura o ejecución.

- Si el `eUID` del proceso es 0 se concede el permiso (estamos en el caso en el que el proceso pertenece al superusuario). Si no...
- Si el `eUID` del proceso es igual al `ownerUID` del fichero, se autoriza la operación si está permitida en el grupo de bits 6 al 8, los que corresponden al propietario. Si no...
- Si el `eGID` del proceso o alguno de los grupos suplementarios del proceso es igual al `ownerGID` del fichero, se autoriza la operación si está permitida en el grupo de bits 3 al 5, los que corresponden al grupo. Si no...
- En cualquier otro caso, se autoriza la operación si está permitida en el grupo de bits 0 al 2, los que corresponden al resto de usuarios.

Debe notarse que sólo se aplica una regla, aquella que corresponde a la primera condición que se cumple para los atributos del proceso. Es decir, el sistema determina *primero* qué grupo de tres bits debe aplicar y *después* autoriza (o deniega) la operación en función del tipo de operación requerida y del estado de dichos tres bits.

4.8. Cambio de atributos de protección en ficheros

Unix establece unas reglas de protección específicas para controlar los cambios de cualquier atributo de protección de un fichero, dado que dichas modificaciones se consideran operaciones *distintas* de la de escritura, y por tanto no pueden ser concedidas/denegadas en función de los bits de permiso del fichero.

En concreto, las reglas establecidas al respecto son las siguientes:

a. *Cambio en los bits de permiso*. Un proceso puede cambiar los bits de permiso de un fichero sólo si:

   - el `eUID` del proceso es 0 (estamos en el caso en el que el proceso pertenece al superusuario), o bien
   - el `eUID` del proceso es igual al `ownerUID` del fichero.

   Es decir, sólo el superusuario o el propietario de un fichero pueden modificar sus bits de permiso.

b. *Cambio de propietario*. El cambio de propietario de un fichero puede realizarlo tan sólo el superusuario.

c. *Cambio de grupo propietario*. El cambio del grupo propietario de un fichero puede realizarse sólo si:

   - lo realiza el superusuario, o bien
   - lo realiza el propietario del fichero, siempre que el nuevo `ownerGID` del fichero sea igual a alguno de los grupos de dicho usuario.

4.9. Los bits SETUID y SETGID en ficheros ejecutables

Estos bits se emplean para permitir que un programa se ejecute bajo los privilegios de un usuario distinto al que lanza la ejecución del programa. Su funcionamiento es el siguiente:

- Si el fichero ejecutable tiene el bit `SETUID` activo, el `eUID` del proceso que ejecuta el fichero es hecho igual al `ownerUID` del fichero.
- Si el fichero ejecutable tiene el bit `SETGID` activo, el `eGID` del proceso que ejecuta el fichero es hecho igual al `ownerGID` del fichero.

Normalmente estos programas pertenecen al superusuario, y permiten a los usuarios ejecutar tareas privilegiadas bajo ciertas condiciones. El cambio de la contraseña de un usuario es un ejemplo de esta técnica.

La existencia de estos ficheros en el sistema de archivos debe ser cuidadosamente supervisada por el superusuario. Muchos de los ataques de seguridad a Unix utilizan estos ficheros para conseguir atentar contra la seguridad del sistema.

4.10. El bit SETGID en directorios

La utilización de este bit está orientada a facilitar el trabajo en grupo cuando varios usuarios deben acceder a una colección común de ficheros y directorios. Su funcionamiento es el siguiente: si un directorio, llamémosle `D`, tiene el bit `SETGID` activo, entonces:

- si se crea un fichero dentro de `D`, el `ownerGID` del nuevo fichero es hecho igual al `ownerGID` del directorio `D`.
- si se crea un directorio dentro de `D`, el `ownerGID` del nuevo directorio es hecho igual al `ownerGID` del directorio `D` y su bit `SETGID` es activado.

Puede observarse que se trata, en esencia, de un mecanismo de herencia. Cuando el bit `SETGID` no está activo, el `ownerGID` de un fichero o directorio se toma del `eGID` del proceso que lo crea. En cambio, utilizando el `SETGID`, aunque varios usuarios creen ficheros dentro de un directorio, todos ellos pertenecerán al menos al mismo grupo, con lo que una utilización adecuada de los bits de permiso puede hacer que todos ellos puedan, por ejemplo, leer y modificar dichos ficheros.

4.11. La máscara de creación de ficheros

Las utilidades de Unix que crean ficheros utilizan por defecto la palabra de protección `rw-rw-rw-` si se trata de ficheros no ejecutables o `rwxrwxrwx` si se crea un directorio o un

ejecutable. Puede observarse, por tanto, que todos los permisos son activados.

Para modificar este funcionamiento, cada usuario puede especificar al sistema aquellos bits que no desea que sean activados, mediante la máscara de creación de ficheros. La máscara se establece mediante el mandato **umask**. Cada bit activo en la máscara es un bit que será desactivado cuando se cree un fichero. Por ejemplo, una máscara 022 desactivaría los bits de escritura para el grupo y el resto de usuarios.

El administrador debe velar porque exista una máscara de creación de ficheros razonable por defecto para cualquier usuario. Esto puede conseguirse fácilmente utilizando el fichero `/etc/profile`, el cual sirve para personalizar el entorno de los usuarios en el momento de su conexión.

4.12. La estrategia de los grupos privados

Esta estrategia, empleada por defecto en Redhat Linux, está orientada a racionalizar y flexibilizar la asignación de usuarios a grupos. Las claves de esta estrategia son:

- Crear un grupo privado para cada usuario (utilizando para ello el mismo nombre del usuario), y hacer que éste sea el grupo primario del usuario.
- Utilizar exclusivamente grupos suplementarios para agrupar usuarios.
- Utilizar el bit `SETGID` y un grupo suplementario en aquellos directorios donde varios usuarios tengan que colaborar.
- Utilizar el grupo privado en el directorio de conexión de cada usuario.
- Asignar como máscara por defecto para todos los usuarios `00X`, es decir, todos los permisos activos para el propietario y para el grupo y lo que se considere oportuno para el resto de usuarios.

Se invita al lector a reflexionar sobre las consecuencias de esta estrategia.

4.13. Sintaxis y funcionamiento de los mandatos Unix relacionados

useradd

```
------------------------------------------------------------------------------
useradd  [-u uid [-o]] [-g group] [-G group,...]
         [-d home] [-s shell] [-c comment] [-m [-k template]]
         [-f inactive] [-e expire mm/dd/yy] [-p passwd] [-n] [-r] name
useradd  -D [-g group] [-b base] [-s shell] [-f inactive]
         [-e expire mm/dd/yy]

Crea un usuario. Opciones:
```

```
        -c  Comentario sobre el usuario
        -d  Directorio de conexión del usario
        -e  Fecha en la cual la cuenta de usuario se desactiva
        -f  Días que transcurrirán desde la caducidad de la
            contraseña hasta la desactivación de la cuenta
        -g  Nombre o número del grupo primario
        -G  Lista de los grupos suplementarios del usuario
            (separados por ,) (sin espacios)
        -m  Crea el directorio de conexión. Si no se especifica la opción
            -k, copia al directorio de conexión los ficheros del directorio
            /etc/skel
        -k  Copia al directorio de conexión los ficheros del directorio
            template
        -p  Asigna una contraseña cifrada al usuario
        -r  Crea una cuenta del sistema
        -s  Asigna el shell a utilizar por el usuario
        -u  Asigna un UID concreto al usuario
        -o  Permite crear un UID duplicado con la opción -u
        -D  Se asignan valores por defecto para las opciones indicadas.
            (No crea usuario)
-------------------------------------------------------------------------------
```

userdel

```
-------------------------------------------------------------------------------
userdel [-r] name

Elimina un usuario. Opciones:

        -r  Elimina el directorio de conexión del usario
-------------------------------------------------------------------------------
```

usermod

```
-------------------------------------------------------------------------------
usermod  [-u uid [-o]] [-g group] [-G group,...]
         [-d home [-m]] [-s shell] [-c comment] [-l new_name]
         [-f inactive] [-e expire mm/dd/yy] [-p passwd] [-L|-U] name

Modifica atributos de un usuario. Opciones:

        -c  Comentario sobre el usuario
        -d  Directorio de conexión del usario
        -e  Fecha en la cual la cuenta de usuario se desactiva
        -f  Días que transcurrirán desde la caducidad de la contraseña
            hasta la desactivación de la cuenta
        -g  Nombre o número del grupo primario
        -G  Lista de los grupos suplementarios del usuario
            (separados por ,) (sin espacios)
        -l  Modifica el nombre de conexión del usuario
        -L  Bloquea la contraseña del usuario
        -m  Crea el directorio de conexión Si no se especifica la opción -k,
            copia al directorio de conexión los ficheros del directorio
            /etc/skel
        -p  Asigna una contraseña cifrada al usuario
        -s  Asigna el shell a utilizar por el usuario
```

```
        -u  Asigna un UID concreto al usuario
        -o  Permite crear un UID duplicado con la opción -u
        -U  Desbloquea la contraseña del usuario
------------------------------------------------------------------------------
```

groupadd

```
------------------------------------------------------------------------------
groupadd [-g gid [-o]] [-r] group

Crea un grupo de usuarios. Opciones:

        -g  Asigna un GID concreto al grupo
        -o  Permite crear un GID duplicado con la opción -g
        -r  Crea una cuenta de grupo del sistema
------------------------------------------------------------------------------
```

groupmod

```
------------------------------------------------------------------------------
groupmod [-g gid [-o]] [-n name] group

Modifica los atributos de un grupo de usuarios. Opciones:

        -g  Asigna un GID concreto al grupo
        -o  Permite crear un GID duplicado con la opción -g
        -n  Cambia el nombre del grupo
------------------------------------------------------------------------------
```

groupdel

```
------------------------------------------------------------------------------
groupdel group

Elimina un grupo de usuarios. No tiene opciones.
------------------------------------------------------------------------------
```

passwd

```
------------------------------------------------------------------------------
passwd [-l] [-u [-f]] [-d] [-S] [ username ]

Modifica la contraseña y el estado de la cuenta un usuario. Opciones:

         -d  Elimina la contraseña del usuario
         -f  Fuerza el desbloqueo
         -l  Bloquea la cuenta del usuario
         -S  Informa sobre el estado de la cuenta de un usuario
         -u  Desbloquea la cuenta del usuario
```

```
Sin argumentos modifica la contraseña del usuario que lo invoca.
Si se especifica un nombre de usuario y no se especifican opciones,
modifica la contraseña del usuario indicado
-------------------------------------------------------------------------
```

chage

```
-------------------------------------------------------------------------
chage [ -l ] [ -m min_days ] [ -M max_days ] [ -W warn ]
      [ -I inactive ] [ -E expire ] [ -d last_day ] user

Modifica la información de caducidad de la contraseña de un usuario.
Opciones:

        -d  Asigna la fecha del último cambio de contraseña
        -E  Asigna la fecha en la cual la cuenta de usuario será desactivada
        -l  Muestra la información de caducidad
        -I  Número de días tras los cuales la cuenta será desactivada si no
            se realiza el cambio de contraseña exigido
        -m  Número de días mínimo permitido entre cambios de contraseña
        -M  Número de días máximo permitido entre cambios de contraseña
        -W  Número de días previos al próximo cambio de contraseña exigido
-------------------------------------------------------------------------
```

chown

```
-------------------------------------------------------------------------
chwon [ -R ] owner file

Modifica el usuario propietario de un fichero o directorio. Opciones:

        -R  Realiza la modificación en todo el contenido del directorio
-------------------------------------------------------------------------
```

chgrp

```
-------------------------------------------------------------------------
chgrp [ -R ] group file

Modifica el grupo propietario de un fichero o directorio. Opciones:

        -R  Realiza la modificación en todo el contenido del directorio
-------------------------------------------------------------------------
```

chmod

```
-------------------------------------------------------------------------
chmod -R  MODE[,MODE]... FILE...
```

```
chmod -R  OCTAL_MODE FILE...


Modifica los bits de permiso de un fichero o directorio. Opciones:

        -R  Realiza la modificación en todo el contenido del directorio

MODE  indica como deben modificarse los bits de permiso. Su sintaxis es:

                      [ugoa][+-=][rwxXstugo]

[ugoa]                u:propietario    g:grupo      o:otros       a:todos

[+-=]           + añadir        - eliminar      = hacer igual

[rwxXstugo]     r lectura
                w escritura
                x ejecución
                X ejecución si activo para el propietario
                s SETUID o SETGID
                t sticky
                u igual al propietario
                g igual al grupo
                o igual a otros
--------------------------------------------------------------------------
```

5

El sistema de ficheros en red: NFS

Tabla de contenidos

5.1. Introducción

NFS (Network File System) permite acceder a ficheros y directorios remotos de la misma forma que se accedería a ficheros locales. NFS utiliza al igual que NIS, RPC. En Linux esto es posible, gracias a una mezcla de funcionalidad del Kernel en el cliente y un demonio servidor de NFS en el servidor.

La forma de trabajar de NFS es la siguiente. Un cliente intenta montar (conectar a su árbol de directorios) un directorio desde un host remoto en un directorio local de la misma forma que si fuera un dispositivo físico. Sin embargo la sintáxis empleada para montar el directorio remoto es diferente:

```
[root@mis01]# mount -t nfs hostname:/dirremoto /dirlocal
```

El comando mount intenta conectar con el demonio mountd en el host remoto via RPC. El servidor chequeará si el cliente tiene permitida la operación, y si es así le devolverá un manejador de fichero.

Cuando alguien accede a un fichero a través de NFS, el kernel coloca un allamada RPC en el demonio nfsd en la máquina servidora. Esta llamada toma el manejador de fichero, el nombre de fichero a ser accedido, y el uid y el gid del usuario como parámetros, que se utilizan para determinar los derechos de acceso.

En la mayoría de implementaciones UNIX, la funcionalidad NFS, tanto en el cliente como en el servidor se implementa como demonios (procesos) a nivel de Kernel que se arrancan desde el espacio de usuario en el boot del sistema. Estos son, el domonio nfsd en el host servidor y el demonio de bloqueo de E/S biod en el cliente.

El NFS de Linux es un poco diferente ya que el código del cliente está integrado en el nivel del Sistema de Ficheros Virtual (VFS) del kernel y no requiere control adicional a través de un demonio biod.

Actualmente existen dos implementaciones del servidor NFS bajo Linux. Una es el servidor NFS en el espacio del usuario y otra el servidor NFS en el espacio del Kernel.La del espacio del usuario tiene que copiar memoria extra entre el espacio del kernel y el espacio del usuario y además es una sobrecarga para el cambio de contexto. No se permiten bloqueos a nivel de registro ni de fichero. Mientras que el servidor NFS en el espacio del kernel no tiene que mover memoria entre los dos espacios ya que se ejecuta en el espacio del kernel y realiza las llamadas RPC dentro del kernel. Si que permite bloqueo de regisros y ficheros lo cual es importante cuando tienes un entorno heterogéneo, y además permite lanzar varias del demonio servidor.

RedHat Linux implementa por defecto el servidor NFS en el espacio del Kernel.

5.2. Configuración de un servidor nfs

Antes de utilizar NFS, ya sea como servidor o como cliente, el kernel debe tener compilado el soporte NFS. Se puede saber si el kernel está preparado para NFS consultando el sistema de ficheros virtual /proc.

```
[root@mis01]# cat /proc/filesystems
        ext2
nodev proc
        iso9660
nodev devpts
nodev nfs
```

En la realidad la salida anterior muestra todos los sistemas de ficheros soportados.

Para proporcionar servicios NFS se deben ejecutar los demonios rpc.mountd y rpc.nfsd. Como son programas basados en RPC, por tanto no son manejados por inetd (el internet daemon), tienen que lanzarse en tiempo de boot y registrarse con el portmap (como ocurría con los servicios de NIS), el cual deberá ejecutarse anteriormente.

El paquete necesario en RedHat Linux para convertir un sistema en un servidor nfs es nfs-utils que se corresponde con el servidor NFS en el espacio del kernel.

5.2.1. El archivo `/etc/exports`

Como se ha indicado anteriormente, un servidor NFS comparte (exporta) directorios a otras máquinas de la red. Luego debe de exisir algún mecanismo que nos permita realizar esta configuración.

El fichero /etc/exports especifica los directorios compartidos y el tipo de acceso permitido. La sintaxis del fichero es la siguiente:

```
FSAEXPORTAR [ NOMBREMAQ | DIRIP | METACHARS ] ( OPCIONES )
```

como se observa, primero se define el sistema de ficheros a exportar, seguido de las máquinas que van a tener acceso, y al final se define el tipo de acceso (de solo lectura, lectura y escritura etc ...)

Para definir la mquina se pueden utilizar metacaracteres. El demonio mountd busca el nombre del host cliente utilizando la llamada gethostbyaddr. Con DNS activado, esta llamada devuelve el nombre canónico, luego quiere decir que no se pueden utilizar alias. Sin DNS, el nombre devuelto es el primero encontrado en el fichero /etc/hosts que coincida con la dirección del cliente.

Entre las opciones que se pueden utilizar a la hora de exportar un directorio podemos encontrar las siguientes:

- INSECURE: permite accesos no autentificados para esa máqina.
- UNIX-RPC: requiere autenticación RPC unix-domain. Simplemente requiere que la autenticación sea hecha desde un puerto de red reservado (menor 1024). Esta opción está establecida por defecto.
- SECURE-RPC: requiere autenticación segura de RPC para esa máquina.
- KERBEROS: requiere autenticación Kerberos.
- ROOT-SQUASH: deniega accesos especiales al superusuario (root) del host especificado, mapeando peticiones del UID 0 en el cliente al UID 65534 en el servidor.
- NOROOTSQUASH: no mapea peticiones del UID 0. Esta es la opción por defecto.
- RO: exporta el directorio como de solo lectura.
- RW: exporta el directorio como lectura escritura.
- LINKRELATIVE: deja los enlaces simbólicos tal y como están.
- MAPIDENTITY: esta opción le indica al servidor que asuma que el cliente utiliza los mismos UID's y GID's que el servidor. Esta opción es por defecto.
- MAPDAEMON: le indica al servidor NFS que se asuma que el cliente y el servidor mo comparten el mismo espacio de UID/GID. Entonces NFS construirá una lista de mapeo de UID's entre el cliente y el servidor, consultando al demonio UGIDD del cliente.

Una configuración típica del fichero /etc/exports podría ser la siguiente:

```
/home mis*.dsic.upv.es(rw)
/cdrom mis*.dsic.upv.es(ro)
```

Donde estamos compartiendo el directorio /home del servidor para que sea accesible desde todos los clientes y en conjunción con las páginas amarillas,, los usuarios puedan tener el mismo entorno en cualquier máquina que entren. Por defecto al usuario root de cada máquina no se le permitirá tener privilegios de superusuario sobre el directorio /home del servidor.

Para que el servidor NFS tenga en cuanta estos cambios, habría que relanzar los demonios mountd y nfsd, pero existe una utilidad llamada exportfs que realiza este trabajo por nosotros.

5.2.2. Arranque y parada del servidor NFS

De nuevo podemos encontrar en el directorio /etc//init.d los scripts necesarios para lanzar los demonios del servidor NFS. Estos scripts son nfs que lanza los servicios rpc.mountd que

cuando recibe una petición de montaje de un cliente NFS chequea el fichero /etc/exports para comprobar si este está autorizado, devolviendo un manejador de fichero al cliente y rpc.nfsd vistos anteriormente y ademas se lanza otro servicio denominado rpc.rquotad que es un servidor RPC que devuelve la cuota de un usuario de un sistema de ficheros local que está montado a través de NFS.

Solo quedaría añadir el script a los diferentes niveles de ejecución donde queremos que se arranque o pare, utilizando linuxconf o el comando chkconfig.

5.3. Configuración del cliente nfs

Como se ha comentado, para que una máquina cliente pueda utilizar un sistema de ficheros nfs, este debe de haber sido compilado en el kernel estáticamente o como módulo.

La forma de que un cliente tiene de utilizar un sistema de ficheros exportado por un servidor NFS, es con el comando mount. Es decir los volúmenes NFS se montan de la misma forma que el resto de sistemas de ficheros:

```
[root@mis01]# mount -t nfs mis01:/home /home
```

Realmente el formato del comando tendría una apariencia como la siguiente:

```
mount -t nfs dirnfs dirlocal opciones
```

Entre las opciones que se le pueden pasar al comando mount se encuentran las siguientes:

- RSIZE=n , WSIZE=n Especifican el tamaño del datagrama utilizado por los clientes NFS cuando realizan peticiones de Lectura/Escritura.
- TIMEO=n Establece el tiempo en segundos que un cliente NFS esperará a que se complete una petición. El valor por defecto es 7.
- HARD Esta opción se refiere al tipo de montaje. En esta ocasión un montaje duro provoca que el programa que está accediendo a un fichero montado por NFS se colgará cuando el servidor falle. El proceso no puede ser interrumpido a menos que se indique la opción INTR. Cuando el servidor esté disponible, el programa continuará como si nada.
- SOFT Esta opción permite al kernel obtener un timeout si el servidor NFS no responde durante algún tiempo. El tiempo puede ser especificado con la opción timeo.
- INTR Colocar esta opción, permite a las señales de interrupción abortar el proceso cuando el servidor no responde.

Pero no sería lógico que el administrador tuviera que montar el directorio exportado por un servidor NFS, cada vez que un usuario lo necesite (ya que el comando mount solo se puede ejecutar con privilegios de superusuario. Debe de existir alguna manera de automatizar este proceso y que el sistema se encargue de montarlo en el arranque.

No existe ningún script que se pueda lanzar en determinado nivel de ejecución que alcance la máquina cada vez que arranque. Lo que se hace es tratar al sistema de ficheros nfs, como si fuera un sistema de ficheros más y añadir una entrada en el fichero /etc/fstab que se encarga de definir los sistemas de ficheros que se montaran en el arranque.

Una definición típica del /etc/fstab podría ser la siguiente.

```
/dev/hda4          /              ext2      defaults        1 1
/dev/hda1          /boot          ext2      defaults        1 2
/dev/hda3          /usr           ext2      defaults        1 2
mis01:/home        /home          nfs       defaults        1 2
```

6

El Servicio de información en red: NIS

Tabla de contenidos

6.1. Introducción

Cuando en una red existen varios sistema Linux que quieren acceder a recursos comunes, hay que asegurar que la lista de usuarios y grupos esté disponible en todas las máquinas clientes. La red debe de ser transparente para el usuario, da igual en la máquina que trabaje, siempre debe de encontrar el mismo entorno. Para conseguir es objetivo utilizaremos el Servicio de Información de Red (NIS).

Las siguientes líneas están sacadas del Sun(tm) System and Network Administration Manual:

```
"NIS se llamaba en un principio Sun Yellow Pages (YP)
 pero el nombre Yellow Pages(tm) es una marca registrada
 en el Reino Unido por la British Telecom plc y no
 puede ser usado sin permiso."
```

NIS viene de *Network Information Service*. Su propósito es proveer información, que tiene que ser conocida a lo largo de la red, a todas las máquinas de la red. La información más indicada para ser distribuida por NIS es:

- nombres de login/passwords/directorios home (/etc/passwd)
- información de grupos (/etc/group)

Así que, por ejemplo, si la entrada de tu password está grabada en la base de datos passwd de NIS, serás capaz de entrar en todas las máquinas de la red que tengan corriendo los programas clientes NIS.

6.2. Funcionamiento

En una red debe haber al menos una máquina actuando como un servidor NIS. Puedes tener múltiples servidores NIS, cada uno sirviendo a diferentes "dominios" NIS - o puedes tener servidores NIS cooperativos, donde uno es el llamado servidor NIS maestro, y todos los demás son los llamados servidores NIS esclavos (para un "dominio" NIS determinado), o puedes tener una mezcla de ellos.

Los servidores esclavos solo tienen copias de las bases de datos NIS y reciben estas copias del servidor NIS maestro cada vez que se realizan cambios a las bases de datos maestras. Dependiendo del número de máquinas que haya en tu red y de la seriedad de tu red, podrías decidir si instalar uno o más servidores esclavos. Cada vez que un servidor NIS se cae o va muy lento respondiendo peticiones, un cliente NIS conectado a ese servidor intentará encontrar otro que no esté caído o que vaya más rápido.

Las bases de datos NIS están en el formato DBM, que deriva de las bases de datos ASCII.

Por ejemplo, los ficheros /etc/passwd y /etc/group pueden ser directamente convertidos a formato DBM usando software de translación ASCII a DBM ("dbload", incluído con el software del servidor). El servidor NIS maestro debería tener ambas, las bases de datos ASCII y las DBM.

Los servidores esclavos serán notificados de cualquier cambio en los mapas NIS, (vía el programa "yppush"), y recibirán automáticamente los cambios necesarios para sincronizar sus bases de datos. Los clientes NIS no necesitan hacer esto ya que éstos siempre hablan directamente con el servidor NIS para leer la información almacenada en sus bases de datos DBM.

Los clientes NIS para Linux son capaces de obtener el servidor a partir de un fichero de configuración --lo que quiere decir que no es necesario un broadcast (lo cual es inseguro, debido al hecho de que cualquiera podría instalar un servidor NIS y contestar a las peticiones de broadcast...).

6.3. Configuración de NIS

Para ejecutar el software de servidor/cliente de NIS se necesitará ejecutar antes el programa *portmap*. En RedHat Linux existe un script en /etc//init.d/portmap para arrancar éste demonio. Todo lo que tienes que hacer es añadirlo a un runlevel si deseas que se lanze en el arranque..

El mapeador RPC es un servidor que convierte números de programas RPC en números de puerto de protocolo TCP/IP (o UDP/IP). Debe estar ejecutándose para poder realizar llamadas RPC (que es lo que el software de cliente NIS hace) a servidores RPC (como un servidor NIS) de esa máquina. Cuando un servidor RPC arranca, avisará al mapeador de puertos por cuál puerto está escuchando, y a qué números de programas RPC está preparado para servir. Cuando un cliente desea hacer una llamada RPC a un número de programa dado, primero deberá contactar con el mapeador de puertos de la máquina servidora para determinar el número de puerto al que los paquetes RPC deben ser enviados. El comando rpcinfo nos dará la información de que programas está preparado para servir.

El software necesario para poner en marcha y configurar un servidor o cliente de NIS en una distribución RedHat Linux, es el siguiente:

```
ypserv
yp-tools
ypbind
```

Normalmente los servidores RPC estándar son arrancados por *xinetd*, de modo que el mapeador de puertos (portmap) debe ser iniciado antes de que inetd sea invocado.

6.3.1. Configuración de un cliente NIS

Hay que asegurarse que tenemos instalados los paquete ypbind y yp-tools.

El programa /usr/sbin/ypbind es el servicio cliente de NIS, que nos permitirá interactuar con un servidor NIS y acceder a los mapas que este sirva (/etc/passwd, /etc/group). Para comprobar que ypbind funciona correctamente habría que seguir los siguientes pasos:

1. Tener arrancado el portmap.
2. Definir el dominio de NIS al cual sirve el servidor con el comando *domainname*.
3. Crear el directorio /var/yp sino existe.
4. Iniciar ypbind: */usr/sbin/ypbind*
5. Usar el comando "rpcinfo -p localhost" para comprobar si ypbind es capaz de registrar su servicio con el mapeador de puertos. El rpcinfo debería producir una salida parecida a:

```
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100007    2   udp    637  ypbind
100007    2   tcp    639  ypbind
300019    1   udp    660
```

6. Comprobar que puedes acceder a los mapas que sirve el servidor NIS: *ypcat passwd* Esto debería devolver el mapa passwd del servidor.

Si todo ha funcionado correctamente, deberías automatizar este proceso para que se realizara cada vez que arranque el cliente. En RedHat Linux existe un script llamado / etc/rc.d/init.d/ypbind que acepta {star|stop} para lanzar o parar el servicio. El dominio de NIS se define añadiendo la variable NISDOMAIN al fichero de configuración / etc/sysconfig/network.

6.3.2. El fichero `/etc/nsswitch.conf`

El fichero de Network Services Switch /etc/nsswitch.conf determina el orden de las búsquedas realizadas cuando se pide una pieza específica de información, de la misma forma que el fichero /etc/host.conf determina la manera en que se realizan las búsquedas de hosts. Por ejemplo, la línea

```
hosts: files nis dns
```

especifica que las funciones de búsqueda de host deben primero mirar en el fichero /etc/hosts local, seguido de una búsqueda NIS y, finalmente, usar el DNS (/etc/resolv.conf y named). Si al llegar a este punto no se encuentra el host correspondiente se devuelve un error.

6.3.3. Configuración de un servidor NIS

Habrá que comprobar que está instalado el paquete ypserv (rpm -q ypserv), que es el que nos permitirá instalar un servidor de páginas amarillas o NIS.

Existen dos tipos de servidores NIS: maestros y esclavos. El primero será el servidor principal donde se generarán los mapas que tenemos que servir, mientras que un servidor esclavo mantendrá una copia de los mapas por si no estuviera disponible en algún momento el servidor maestro.

Habrá que editar el fichero /etc/ypserv.conf que nos permitirá configurar algunas opciones para el sevidor ypserv. Normalmnete contiene una lista de reglas para hosts especiales y reglas de acceso a los mapas. El fichero /var/yp/securenets define los derechos de acceso al servidor NIS por parte de los clientes. El fichero contiene pares netmask/network de forma que si la dirección IP del cliente se corresponde con la entrada podrá actuar como cliente de NIS.

Hay que asegurarse que el portmap está corriendo (rpcinfo -p localhost) y entonces y solo entonces lanzar el demonio ypserv. El siguiente paso sería generar las bases de datos (mapas) de NIS. En el servidor maestro habrá que ejecutar lo siguiente:

```
[root@mis01]# /usr/lib/yp/ypinit -m
```

De esta forma se definirán los servidores esclavos y los mapas que sirve el servidor de NIS. Si se necesita actualizar un mapa, se tiene que ejecutar *make* en directorio /var/yp del servidor maestro de NIS. Esta acción actualizará el mapa si el fichero fuente es mas nuevo, y colocará los ficheros en los servidores esclavos.

En el servidor esclavo (puede haber más de uno) se tendrán que realizar los siguientes pasos. Asegurarse que el portmap está corriendo, ejecutar el demonio ypserv, y entonces llamar al siguiente comando:

```
[root@mis02]# /usr/lib/yp/ypinit -s mis01
```

Habrá que asegurarse de que el nuevo servidor esclavo tiene una entrada en el fichero de configuración /etc/ypservers del servidor maestro.

También sería una buena idea editar el fichero crontab de root del servidor esclavo y añadir las sigueintes lineas:

```
20 *      * * *         /usr/lib/yp/ypxfr_1perhour
40 6      * * *         /usr/lib/yp/ypxfr_1perday
55 6,18   * * *         /usr/lib/yp/ypxfr_2perday
```

Esto asegurará que la mayoría de los mapas NIS estén actualizados, incluso si una actualización se perdió debido a que el servidor esclavo estaba fuera de servicio en el momento en que el servidor maestro actualizó los mapas.

Si quisiéramos restringir el acceso de los usuarios a nuestro servidor NIS, tendríamos que configurarlo también como un cliente NIS y añadir las entradas necesarias en el /etc/passwd.

El último paso sería automatizar el proceso de poner en marcha el servidor NIS cada vez que arranque la máquina. En RedHat Linux existe el script /etc//init.d/ypserv que acepta los parámetros {start|stop} para lanzar o parar el servicio. Colcándolo en los niveles de ejecución apropiados tendremos listo este proceso.

Habría que considerar que con este método de autentificación en red, si los usuarios quisieran cambiarse sus passwords, el comando normalmente utilizado passwd no cumplirá su función ya que debería actualizar los mapas NIS del servidor pertinentes. Para ello en el servidor NIS debe estar corriendo el demonio rpc.yppasswdd, el cual maneja los cambios de passwords y asegura que la información de los mapas de NIS se actualiza correctamente. En los clientes deberíamos utilizar el comando yppasswd en vez de passwd.

7

El núcleo de Linux

Tabla de contenidos

7.1. Introducción

El núcleo de un sistema operativo es el programa que gobierna el hardware del ordenador y ofrece una interfaz de servicios al resto de programas (procesos) que se ejecutan en dicho ordenador. Normalmente, el núcleo es el primer código que se carga en memoria cuando encendemos el ordenador y permanece en dicha memoria mientras el sistema está en funcionamiento, entrando en ejecución cada vez que un proceso requiere uno de sus servicios o un dispositivo físico requiere su atención. Una de las diferencias fundamentales entre el núcleo del sistema y los procesos que se ejecutan "por encima" del mismo es que uno y otros se ejecutan en *modos de procesador* distintos (siempre que el hardware del ordenador soporte esta característica). El núcleo suele ejecutarse en alguno de los modos privilegiados del procesador, en el que tiene completo acceso a todo el hardware (incluyendo la programación de los dispositivos), mientras que los procesos se ejecutan en "modo usuario", en el que no pueden ejecutar un buen número de de instrucciones máquina del procesador. De esta forma, el núcleo se convierte en una especie de *intermediario* obligado en la mayoría de las acciones que los procesos ejecutan, lo que le permite implementar de forma adecuada la multitarea, repartiendo de forma equitativa y segura los recursos del sistema entre los diferentes procesos que se ejecutan en cada momento.

En el caso concreto de Linux, el tema del núcleo puede inducir a error, dado que se suele denominar Linux al sistema operativo completo (incluyendo todos los programas y utilidades que podemos ejecutar cuando instalamos el sistema) además de al núcleo en sí mismo. En otros sistemas operativos comerciales (como por ejemplo Windows 2000 o Solaris) esta distinción no es necesaria, puesto que ambos (el núcleo y dicho conjunto de utilidades y programas que se distribuyen junto con él) provienen de la misma empresa (en el ejemplo, Microsoft y SUN Microsystems, respectivamente) y se distribuyen conjuntamente. Sin embargo, en el caso de Linux esto no es así. El núcleo propiamente dicho es desarrollado por un grupo de personas (lideradas por el creador de Linux, Linus Torvalds) de forma *independiente* del resto de utilidades, herramientas y aplicaciones que tenemos disponibles cuando instalamos Linux. A este conjunto de utilidades (más un núcleo concreto) se lo conoce como una *distribución* de Linux, y existen numerosas empresas, organizaciones e incluso individuos que mantienen distintas distribuciones: Red Hat, Fedora, Debian, SuSe, Mandrake, Caldera, Gento, etc. Puesto que el núcleo se distribuye líbremente como código GPL, en realidad cualquiera de dichas organizaciones puede realizar modificaciones en el núcleo (y algunas lo hacen), pero Linus y su equipo siguen manteniendo el control sobre el núcleo reconocido.

De esta manera, las distribuciones y los núcleos tienen versiones distintas e independientes. Por ejemplo, en estos momentos la versión actual de Fedora Linux se denomina "Core 1", y el núcleo que instala por defecto es el 2.4.22. La nomenclatura de la versión del núcleo es significativa del estado del mismo: el segundo número indica si la versión es estable (número par) o de desarrollo (número impar), mientras que el último dígito indica un número de orden (creciente) de la versión. Un número mayor indica un núcleo más reciente y (probablemente) con mayor y mejor soporte.

En la mayoría de situaciones, el núcleo pasa desapercibido al usuario común (incluso al administrador), puesto que los usuarios interactúan con los "programas" (o más correctamen-

te con los *procesos*), que internamente interactúan con el núcleo. Sin embargo, existen situaciones en las que necesitamos *modificar* el soporte ofrecido por el núcleo, para conseguir mejores prestaciones, un sistema más seguro o mejor adaptado a un uso concreto, o simplemente para poder acceder a un nuevo tipo de dispositivo físico. Este capítulo explica los pasos necesarios para poder modificar el soporte del núcleo de Linux.

Los siguientes apartados explican los conocimientos mínimos imprescindibles sobre el núcleo de Linux (los ficheros binarios involucrados, el concepto de módulo de núcleo, donde encontrar los fuentes, etc.). Finalmente, se explica cómo obtener un núcleo nuevo, mediante la compilación de una versión en código fuente y su instalación en el sistema.

7.2. El código binario del núcleo

El código ejecutable del núcleo de Linux (el que se carga en memoria al arrancar el ordenador) reside en un fichero denominado `vmlinux` (o en su homólogo comprimido, `vmlinuz`). Normalmente, este fichero se encuentra en el directorio `/boot/` y suele tener como sufijo la versión del núcleo que generó el ejecutable (por ejemplo, `/boot/vmlinuz-2.4.22-1`). Cuando instalamos una distribución de Linux, el fichero binario del núcleo que incorpora la distribución es instalado en ese directorio y el cargador del sistema operativo (habitualmente, LILO o GRUB) lo carga en memoria y le ofrece el control al encender el ordenador. Este fichero binario es por tanto imprescindible para el funcionamiento del sistema. Por otra parte, si instalamos un binario nuevo (de una versión más moderna, por ejemplo), debemos reiniciar el ordenador para que vuelva a ser cargado en memoria. A diferencia de otros sistemas operativos, esta viene a ser la única situación en Linux en la que instalar un nuevo software obliga a reiniciar ("...para que los cambios tengan efecto").

En realidad, el código binario del núcleo puede dividirse entre diferentes ficheros: el fichero binario principal `vmlinuz` mencionado arriba (que es absolutamente necesario) y, opcionalmente, un conjunto de ficheros denominados *módulos de núcleo* (*kernel modules*). Los módulos de núcleo son ficheros objeto (compilados de forma especial) que incorporan funcionalidades concretas que pueden ser instaladas y desinstaladas del núcleo de Linux sin tener que generar un nuevo ejecutable `vmlinuz` e incluso sin tener que reiniciar el ordenador. Normalmente, la misma funcionalidad puede ser incorporada al núcleo (en fase de compilación) como parte del binario principal o como un módulo de núcleo.

Un buen ejemplo del uso de estos módulos es la instalación de manejadores (*drivers*) de dispositivos. Suponga que acabamos de instalar una nueva tarjeta de red en nuestro ordenador. Si el manejador de dicha tarjeta no está integrado en el binario principal del núcleo (`vmlinuz`), pero sí disponemos de él como un módulo de núcleo, podemos simplemente "cargarlo" en el núcleo mediante una orden de shell, sin tener que recompilar el `vmlinuz` y reiniciar el sistema. Los módulos de núcleo se encuentran disponibles en Linux desde la versión 1.2, y resultan una herramienta muy útil para multitud de situaciones, en especial para poder ofrecer un núcleo con un amplio soporte de manejadores, sistemas de archivos, protocolos de red, etc., sin tener que incluirlos necesariamente en el binario principal.

De hecho, el soporte de módulos está tan bien integrado en el sistema que también resulta transparente para los usuarios en la mayoría de ocasiones. Por ejemplo, si montamos por pri-

mera vez una partición de tipo FAT32 en un sistema en el que dicho sistema de archivos está soportado por módulos de núcleo, Linux cargará automáticamente dichos módulos y luego montará la partición en el directorio correspondiente, sin ninguna intervención al respecto del usuario (ni del administrador).

Hoy en día, distribuciones como Fedora incorporan un núcleo con un soporte razonable en el binario principal y un número muy elevado de módulos de núcleo que cubren buena parte del soporte extra que podemos necesitar en el sistema. Sin embargo, existen ocasiones en las que necesitamos conseguir un soporte distinto. A continuación se enumeran algunas de ellas:

- Núcleo más pequeño. Algunos sistemas necesitan ejecutarse con poca memoria principal. Puesto que el binario principal del núcleo nunca se desaloja de memoria, en ocasiones necesitamos eliminar de dicho binario código que nuestro sistema no necesita y así conseguir más memoria para los procesos de usuario.

- Núcleo más seguro. Otra razón para eliminar código del núcleo que no necesitamos es incrementar su robustez y seguridad. Mantener más código (que no se usa) eleva la probabilidad de que nuestro sistema presente vulnerabilidades ante ataques que comprometan la seguridad del mismo.

- Núcleo más rápido. Un núcleo completamente "monolítico" (sólo el binario principal y sin módulos de núcleo) es más rápido que otro que tenga su soporte repartido entre ambos. En instalaciones en las que el hardware es lento o la velocidad del núcleo es crítica, es recomendable conseguir un núcleo monolítico (y optimizado). De esta forma conseguimos un núcleo altamente especializado, que probablemente sólo funcionará bien en el ordenador para el que ha sido compilado (u otros con idéntico hardware).

- Núcleo más portable. El caso contrario al anterior es conseguir un núcleo que funcione correctamente en el mayor número posible de ordenadores. Esto se consigue mediante un binario principal mínimo y el máximo número posible de módulos de núcleo.

7.3. El código fuente del núcleo

Como hemos visto en la sección anterior, existen ocasiones en las que es conveniente o necesario obtener un nuevo núcleo (binario principal más módulos).

Puesto que la mayoría de distribuciones incorporan los binarios de varios núcleos como ficheros preparados para instalar (por ejemplo, en formato rpm en el caso de Fedora), si sólo queremos obtener un núcleo más moderno para nuestra instalación de Linux lo más sencillo es obtener el último de dichos paquetes e instalarlo en el sistema.

Sin embargo, si desamos tener un núcleo para el que aún no hay versión binaria instalable, o simplemente deseamos cambiar el tipo de soporte del núcleo que tenemos actualmente instalado, necesitamos obtener el *código fuente* de dicho núcleo y recompilarlo.

Es posible que la distribución que estamos utilizando posea también como paquetes instalables los fuentes de varios núcleos (normalmente sólo de versiones estables). Si nos interesa uno de dichos núcleos, tendremos que obtener e instalar el paquete correspondiente, lo que normalmente instalará el código fuente bajo el directorio `/usr/src/`, en un directorio denominado `linux-(versión)/` (por ejemplo, `/usr/src/linux-2.4.22-1`).

Por el contrario, si el núcleo que buscamos no lo tenemos disponible como paquete instalable en nuestra distribución, tendremos que acudir a los lugares de internet que mantienen en línea los distintos núcleos disponibles de Linux, como por ejemplo *The Linux kernel archives* [http://www.kernel.org].

Si optamos por esta última opción, tenemos que considerar que el código fuente se distribuye normalmente en dos formatos distintos: como un fichero en formato tar comprimido que contiene todos los ficheros fuentes del núcleo, o bien como un "parche" que contiene sólo las *diferencias* respecto a otra versión anterior. Si no se es un usuario experto en aplicar parches a código fuente (se realiza mediante la orden **patch**) se recomienda la primera opción, porque aunque lógicamente el fichero a descargarse es mucho mayor, su instalación resulta mucho más sencilla. Una vez descargado el fichero, se descomprime en un directorio (`/usr/src/` es una opción razonable) y ya estamos en condiciones de comenzar la compilación del nuevo núcleo.

7.4. Compilación e instalación de un nuevo núcleo

Suponiendo que disponemos del código fuente de un núcleo de Linux bajo el directorio `/usr/src/linux/`, a continuación se listan las acciones que deben realizarse para construir un nuevo núcleo binario e instalarlo en el sistema.

1. Eliminar cualquier resto de ficheros de compilación en el núcleo, para recompilarlo desde cero.

```
[root@yoda root]# cd /usr/src/linux
[root@yoda root]# make mrproper
```

2. Configurar el nuevo núcleo. Existen tres herramientas de configuración alternativas: una basada en texto puro (**make config**), otra en un entorno de ventanas simulado en modo texto y (**make menuconfig**) finalmente otra en modo gráfico (**make xconfig**). Si utilizamos ésta última:

```
[root@yoda root]# make xconfig
```

   obtenemos una ventana que ejecuta un script escrito en Tcl/Tk que visualiza un menú como el que aparece en la Figura 7.1, “El menú principal de configuración del núcleo.”

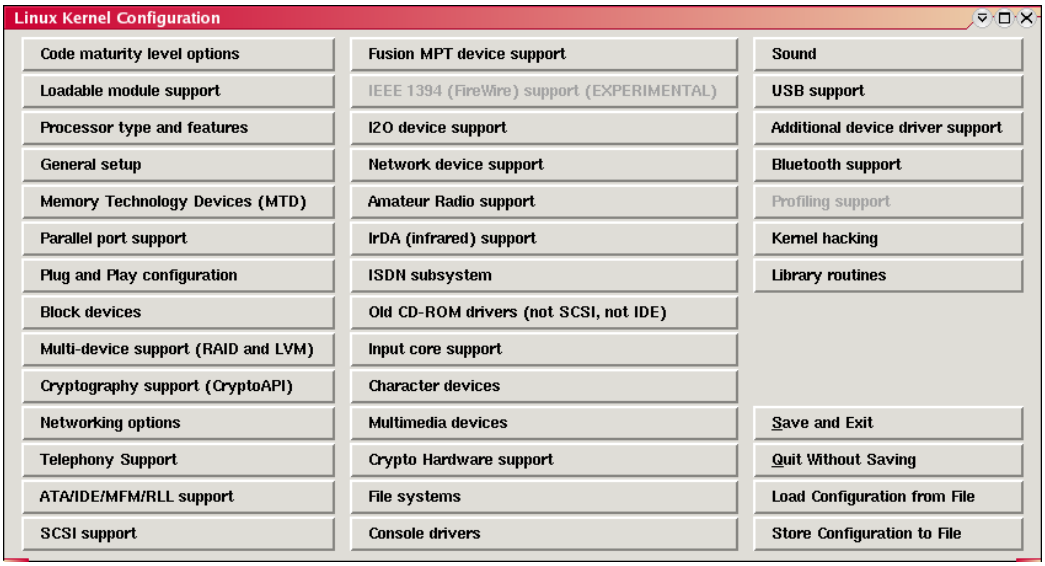

Figura 7.1. El menú principal de configuración del núcleo.

Como vemos, la ventana muestra multitud de botones, que ocultan a su vez menús de configuración organizados temáticamente. Al pulsar cada botón nos aparece el menú correspondiente, en donde configuraremos el soporte que deseamos para nuestro nuevo núcleo. La Figura 7.2, “Uno de los submenús de configuración del núcleo.” muestra, como ejemplo, el menú de configuración del tipo de procesador para el que se compilará el núcleo y sus características principales.

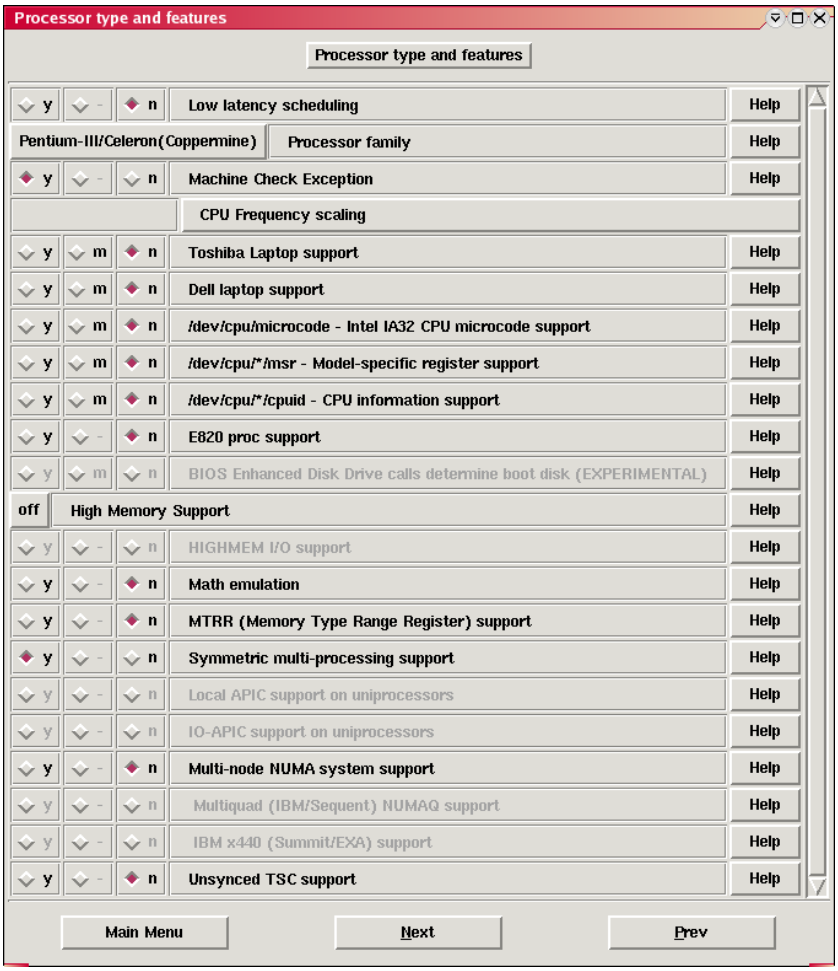

Figura 7.2. Uno de los submenús de configuración del núcleo.

En esta figura aparecen buena parte de las alternativas de configuración que podemos encontrarnos en todos los submenús: las opciones que aparecen en gris están habilitadas, porque son incompatibles con otras que hemos escogido (posiblemente en otros submenús). Las opciones que aparecen en negrita (opciones selecionables) pueden ser de tres tipos fundamentalmente:

- Opciones "sí/módulo/no". En este tipo de opciones podemos elegir entre tres alternativas: `y`, indicando que queremos incluir ese soporte en el binario principal del nuevo núcleo; `m`, indicando que queremos el soporte pero como módulo de núcleo; y finalmente `n`, indicando que no queremos ese soporte. Como vemos en la figura, en algunas de las opciones no existe la posibilidad de compilarla como módulo.

- Opciones múltiples. En estas opciones podemos elegir entre un conjunto de alternativas. Por ejemplo, en el tipo de procesador podemos elegir entre 16 tipos distintos.

El menú con dichos tipos aparece al pulsar el botón correspondiente del submenú.

- Valor numérico. Existen opciones que permiten introducir un valor numérico concreto, que luego se pasará a los ficheros fuente involucrados como una constante de preproceso.

Cuando hemos acabado de configurar todas las opciones del núcleo que necesitamos para nuestro nuevo núcleo, hemos de pulsar el botón de salvar los cambios (botón "`Save and Exit`" en la Figura 7.1, “El menú principal de configuración del núcleo.”) y salir de la herramienta de configuración.

3. Comprobar el número de versión. Antes de iniciar la compilación, es conveniente asegurarse de que el núcleo tendrá un número de versión distinto del que tenemos instalado actualmente, puesto que mientras no nos aseguremos que el nuevo núcleo funciona correctamente no es buena idea deshacerse (o sobreescribir) el antiguo. Para ello editaremos el fichero `/usr/src/linux/Makefile` y observaremos sus primeras líneas, donde se muestra la información sobre el número de versión:

```
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 20
EXTRAVERSION = -20.9custom
```

Por ejemplo, con esta información se generaría un núcleo cuya versión sería `2.4.20-20.9custom`. En el caso de que coincidiera con un núcleo actualmente instalado y que no queremos sobreescribir, deberíamos modificar *sólo* el campo `EXTRAVERSION`. De hecho, el `Makefile` ya nos propone un sufijo denominado "`custom`" que no se encuentra en ningún núcleo "oficial".

Este número de versión (en el ejemplo, `2.4.20-20.9custom`) será el sufijo de todos los ficheros del directorio `/boot/` que dependen del núcleo concreto (`vmlinuz`, `initrd`, `System.map`, `config` y `module-info`), así como el nombre del directorio donde residirán los módulos de núcleo, debajo del directorio `/lib/modules/`.

4. Compilar el núcleo. Para compilar el binario principal y los módulos del núcleo hay que ejecutar las siguientes órdenes:

```
[root@yoda root]# make dep
[root@yoda root]# make bzImage
[root@yoda root]# make modules
```

5. Instalar el núcleo. Los ficheros binarios del nuevo núcleo (binario principal y módulos) residen todavía en ciertos subdirectorios de `/usr/src/linux/` y no en los lugares

definitivos donde el sistema podrá encontrarlos. Para llevar a cabo esta acción hay que ejecutar:

```
[root@yoda root]# make modules_install
[root@yoda root]# make install
```

que, siguiendo el ejemplo anterior, instalará los módulos de núcleo en el directorio denominado `/lib/modules/2.4.20-20.9custom/` y creará los siguientes ficheros y enlaces simbólicos en el directorio `/boot/`:

```
config-2.4.20-20.9
initrd-2.4.20-20.9.img
module-info-2.4.20-20.9
System.map-2.4.20-20.9
vmlinux-2.4.20-20.9
vmlinuz-2.4.20-20.9
module-info -> module-info-2.4.20-20.9
vmlinuz -> vmlinuz-2.4.20-20.9
System.map -> System.map-2.4.20-20.9
```

Además de realizar estas acciones, la instalación también realiza la configuración necesaria para que nuestro gestor de arranque (lilo o grub) sepa de la existencia del nuevo núcleo para que podamos arrancarlo en el próximo reinicio.

6. Reiniciar. Para que el núcleo nuevo sea cargado y ejecutado debemos reiniciar el sistema y asegurarnos de que en la lista de arranque de lilo o grub elegimos este núcleo y no el antiguo. En el caso de que algo salga mal (el núcleo no arranca, ciertos servicios no funcionan, etc.), deberemos reiniciar el sistema con el núcleo antiguo y probablemente volver al punto 2 para revisar la combinación de opciones con las que hemos configurado el nuevo núcleo.

7. Eliminar los ficheros objeto. Una vez el núcleo nuevo funciona correctamente y posee todas las funcionalidades que queríamos de él, podemos eliminar los objetos del directorio `/usr/src/linux/`, puesto que no los necesitamos más. Para ello ejecutaremos lo siguiente:

```
[root@yoda root]# cd /usr/src/linux
[root@yoda root]# make clean
```

La diferencia entre esta orden y la del punto 1 es que en este caso no se borran los ficheros de configuración del núcleo, con lo que si volvemos a ejecutar **make xconfig** tendremos en pantalla las mismas opciones con las que compilamos el núcleo la última vez.

Como alternativa, también podemos guardarnos esta configuración en un fichero,

para poder recuperarla más tarde. Esto se realiza mediante los botones "`Store Configuration to File`" y "`Load Configuration from File`" de la ventana principal de configuración (ver Figura 7.1, “El menú principal de configuración del núcleo.”). Si elegimos esta opción (guardarnos el fichero de configuración) podemos incluso borrar el directorio completo de los fuentes, puesto que siempre podremos volver a obtenerlo más tarde si es necesario.

8

El sistema de archivos Proc

Tabla de contenidos

8.1. Introducción

En la mayoría de sistemas operativos suele ser habitual disponer de herramientas que permiten conocer el estado interno del núcleo desde nivel de usuario. Normalmente, existen diferentes herramientas para cubrir distintos aspectos de dicho estado (procesos e hilos, memoria, ficheros, interrupciones y dispositivos, etc.) y, en función del sistema operativo, la información que puede obtenerse es más o menos completa. Cualquier distribución de Linux incorpora herramientas de este tipo, tales como:

- **ps**. Visualiza los procesos en memoria el momento actual. En función de los modificadores, puede mostrar sólo los procesos lanzados en el *shell* actual o todos los procesos del sistema y, para cada uno, más o menos información. Para mostrar el máximo de información, puede utilizarse: "**ps aux**"

- **mount**. Invocado sin argumentos, muestra una lista de las particiones (locales o remotas) montadas actualmente en el sistema.

- **free**. Muestra información acerca de la memoria ocupada y libre del sistema, tanto la física como la virtual (ubicada en las particiones "*swap*").

- **uptime**. Muestra cuánto tiempo ha estado funcionando el sistema (desde el último reinicio), así como el número de usuarios conectados actualmente y la carga media del procesador en los últimos 1, 5 y 15 minutos.

A diferencia de muchos de esos sistemas operativos, Linux incorpora además una forma totalmente distinta de conocer, e incluso modificar, el estado interno del núcleo y de los procesos de usuario, mediante una interfaz de sistema de archivos denominado "`proc`". Este capítulo comenta las características fundamentales de este "sistema de archivos" y cómo podemos utilizarlo para interrogar el estado (y modificar el comportamiento) del núcleo de Linux en cualquier momento.

8.2. El sistema de archivos `/proc`

Tal como muestra el título, el sistema de archivos "proc" se monta en el directorio `/proc/` y, si listamos su contenido, veremos que se encuentra lleno de archivos y directorios. Por ejemplo:

```
[root@yoda root]# ls -Cp /proc/
1/      24693/  5039/  5284/  5371/  apm          locks
10/     24696/  5048/  5289/  5373/  bus/         mdstat
11/     24697/  5113/  5303/  5385/  cmdline      meminfo
15/     24699/  5122/  5311/  5395/  cpuinfo      misc
2/      24701/  5126/  5313/  5398/  devices      modules
20554/  3/      5144/  5314/  5412/  dma          mounts@
22394/  4/      5152/  5316/  5642/  driver/      mtrr
22397/  4419/   5153/  5318/  5674/  execdomains  net/
24493/  4420/   5154/  5320/  5675/  fb           partitions
24507/  4743/   5155/  5325/  5707/  filesystems  pci
24556/  4865/   5156/  5326/  5813/  fs/          scsi/
24565/  4869/   5157/  5328/  5814/  ide/         self@
24620/  4887/   5158/  5329/  6/     interrupts   slabinfo
24653/  4906/   5170/  5331/  6912/  iomem        stat
24654/  4965/   5171/  5332/  6915/  ioports      swaps
24657/  4966/   5190/  5333/  7/     irq/         sys/
24659/  4978/   5233/  5335/  72/    kcore        sysvipc/
24685/  5/      5276/  5336/  7775/  kmsg         tty/
24688/  5015/   5279/  5348/  8/     ksyms        uptime
24689/  5029/   5282/  5367/  9/     loadavg      version
```

Sin embargo, este contenido no se encuentra guardado en ningún dispositivo físico (disco, disquete, cd, etc.), sino que es *construido* y presentado dinámicamente cada vez que le pedimos al núcleo que lo muestre, y lo mismo ocurre cuando visualizamos el contenido de sus archivos y subdirectorios. Este tipo de sistema de archivos se denomina sistema de archivos *virtual*. Si listamos de nuevo este directorio en otro momento, o en otro sistema, es posible que el listado no coincida exactamente con el del ejemplo, aunque algunos ficheros siempre serán listados; de igual forma, es posible que el *contenido* de los archivos y directorios también difiera. Ello es debido a que el contenido del directorio refleja el estado actual del núcleo de Linux, y evidentemente este estado varía con el tiempo y de un sistema a otro (por ejemplo, por disponer de hardware distinto).

Más formalmente, podemos definir `/proc/` como una *interfaz* entre el núcleo de Linux y el nivel de usuario con la forma de un sistema de archivos virtual. Así, el núcleo de Linux presenta una manera *única y homogénea* de presentar su información interna y se convierte en la alternativa a utilizar múltiples herramientas particulares como las que citábamos en el apartado anterior. La ventaja principal es que, aunque Linux incorpora muchas de ellas (en algunos casos, por compatibilidad con otras versiones de UNIX), no es imprescindible disponer de una herramienta por cada información que el núcleo pone a disposición del usuario, puesto que toda esa información está en `/proc`. Esto es especialmente útil en el caso de Linux, puesto que como veíamos en el Capítulo 7, *El núcleo de Linux*, el desarrollo del núcleo es independiente del desarrollo de las herramientas y programas que se ejecutan por encima (lo que se denomina "distribución").

Por ejemplo, veamos la información del mandato **mount**, seguida de la consulta del fichero correspondiente de `/proc/`:

```
[root@yoda root]# mount
/dev/hdb1 on / type ext2 (rw)
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/hdb3 on /mnt/backup type ext2 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
none on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
[root@yoda root]#
[root@yoda root]# cat /proc/mounts
/dev/root / ext2 rw 0 0
/proc /proc proc rw 0 0
usbdevfs /proc/bus/usb usbdevfs rw 0 0
/dev/hdb3 /mnt/backup ext2 rw 0 0
none /dev/pts devpts rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
```

Como vemos, el resultado es el mismo (probablemente **mount** interroga internamente a `/proc/mounts`), aunque normalmente el mandato muestra la información de manera algo más elaborada para el usuario. En muchos casos, la información que muestran los archivos de `/proc/` suele ser poco autoexplicativa, y es conveniente conocer de antemano el significado de los diferentes campos del "archivo" para interpretar correctamente lo que está comunicándonos el núcleo. Sin embargo, esto no es problema porque existe buena documentación al respecto, como por ejemplo la documentación de Red Hat Linux, en concreto la *"Reference Guide" de Red Hat 9* [https://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/ch-proc.html].

Los dos siguientes apartados enseñan, respectivamente, cómo obtener y modificar la información del núcleo de Linux interactuando con el sistema de archivos `/proc/`.

8.3. Obtener información del núcleo

A pesar de que el sistema de archivos `proc` es virtual y la información que contiene se construye cada vez que el usuario realiza una operación de lectura, el nombre de los archivos y directorios que podemos encontrar en el directorio `/proc/` está predefinido por el núcleo y es bien conocido.

Esta sección presenta un repaso de los archivos y directorios más relevantes, junto con instrucciones para interpretar su contenido. Para una descripción más exhaustiva, se recomienda documentación especializada como la "*Reference Guide*" de Red Hat 9 que mencionábamos en el apartado anterior, o la propia página de manual de "proc" (**man proc**).

Algunos de los archivos más importantes son los siguientes:

1. `/proc/cmdline`. Muestra los parámetros con los que se inició el núcleo en tiempo de arranque. Por ejemplo:

```
auto BOOT_IMAGE=linux ro BOOT_FILE=/boot/vmlinuz-2.4.20-20.8 hdc=ide-scsi
root=LABEL=/
```

2. `/proc/cpuinfo`. Identifica el tipo de procesador del sistema y sus parámetros. Por ejemplo:

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 7
model name      : Pentium III (Katmai)
stepping        : 3
cpu MHz         : 498.858
cache size      : 512 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 sep mtrr pge mca cmov
pat pse36 mmx fxsr sse
bogomips        : 996.14
```

3. `/proc/filesystems`. Muestra los sistemas de ficheros que actualmente soporta el núcleo. Es necesario disponer del soporte de sistema de ficheros con el que está formateada una partición (o disquete, cd, etc.) para poder montarlo y así acceder a su contenido. Por ejemplo:

```
nodev   rootfs
nodev   bdev
nodev   proc
nodev   sockfs
nodev   tmpfs
nodev   shm
nodev   pipefs
        ext2
nodev   ramfs
        iso9660
nodev   devpts
nodev   usbdevfs
nodev   usbfs
nodev   autofs
nodev   binfmt_misc
        vfat
```

La primera columna indica si actualmente existe alguna partición montada con dicho sistema de archivos (vacío) o no (`nodev`). La segunda columna indica el tipo de sistema de archivos. Si el núcleo soporta otros sistemas de archivos como módulos que aún no se han cargado, estos no aparecen en la lista (hasta que son cargados, obviamente).

4. `/proc/meminfo`. Muestra información sobre el uso actual de la memoria RAM, indicando la cantidad de memoria utilizada en varios conceptos (compartida, en "buffers" de entrada/salida, libre, intercambio, etc.). Veamos un ejemplo:

```
        total:    used:    free:  shared: buffers:  cached:
Mem:  393965568 203403264 190562304        0  5488640 111288320
Swap: 526409728         0 526409728
MemTotal:       384732 kB
MemFree:        186096 kB
MemShared:           0 kB
Buffers:          5360 kB
Cached:         108680 kB
SwapCached:          0 kB
Active:         175636 kB
ActiveAnon:      65804 kB
ActiveCache:    109832 kB
Inact_dirty:       452 kB
Inact_laundry:       0 kB
Inact_clean:      3756 kB
Inact_target:    35968 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:       384732 kB
LowFree:        186096 kB
SwapTotal:      514072 kB
SwapFree:       514072 kB
```

Las dos primeras líneas muestran un resumen del estado de la memoria RAM y de la memoria de intercambio (o *swap*), con cantidades expresadas en bytes. El resto de líneas muestran la misma información de manera más detallada y expresada en Kbytes. Los campos más relevantes son `MemTotal` (memoria física total), `MemFree` (memoria física no utilizada actualmente), `SwapTotal` (total de memoria de intercambio) y `SwapFree` (cantidad de memoria de intercambio no utilizada actualmente).

5. `/proc/modules`. Muestra un listado de los módulos de núcleo presentes en el sistema (cargados actualmente o no). Este listado puede variar sensiblemente de un sistema a otro (en un núcleo monolítico puro, estaría vacío). Por ejemplo:

```
nls_iso8859-1           3516   1 (autoclean)
nls_cp437               5148   1 (autoclean)
vfat                   13100   1 (autoclean)
fat                    38776   0 (autoclean) [vfat]
sr_mod                 18200   0 (autoclean)
ymfpci                 45516   0 (autoclean)
ac97_codec             14600   0 (autoclean) [ymfpci]
uart401                 8420   0 (autoclean) [ymfpci]
sound                  74196   0 (autoclean) [uart401]
```

```
soundcore               6500   4 (autoclean) [ymfpci sound]
binfmt_misc             7464   1
autofs                 13332   0 (autoclean) (unused)
ipt_REJECT              4024   2 (autoclean)
iptable_filter          2412   1 (autoclean)
ip_tables              15096   2 [ipt_REJECT iptable_filter]
ide-scsi               12176   0
scsi_mod              107608   2 [sr_mod ide-scsi]
ide-cd                 35808   0
cdrom                  33728   0 [sr_mod ide-cd]
CDCEther               14492   1
acm                     7904   0 (unused)
mousedev                5588   1
keybdev                 2976   0 (unused)
hid                    22340   0 (unused)
input                   5920   0 [mousedev keybdev hid]
usb-uhci               26412   0 (unused)
usbcore                78944   1 [CDCEther acm hid usb-uhci]
```

La primera columna muestra el nombre del módulo, la segunda su tamaño en memoria (en bytes), la tercera si está actualmene cargado (1) o no (0) y la cuarta y la quinta muestran información sobre si está en uso por otros módulos o no y, en su caso, cuáles son estos módulos).

6. `/proc/mounts`. Proporciona información sobre los sistemas de archivos actualmente montados en el sistema (ver ejemplo en sección anterior).

7. `/proc/swaps`. Muestra información sobre el estado de la memoria de intercambio, pero subdividido por particiones de intercambio. Conocer esta información puede ser interesante si el sistema dispone de múltiples particiones de este tipo.

8. `/proc/uptime`. Ofrece el tiempo (en segundos) desde que el sistema se inició, y la cantidad de este tipo en que el procesador ha estado ocioso. Por ejemplo:

```
3131.40 2835.67
```

9. `/proc/version`. Muestra la versión del núcleo y la del compilador de C (gcc). En el caso de Red Hat Linux, muestra también la versión de Red Hat instalada:

```
Linux version 2.4.20-20.8 (bhcompile@daffy.perf.redhat.com)
(gcc version 3.2 20020903 (Red Hat Linux 8.0 3.2-7))
#1 Mon Aug 18 14:59:07 EDT 2003
```

En cuanto a los subdirectorios más relevantes de `/proc/`, podríamos destacar los siguientes:

1. `/proc/#número#/`. Dentro de `/proc/` existe un subdirectorio por cada proceso que actualmente es conocido por el sistema. El nombre del subdirectorio es precisamente el PID (*Process IDentifier*) del proceso, es decir, el número entero positivo por el cual el núcleo de Linux identifica unívocamente a cada proceso en el sistema. Dentro del subdirectorio correspondiente a cada proceso podemos encontrar varios subdirectorios y archivos. Por ejemplo, si sabemos que el PID del proceso que ejecuta el **emacs** en el que estamos editando estas líneas es el `20540`, podemos ver su contenido:

```
[root@yoda root]# ll /proc/20540
total 0
-r--r--r--    1 aterrasa aterrasa        0 dic  9 09:57 cmdline
lrwxrwxrwx    1 aterrasa aterrasa        0 dic  9 09:57 cwd -> /home/aterrasa/
/DOCBOOK/comos-linux-basico
-r--------    1 aterrasa aterrasa        0 dic  9 09:57 environ
lrwxrwxrwx    1 aterrasa aterrasa        0 dic  9 09:57 exe -> /usr/bin/emacs
dr-x------    2 aterrasa aterrasa        0 dic  9 09:57 fd
-r--r--r--    1 aterrasa aterrasa        0 dic  9 09:57 maps
-rw-------    1 aterrasa aterrasa        0 dic  9 09:57 mem
-r--r--r--    1 aterrasa aterrasa        0 dic  9 09:57 mounts
lrwxrwxrwx    1 aterrasa aterrasa        0 dic  9 09:57 root -> /
-r--r--r--    1 aterrasa aterrasa        0 dic  9 09:57 stat
-r--r--r--    1 aterrasa aterrasa        0 dic  9 09:57 statm
-r--r--r--    1 aterrasa aterrasa        0 dic  9 09:57 status
```

   De estos archivos, los más importantes son:

   - `cmdline`. Contiene la línea de órdenes que inició el proceso.
   - `cwd`. Es un enlace simbólico al directorio de trabajo actual del proceso.
   - `environ`. Contiene la lista de variables de entorno del proceso.
   - `exe`. Enlace simbólico que apunta al fichero ejecutable (normalmente binario) del proceso.
   - `fd/`. Directorio que contiene enlaces simbólicos a los descriptores de fichero abiertos por el proceso actualmente.
   - `maps`. Contiene información acerca de los mapas de memoria (virtual) ocupada por el proceso, incluyendo información sobre la ubicación de sus ejecutables y bibliotecas.
   - `stat`, `statm`, y `status`. Estos archivos contienen diferentes imágenes del estado actual del proceso, incluyendo estado de la memoria, información de protección, etc.

2. `/proc/self/`. Enlace simbólico al subirectorio del proceso que actualmente está en ejecución.

3. `/proc/fs/`. Contiene información sobre los directorios que actualmente se están exportando (sólo tiene información si el sistema es un servidor NFS).

4. `/proc/ide/`. Contiene información sobre los discos duros IDE que están conectados en el sistema, incluyendo datos de los manejadores, opciones de configuración y estadísticas de uso.

5. `/proc/net/`. Este directorio contiene información exhaustiva sobre protocolos, parámetros y estadísticas relacionadas con el acceso a la red.

6. `/proc/scsi/`. Equivale al directorio `/proc/ide/`, pero para dispositivos SCSI.

7. `/proc/sys/`. Este subdirectorio es especial y distinto del resto de directorios que hemos visto hasta ahora, puesto que no sólo permite leer informacion actual del núcleo, sino que permite *modificarla* en algunos casos. Esto se explica con más detalle en la siguiente sección.

8.4. Modificar información del núcleo

Una de las funcionalidades avanzadas de `/proc/` es, como decíamos, la posibilidad de modificar algunos de los parámetros internos del núcleo sin tener que recompilarlo y reiniciar el sistema. Esta sección presenta un pequeño resumen de cómo realizarlo.

Como cualquier programa en ejecución, el núcleo de Linux posee internamente un conjunto de variables globales (o parámetros) que reflejan y/o condicionan su funcionamiento. Algunos de estas variables, como por ejemplo los procesos que se están ejecutando, la ocupación de la memoria o los sistemas de archivos montados, pueden ser consultados por el usuario mediante la lectura de los archivos correspondientes del directorio `/proc/`, tal como veíamos en la sección anterior. Pero `/proc/` permite también la *modificación* en línea de algunas de esas variables, sin necesidad de reiniciar el sistema, de forma que podemos ajustar el comportamiento del núcleo dinámicamente. Para ello se dispone del directorio `/proc/sys/`.

8.4.1. El directorio `/proc/sys`

El directorio `/proc/sys/` contiene un subárbol de directorios y archivos en el que se organizan muchos parámetros del núcleo, subdivididos por categorías (es decir, por subdirectorios). Los archivos pueden tener permisos sólo de lectura o bien permitir su modificación, aunque sólo al administrador (`root`). Estas dos posibilidades indican, respectivamente, archivos que muestran información sobre parámetros y otros que permiten además, la modificación de estos parámetros. En este último caso, si sobreescribimos el contenido del archivo, estaremos cambiando el parámetro correspondiente del núcleo. Por ejemplo:

```
[root@yoda root]# cat /proc/sys/kernel/hostname
yoda.dsic.upv.es
[root@yoda root]# echo obiwan.dsic.upv.es > /proc/sys/kernel/hostname
```

```
[root@yoda root]# cat /proc/sys/kernel/hostname
obiwan.dsic.upv.es
```

Los subdirectorios más significativos de `/proc/sys/` son los siguientes:

- `/proc/sys/dev/`. Este directorio proporciona los parámetros de configuración de algunos dispositivos físicos conectados al sistema, como por ejemplo unidades de cdrom. Por ejemplo:

```
[root@yoda root]# ls -l /proc/sys/dev/cdrom/
total 0
-rw-r--r--    1 root     root            0 dic  9 11:51 autoclose
-rw-r--r--    1 root     root            0 dic  9 11:51 autoeject
-rw-r--r--    1 root     root            0 dic  9 11:51 check_media
-rw-r--r--    1 root     root            0 dic  9 11:51 debug
-r--r--r--    1 root     root            0 dic  9 11:51 info
-rw-r--r--    1 root     root            0 dic  9 11:51 lock
[root@yoda root]# cat /proc/sys/dev/cdrom/info
CD-ROM information, Id: cdrom.c 3.12 2000/10/18

drive name:             sr0
drive speed:            40
drive # of slots:       1
Can close tray:         1
Can open tray:          1
Can lock tray:          1
Can change speed:       1
Can select disk:        0
Can read multisession:  1
Can read MCN:           1
Reports media changed:  1
Can play audio:         1
Can write CD-R:         1
Can write CD-RW:        1
Can read DVD:           0
Can write DVD-R:        0
Can write DVD-RAM:      0
```

Mediante la modificación de archivos como `autoclose` o `autoeject` (que pueden contener `0` o `1`) podemos controlar la activación o desactivación de esas características de la unidad de cd.

- `/proc/sys/fs/`. Este directorio contiene numerosos parámetros concernientes a los sistemas de archivos montados en el sistema, incluyendo cuotas, manejadores de archivo, inodos, etc:

```
[root@yoda root]# ls -l /proc/sys/fs/
total 0
dr-xr-xr-x    2 root     root            0 dic  9 12:10 binfmt_misc
-r--r--r--    1 root     root            0 dic  9 12:10 dentry-state
-rw-r--r--    1 root     root            0 dic  9 12:10 dir-notify-enable
```

```
-rw-r--r--    1 root     root            0 dic  9 12:10 file-max
-r--r--r--    1 root     root            0 dic  9 12:10 file-nr
-r--r--r--    1 root     root            0 dic  9 12:10 inode-nr
-r--r--r--    1 root     root            0 dic  9 12:10 inode-state
-rw-r--r--    1 root     root            0 dic  9 12:10 lease-break-time
-rw-r--r--    1 root     root            0 dic  9 12:10 leases-enable
-rw-r--r--    1 root     root            0 dic  9 12:10 overflowgid
-rw-r--r--    1 root     root            0 dic  9 12:10 overflowuid
dr-xr-xr-x    2 root     root            0 dic  9 12:10 quota
```

Por ejemplo, si quisiéramos incrementar la cantidad de manejadores de fichero (es decir, el número máximo de ficheros que pueden ser abiertos simultáneamente), deberíamos incrementar la cantidad que aparece en `file-max`.

- `/proc/sys/kernel/`. El contenido de este directorio incluye aspectos de configuración y parámetros que afectan directamente el funcionamiento del núcleo, como por ejemplo el comportamiento de [Ctl]-[Alt]-[Supr] para reiniciar el sistema, el nombre del ordenador (ver ejemplo arriba), el número máximo de hilos de ejecución que el núcleo puede ejecutar, etc.

- `/proc/sys/net/`. Este directorio permite ver y controlar el funcionamiento de muchos aspectos del núcleo relacionados con la red, incluyendo los diferentes protocolos queimplementa Linux (`ethernet`, `ipx`, `ipv4`, `ipv6`, etc). Por ejemplo, para permitir la retransmisión de paquetes entre dos tarjetas de red conectadas al ordenador, haríamos lo siguiente:

```
[root@yoda root]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- `/proc/sys/vm/`. Este subdirectorio permite la configuración del subsistema de memoria virtual del núcleo, como por ejemplo el funcionamiento del servicio (o "demonio") de intercambio (**kswapd**), la cantidad de memoria que se dedicará a buffers del núcleo, el número máximo de áreas de memoria que pueden tener los procesos, etc.

8.4.2. El mandato sysctl

Como alternativa a escribir directamente el valor (o los valores) en los archivos de `/proc/sys/`, Linux dispone de un mandato denominado **sysctl**, que puede ser utilizado para leer y modificar todos los parámetros del kernel que se ubican en ese directorio. Por ejemplo, para modificar el nombre de la máquina, como veíamos arriba, podríamos ejecutar lo siguiente:

```
[root@yoda root]# cat /proc/sys/kernel/hostname
yoda.dsic.upv.es
[root@yoda root]# sysctl -w kernel.hostname=obiwan.dsic.upv.es
kernel.hostname = obiwan.dsic.upv.es
```

```
[root@yoda root]# cat /proc/sys/kernel/hostname
obiwan.dsic.upv.es
```

La orden **sysctl** se utiliza para leer y modificar *claves* que, como vemos, coinciden con las rutas de los archivos correspondientes dentro del directorio `/proc/sys/`. Para un listado completo de las claves que podemos modificar, podemos teclear **sysctl -a**.

A

Nota Legal

Administración básica de Linux

por Fernando Ferrer García y Andrés Terrasa Barrena

Versión 1.0, diciembre 2003