Preparación para el examen LPI 101

Tema 102

Instalación y administración de paquetes

Créditos y licencia de uso

Coordinación:
Manuel Guillán (xLekOx) lpi@xlekox.org
Kiefer Von Jammo (Kiefer) kiefer@khrooon.net

Traducción:
Juan Maria Gil (Smooth) yo@juanmaria.com
Pablo Taboada (java) ptaboada@wanadoo.es
Kiefer Von Jammo (Kiefer) kiefer@khrooon.net
Carmen Eugenio (nemrac) meneiro@ono.com
Ivan Servia (katas) ivanservia@hotmail.com

Maquetación y corrección:
Manuel Guillán (xLekOx) lpi@xlekox.org
Kiefer Von Jammo (Kiefer) kiefer@khrooon.net
Javier Pulido (jpulido) javier.pulido@wanadoo.es
(alexasi) batra1@terra.es

Versión 1.0 (07-03-2005 16:30)

ÍNDICE

Índice de contenido

Tema 102.1
Las particiones en GNU/Linux

Introducción

Este capítulo cubre las herramientas y tareas asociadas con el sistema de archivos y su administración. Para resumirlo de una forma sencilla, se podría decir que un sistema de archivos es la forma en la que un sistema operativo organiza los archivos en un medio de almacenamiento físico de forma que pueda encontrarlos cuando los necesite. Se podran utilizar estas herramientas para crear, mantener y controlar el sistema de archivos.

Los comandos y términos que se verán en este tema son:

- / (root) partición primaria
- /var
- / home y otras
- Partición swap
- Puntos de montaje
- Particiones
- Cilindro 1024

Este tema tiene un peso (importancia) de 5 de cara al examen final de la certificación LPI 101. El total de la suma de pesos de todos los temas es de 106.

Generalidades del Sistema de Archivos Linux

El sistema de archivos es el principal componente de cualquier sistema operativo, y es importante comprender el sistema de funcionamiento y uso de este sistema. Es necesario conocer las diferencias entre los distintos sistemas de archivos, así como el uso adecuado de cada uno.

Distintos dispositivos de almacenamiento pueden contener archivos en un sistema Linux. Discos duros, CD-ROMs, disquetes, discos de red y otros dispositivos extraibles se pueden utilizar para almacenar archivos. Cada uno de estos dispositivos utiliza el sistema de archivos para organizarlos. Estos sistemas de archivos organizan los archivos en una estructura de directorios en forma de árbol, con subdirectorios colgando a partir del directorio raíz. El dispositivo y el sistema operativo son los que establecen el sistema de archivos utilizado. Linux puede utilizar una gran cantidad de dispositivos y sistemas de archivos diferentes, dependiendo de la configuración del kernel de Linux.

El comando mount se utiliza para conectar otros sistemas de archivos con el sistema de archivos principal de Linux, que generalmente y por defecto es el sistema de archivos ext2. El usuario root tiene control sobre la localización de otros sistemas de archivos adicionales. El root puede proporcionar privilegios a otros usuarios para el montado de sistemas de archivos específicos, como los de CD-ROMs y disquetes, de forma que puedan ser utilizados en el sistema. Cuando se trabaja con dispositivos extraibles es importante recordar que cada disco debe ser montado para poder trabajar con él. Si se desea acceder a otro disco diferente, primero deberemos desmontar el disco actual, cambiar el disco en la unidad y volver a montar el nuevo disco para utilizarlo. También se pueden configurar los sistemas de archivos para que se monten automáticamente cuando arranque el sistema. Esta posibilidad es útil cuando se trabaja con sistemas de archivos almacenados en una red, o en discos duros locales del equipo. Las herramientas utilizadas para para permitir estas funciones se recogen en este capítulo.

Tipos de Sistemas de Archivos

Se puede acceder a sistemas de archivos muy diferentes utilizando un sistema Linux. La tabla 2-1 muestra algunos de estos sistemas y su uso:

Tabla 2-1 Sistemas de Archivos en Linux

Formato	***Uso***
ext2	Sistema de archivos de Linux.
iso9660	Sistema de archivos de CD-ROM.
minux	Sistema de archivos Minux.
msdos	Sistema de archivos MS-DOS FAT de 16 bits
vfat	Sistema de archivos Windows FAT de 32 bits, utilizando nombres largos de archivo.
hpfs	Sistema de archivos OS/2.
proc	Sistema de archivos de procesos Linux.
nfs	Sistema de archivos de red, utilizado para acceder a sistemas remotos.
swap	Sistema de archivos Linux swap.
sysv	Sistema de archivos V de sistemas UNIX .

Estos sistemas de archivos se pueden entender como lenguajes. Linux es políglota, pero debe conocer el lenguaje adecuado que debe hablar para comunicarse con otro sistema de archivos. Como se puede observar, Linux soporta sistemas de archivos utilizados por otros sistemas operativos. Esto es muy útil para un equipo que disponga de arranque dual con esos otros sistemas operativos. Utilizando el soporte para otros sistemas de archivos, podemos acceder a los datos de particiones no-Linux y leer y escribir sobre ellas. El sistema de archivos NTFS, utilizado por Windows NT y Windows 2000 no se encuentra listado en la tabla anterior; sin embargo actualmente se está desarrollando soporte para este sistema de archivos de modo que se pueda escribir sobre NTFS al igual que sobre otros sistemas de archivos (Se puede leer sin problema y escribir en determinadas circunstancias, aunque hay proyectos que soportan la escritura total sin problemas). El sistema de archivos Reiser, reiserfs, es otro de los sistemas no listados en la tabla. Se trata de un sistema de archivos JOURNALING utilizado por algunos sistemas Linux para permitir la recuperación del sistema en caso de fallo. Este sistema de archivos será incluido en las últimas versiones del kernel de Linux.

Junto con los distintos sistemas de archivos, es importante entender como se nombran los dispositivos en un sistema Linux. La tabla 2-2 muestra algunos de los prefijos utilizados para dispositivos en sistemas Linux:

Tabla 2-2 Nombres de Dispositivos

Nombre	***Tipo***
hd	Particiones de discos duros IDE
sd	Particiones de discos duros SCSI
sr	Discos CD-ROM SCSI
fd	Disquetes
st	Dispositivos tipo cinta SCSI
ht	Dispositivos tipo cinta IDE
tty	Terminales
lp	Impresoras
pty	Terminales remotos
js	Puertos de joystick
midi	Puertos MIDI
ttyS	Puertos Serie
cua	Puertos COM
cdrom	Discos CD-ROM. A menudo es un simple enlace al dispositivo IDE o SCSI real.
modem	Modems.

Los prefijos se combinan con un número de dispositivo. En los discos duros, el disco se especifica por medio de una letra como “a” para el primer disco, “b” para el segundo, etc. La partición se especifica por medio de un número, siendo “1” para la primera partición, “2” para la segunda, etc.

Ejemplos de nombres de dispositivos:

hda1	Primera partición en el primer disco duro IDE.
hdb2	Segunda partición en el segundo disco duro IDE.
cdrom	Primer unidad de CD-ROM.
cdrom1	Segunda unidad de CD-ROM.
sda1	Primera partición del primer disco duro SCSI.
fd0	Primera unidad de disquete.

Estos nombres de dispositivos se utilizan para direccionar los dispositivos dentro de un sistema Linux. Todos los dispositivos se almacenan en el directorio /dev. Los nombres de dispositivos se pueden enlazar a otros dispositivos; por ejemplo, cdrom se puede enlazar a /dev/sr0 si hay un CD-ROM SCSI instalado en el sistema. Estos enlaces permiten el direccionamiento estándar de dispositivos dentro del sistema. Examinando el contenido del directorio /dev podemos ver los enlaces y la localización exacta de los dispositivos del sistema.

Recordatorio de Examen: En el examen habrá preguntas relativas a los dispositivos, por lo que es importante comprender correctamente el sistema de nombrado de dispositivos.

Consideraciones cuando se crea un sistema de archivos

Se deben tener en cuenta algunas consideraciones cuando se crea un sistema de archivos nuevo. El sistema de archivos NO contiene unicamente datos de los archivos almacenados en el disco. Parte del disco se utiliza para almacenar etiquetas asociadas al sistema de archivos. Esto incluye espacio para punteros que almacenan la dirección de los datos incluidos en los archivos, así como el tamaño y la etiqueta del sistema de archivos. Toda esta información utiliza espacio del disco. La configuración por defecto para estos componentes puede afectar a lo que está almacenado en la partición, por lo que es importante comprender estos componentes antes de crear una nueva partición. Es más difícil corregir problemas una vez que la partición ha sido creada y los datos han sido almacenados en el sistema de archivos.

i-nodes

Los punteros utilizados para identificar la localización de los datos almacenados se conocen como i-nodes. Éstos se utilizan en sistemas de archivos basados en UNIX y no se utilizan en sistemas de archivos tipo FAT. Cuando se crea un sistema de archivos, también se crean los i-nodes que serán utilizados. Esto establece el número de archivos que podrán ser almacenados en el sistema. A menos que se especifique el número de i-nodes, Linux tratará de determinar el número de i-nodes necesario basándose en el tamaño de la partición. Esto puede provocar espacio desaprovechado si el sistema de archivos va a contener un pequeño número de archivos muy grandes. También se puede perder espacio en el disco si el sistema almacenase un gran número de archivos muy pequeños. Una vez que se han utilizado todos los i-nodes creados, no se podrán almacenar más archivos en el sistema, aunque tengamos espacio libre para ello. La configuración de i-nodes por defecto permite que la partición sea llenada con archivos de 2K.

Atención!!: Es muy importante entender la importancia de los i-nodes. Una vez que se han agotado los i-nodes en un sistema de archivos, no se podrán crear nuevos archivos, y el resto del espacio en el sistema de archivos será inutilizable.

Superblocks

Los i-nodes de un sistema de archivos se almacenan dentro de lo que se conoce como superbloque (superblock. El superbloque es un registro que también contiene información sobre el tamaño del sistema de archivos y su localización en el disco. También se almacena aquí otra información importante sobre la configuración del sistema de archivos como los cilindros y los bloques de disco utilizados. La información almacenada dentro del superbloque es crucial para acceder al sistema de archivos. Por ello, a lo largo del disco se almacenan varias copias del superbloque. Esto proporciona tolerancia a fallos, de forma que, si se daña un superbloque se puede utilizar otro, y recuperar el sistema. Una copia de seguridad del superbloque se almacena siempre cada bloque de 8K del sistema de archivos.

Creando Particiones y Sistemas de Archivos

Cuando se trabaja con unidades de disco se deben llevar a cabo varios pasos antes de que el disco sea utilizable por el sistema. Primero, se debe particionar el disco; esto permite que el disco se estructure para almacenar datos. Una vez que el disco haya sido segmentado en particiones, se debe crear el sistema de archivos.
Linux proporciona las herramientas necesarias para particionar y crear el sistema de archivos en un disco duro. Esta sección cubre estas herramientas y cómo utilizarlas.

Tipos de Partición

Las unidades de disco duro utilizadas por Linux y otros sistemas siguen unas estrategias de partición estándar. La información de la partición se almacena en el disco físico y permite que coexistan diferentes sistemas operativos dentro de un único equipo. El particionado de discos es útil por diversos motivos. Se pueden almacenar los datos del sistema en particiones separadas para asegurar que las diferentes partes del sistema operativo tienen suficiente espacio en el disco. Manteniendo los datos del sistema y los datos de los usuarios en particiones separadas también permite cierto grado de seguridad, proporcionando una barrera lógica entre el espacio al que acceden los usuarios y el espacio al que accede el sistema.

Las razones para particionar un disco son demasiado numerosas como para listarlas. Pueden variar desde temas relacionados con la seguridad, temas de política hasta física del disco. Independientemente de las razones para crear particiones, los tipos de particiones son las mismas. Un disco puede contener particiones primarias, extendidas y particiones de intercambio (swap).

Particiones Primarias

Todos los discos duros que tengan un sistema de archivos usan una partición primaria. Es la primera partición creada en el disco. Si todo el espacio del disco es utilizado por la partición primaria, ésta será la única partición del disco. Es posible tener varias particiones primarias en un único disco físico. Estas particiones se utilizan para arrancar el sistema y están limitadas a un máximo de cuatro en un mismo disco físico.

Particiones Extendidas

Si se necesitan más de cuatro particiones en el disco, es necesario crear una partición extendida. Cuando existe una partición extendida en un disco, no puede haber más de 3 particiones primarias en el mismo. Una partición extendida por si misma carece de utilidad. En realidad actúa como un contenedor de particiones lógicas, y puede contener varios de estos discos lógicos. Estas particiones no son arrancables, pero permiten tener un gran número de particiones en el sistema. Las particiones lógicas sólo pueden existir dentro de una partición extendida.

Particiones de Intercambio (Particiones swap)

Los sistemas Linux también utilizan hasta 8 particiones swap, o de intercambio. Estas particiones se utilizan para almacenar datos temporales y mejoran el rendimiento del sistema. Una partición swap se utiliza como memoria virtual y es necesaria para sistemas con menos de 16MB de RAM. En el pasado, el tamaño recomendado para la partición swap era el mismo que el de la memoria RAM del sistema. Actualmente se recomienda que el tamaño de la partición swap sea el doble que la memoria RAM del sistema, de modo que un sistema con 128MB de RAM debería tener una partición swap de al menos 256MB. Los kernel anteriores al 2.2 estaban limitados a particiones swap de 128MB; sin embargo, a partir de la versión 2.2, la partición swap en sistemas basados en arquitecturas i386 puede llegar a ser de 2GB. El sistema Linux combina la cantidad de RAM y la partición swap para determinar la cantidad total de memoria virtual disponible para el sistema. La cantidad óptima de memoria virtual necesaria para un sistema varía en función de las aplicaciones que se estén ejecutando en el mismo. Teniendo en cuenta que es normal hoy en día tener memorias superiores a 512MB, con tener una swap de la mitad de la RAM es más que suficiente, incluso se puede prescindir de la misma, debido a la buena gestión de la memoria que realiza el nucleo.

En el mundo real: Se debe recordar que la memoria RAM es más rápida que la partición swap contenida en el disco duro. Si trabajamos con aplicaciones que consuman grandes cantidades de memoria, probablemente estará indicado invertir en añadirle más memoria RAM al sistema.

Disposición estándar de archivos

La instalación de Linux crea una serie de directorios para almacenar los archivos del sistema. Cualquier instalación normal, independientemente de la distribución que se trate, crea una estructura de directorios entre los que se encuentran los siguientes:

El directorio /

Todo surge a partir del directorio raíz (/). El directorio raíz es el directorio a partir del cual todos los demás son subdirectorios o subcomponentes. Cuando se especifican localizaciones utilizando direcciones absolutas, siempre se comienza por este directorio, porque es el origen último, y es imposible moverse más allá del mismo, ya que no hay directorio sobre él.

El directorio /bin

El directorio bin contiene los ejecutables, que son esenciales para el funcionamiento del sistema operativo Linux. Gran parte de las utilidades vistas hasta ahora se localizan en el directorio bin, incluyendo: cat, cp, date, ls, mkdir, mv, ps, sed, ...

Como regla general, los ejecutables o archivos binarios localizados en el directorio bin son accesibles para todos los usuarios. Los ejecutables que no son críticos para el funcionamiento del sistema, o aquellos que son necesarios para todos los usuarios, generalmente aparecen en el directorio /usr/bin en lugar de en /bin.

El directorio /boot

Este directorio almacena los archivos necesarios para arrancar el sistema, excepto los archivos de configuración, así como el kernel del sistema. En algunas implementaciones, el kernel se almacena en el directorio raíz (como recuerdo de los sistemas UNIX), pero en las versiones más modernas se usa el directorio /boot.

El directorio /dev

El directorio dev almacena las definiciones de dispositivos. El hecho de copiar un archivo sobre un icono gráfico de la disquetera que se encuentre en el escritorio es posible gracias a que la definición de la disquetera figura en el directorio /dev. Cada dispositivo tiene asociado un archivo, tanto si se trata de un disco, de una terminal, de una controladora, etc. El siguiente listado muestra algunos de los archivos que se pueden encontrar en el directorio /dev.

```
brw-rw-rw-   1root root      2, 4   Aug 10 1999   floppy
brw-r——      1 root operator 3, 1   Aug 10 1999   hard drive1
crw-rw——     1 root lp       6, 0   Aug 10 1999   lp0
crw-rw——     1 root lp       6, 1   Aug 10 1999   lp1
crw-rw——     1 root lp       6, 2   Aug 10 1999   lp2
brw-rw-r—    1 root disk     23, 0  Aug 10 1999   cd
crw-r——      1 root kmem     1, 1   Aug 10 1999   mem
crw-rw-rw-   1 root root     1, 3   Aug 10 1999   null
crw-rw-rw-   1 root root     10, 1  Sep 13 10:29  mouse
brw———       1 root root     1, 0   Aug 10 1999   ram0
brw———       1 root root     1, 1   Aug 10 1999   ram1
brw———       1 root root     31, 0  Aug 10 1999   rom0
brw———       1 root root     31, 1  Aug 10 1999   rom1
br————       1 root root     31, 8  Aug 10 1999   rrom0
br————       1 root root     31, 9  Aug 10 1999   rrom1
brw-rw-r—    1 root disk     15, 0  Aug 10 1999   sonycd
crw—w—w-     1 root root     4, 0   Aug 10 1999   tty0
crw-rw——     1 root tty      4, 1   Jul 6 15:27   tty1
crw-rw——     1 root tty      4, 10  Aug 10 1999   tty10
crw-rw——     1 root tty      4, 11  Aug 10 1999   tty11
crw-rw——     1 root tty      4, 12  Jul 6 15:27   tty12
```

La primera cosa a tener en cuenta es que la lista no se parece a los listados de archivos vistos hasta ahora. El primer carácter del campo de permisos es siempre “b” o “c”, para indicar cómo se tratan

los datos (por bloques o por caracteres). Por norma general, los dispositivos que requieren un grado de interacción constante, como un ratón o un terminal (tty), se basan en caracteres. Los dispositivos que no requieren ese grado de interacción una vez que un proceso ha comenzado, como las disqueteras, memorias (RAM y ROM) y lectores de CD, se basan en el tratamiento por bloques.

La segunda diferencia a tener en cuenta es que el el tamaño de los archivos no figura en bytes, sino en un par de números separados por comas. La creación de este tipo de archivos especiales sale fuera del objetivo de estudio del examen LPI, pero se debe saber que debemos utilizar la utilidad mknod para crear archivos de dispositivo.

El directorio /etc

En cualquier lenguaje, etc significa etcétera. En el mundo Linux, sin embargo, el directorio /etc contiene archivos específicos de la máquina. Por ejemplo, tanto ABC Corporation como DEF Corporation pueden instalar sistemas OpenLinux Caldera en máquinas tipo Intel. Cuando se hace esto, ambos tienen directorio raíz, ambos tienen directorio /bin con los mismos conjuntos de utilidades en ellos, etc. La principal diferencia entre ambas máquinas es el contenido de sus directorios /etc. Los usuarios que entren en la máquina ABC no serán los mismos que los que entren en la DEF; sus cuentas de usuario se almacenarán en /etc. Los grupos no serán los mismos en las dos organizaciones; sus archivos relacionados se almacenarán en /etc.
Otros archivos que se incluyen en este directorio son:

- motd: El archivo del "mensaje del día" con el texto que se mostrará al entrar al sistema.
- X11: Una carpeta que contiene los valores de X Window.
- HOSTNAME: Un archivo que contiene el nombre de la máquina.
- hosts: Un archivo de mapeo de nombres de máquina y direcciones IP de otras máquinas disponibles en la red.

En resumen, el directorio /etc mantiene los archivos de configuración del sistema para una máquina específica.

El directorio /home

Como su propio nombre indica, el directorio /home contiene los subdirectorios que son directorio de origen para cada uno de los usuarios. Por ejemplo, el usuario "pepe", cuando ejecuta el comando cd, se situa en el directorio /home/pepe.
Cada directorio /home/usuario de cada usuario proporciona el lugar para almacenar sus archivos, así como para almacenar los archivos de configuración individuales de ese usuario. Algunos servicios, como FTP o HTTP, también crean directorios bajo /home.

Atención!: Se debe recordar, que por motivos de seguridad, no existe un directorio /home/root. El directorio de inicio para el root es el directorio /root.

El directorio /lib

Los archivos de librerías compartidas que necesitan los ejecutables (como los que se almacenan en /bin), se encuentran en el directorio /lib y en los subdirectorios que descienden de él. Generalmente, las librerías son ejecutables escritos en lenguaje C.

El directorio /mnt

El directorio /mnt contiene sistemas de archivos externos que hayan sido montados. Las entidades que aparezcan dentro de este directorio nunca pertenecen al sistema de archivos del sistema, sino que representan recursos externos a los que se puede acceder por medio del directorio /mnt. Los recursos externos pueden ser otros sistemas de archivo o dispositivos.

Los dispositivos aparecen como como directorios con nombres comunes (cdrom, floppy, ...). El subdirectorio /mnt/tmp se usa para mantener archivos temporales, pero es preferible el uso del directorio /tmp para ello.

El directorio /opt

El directorio /opt contiene complementos de las aplicaciones (add-ins). No todas las aplicaciones instalan sus complementos en este directorio, pero cuando lo hacen, crean un subdirectorio dentro de /opt utilizando el nombre de la aplicación. Por ejemplo, una aplicación llamada DEF creará un subdirectorio /opt/DEF en el que almacenará sus variables.
No hay ninguna norma que obligue a que aplicaciones de terceros deban incluir sus complementos en /opt, pero este comportamiento se ha heredado de los días de UNIX. Algunos subdirectorios comunes en /opt son:
- kde: para las variables de entorno del escritorio KDE.
- netscape: para el navegador.

El directorio /proc

El directorio /proc es el sistema de archivos virtual. Se genera y actualiza dinámicamente, y contiene información sobre los procesos, el kernel e información relativa al sistema.

Los procesos se representan por carpetas, cada una de las cuales tiene permisos y variables asociadas con ella. Otras informaciones del sistema se mostrarán como archivos, como en el ejemplo que se muestra a continuación:

```
$ ps
PID TTY TIME CMD
15193 pts/0 00:00:00 bash
15220 pts/0 00:00:00 sleep
15222 pts/0 00:00:00 sleep
15236 pts/0 00:00:00 ps


$ ls -l
dr-xr-xr-x 3 root root 0 Sep 20 08:34 15193
dr-xr-xr-x 3 root root 0 Sep 20 08:34 15220
```

```
dr-xr-xr-x 3 root root 0 Sep 20 08:34 15222
dr-xr-xr-x 4 root root 0 Sep 20 08:34 bus
-r—r—r— 1 root root 0 Sep 20 08:34 cmdline
-r—r—r— 1 root root 0 Sep 20 08:34 cpuinfo
-r—r—r— 1 root root 0 Sep 20 08:34 devices
-r—r—r— 1 root root 0 Sep 20 08:34 dma
-r—r—r— 1 root root 0 Sep 20 08:34 fb
-r—r—r— 1 root root 0 Sep 20 08:34 filesystems
dr-xr-xr-x 2 root root 0 Sep 20 08:34 fs
dr-xr-xr-x 4 root root 0 Sep 20 08:34 ide
-r—r—r— 1 root root 0 Sep 20 08:34 interrupts
-r—r—r— 1 root root 0 Sep 20 08:34 ioports
-r————— `1 root root 67112960 Sep 20 08:34 kcore
-r————— 1 root root 0 Sep 20 08:16 kmsg
-r—r—r— 1 root root 0 Sep 20 08:34 ksyms
-r—r—r— 1 root root 0 Sep 20 08:34 loadavg
-r—r—r— 1 root root 0 Sep 20 08:34 locks
-r—r—r— 1 root root 0 Sep 20 08:34 mdstat
-r—r—r— 1 root root 0 Sep 20 08:34 meminfo
-r—r—r— 1 root root 0 Sep 20 08:34 misc
-r—r—r— 1 root root 0 Sep 20 08:34 modules
-r—r—r— 1 root root 0 Sep 20 08:34 mounts
dr-xr-xr-x 4 root root 0 Sep 20 08:34 net
dr-xr-xr-x 3 root root 0 Sep 20 08:34 parport
-r—r—r— 1 root root 0 Sep 20 08:34 partitions
-r—r—r— 1 root root 0 Sep 20 08:34 pci
-r—r—r— 1 root root 0 Sep 20 08:34 rtc
dr-xr-xr-x 2 root root 0 Sep 20 08:34 scsi
lrwxrwxrwx 1 root root 64 Sep 20 08:34 self -> 15252
-r—r—r— 1 root root 0 Sep 20 08:34 slabinfo
-r—r—r— 1 root root 0 Sep 20 08:34 sound
-r—r—r— 1 root root 0 Sep 20 08:34 stat
-r—r—r— 1 root root 0 Sep 20 08:34 swaps
dr-xr-xr-x 10 root root 0 Sep 20 08:34 sys
dr-xr-xr-x 4 root root 0 Sep 20 08:34 tty
-r—r—r— 1 root root 0 Sep 20 08:34 uptime
-r—r—r— 1 root root 0 Sep 20 08:34 version
```

Tema 102.2 Instalando un boot manager

Introducción

En este capítulo se verá como seleccionar, instalar y configurar un boot manager.

Los comandos que se verán en este tema son:
- lilo
- grub-install

La configuración de los archivos:
- /etc/lilo.conf
- /boot/grub/grub.conf

Y los conceptos:
- MBR
- superblock
- first stage boot loader

Este tema tiene un peso (importancia) de 1 de cara al examen final de la certificación LPI 101. El total de la suma de pesos de todos los temas es de 106.

LILO

Lilo es el boot manager más famoso de los sistemas GNU/Linux y se viene usando desde hace muchos años. Un boot manager es un gestor de arranque que se encarga de cargar el SO seleccionado. Otros boot manager traen más opciones (y en ocasiones mejores) pero Lilo continúa siendo la solución más empleada en las distribuciones.

Por norma general se instala en el MBR (master boot record) del disco duro. El MBR es donde la BIOS del sistema mira la información del boot (arranque). También puede ser instalado en el sector boot de una partición, normalmente cuando otro boot manager es el empleado como gestor predefinido, dejando al Lilo como gestor secundario.

Cuando el sistema arranca se verá un prompt parecido a:

LILO:

Presionando la tecla de tabulado se verán las posibles opciones de arranque, presionando la tecla enter se cargará la opción predeterminada. En este prompt se pueden especificar varias opciones del kernel, consultar las páginas del manual para más información.

Configurando el lilo

La configuración del Lilo reside en el fichero /etc/lilo.conf, en este fichero se guardan las opciones y parámetros del gestor de arranque. Se pueden usar multitud de configuraciones y particularidades de cada sistema o multisistemas, la revisión a fondo de su configuración se sale del temario del LPI, si se está más interesado en profundizar en el tema hay excelentes manuales y documentación sobre ello en http://es.tldp.org/ y como siempre en el propio manual: man lilo.conf

Para que los cambios realizados en el fichero de configuración tengan efecto es necesario ejecutar el comando lilo, el cual grabará los cambios en el sector de arranque y serán efectivos en el próximo arranque, cantidad de opciones y más información en man lilo.

Grub

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 gestor favorito respecto al resto de gestores de arranque disponibles para la arquitectura x86. A continuación se expone 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.

Configuración del GRUB

El grub se configura a través del fichero /boot/grub/grub.conf, al igual que el gestor lilo, tiene muchas opciones y funcionalidades, a diferencia del lilo, los cambios efectuados en grub.conf serán efectivos en el próximo arranque, sin necesidad de ejecutar ningún comando. Más info en la página citada anteriormente y en man grub. El comando para la instalación del grub es el grub-install, aunque en la mayoría de los casos se instala durante el proceso de creación del SO.

Grub o Lilo

GRUB y LILO constituyen los métodos más usados para arrancar un sistema GNU/Linux. Como cargadores de sistemas operativos, funcionan "fuera" de cualquier sistema operativo, usando tan sólo el sistema básico de entrada/salida (o BIOS) incluido en el hardware del mismo sistema.

GRUB y LILO están sujetos a algunas limitaciones impuestas por la BIOS en muchos ordenadores basados en Intel. La mayor parte de las BIOS no pueden acceder a más de dos discos duros y no pueden acceder a los datos localizados más allá del cilindro 1023 de cualquier unidad. Algunas BIOS nuevas no tienen estas limitaciones, aunque no sea lo más habitual.

Todos los datos que GRUB y LILO necesitan para acceder al momento de arranque de la máquina (incluido el kernel de Linux) están contenidos en el directorio /boot y que deben seguir unas normas:

En los dos primeros discos IDE
Si se tienen dos discos IDE (o EIDE), /boot debe estar en uno de estos. Observe que este límite de dos unidades también incluye cualquier CD-ROM IDE en el controlador primario IDE. Por tanto, si se tiene un disco duro IDE, y un CD-ROM IDE en el controlador primario, /boot debe estar localizado sólo en el primer disco duro, incluso si se tiene discos duros en su controlador IDE secundario.

En el primer disco IDE o SCSI
Si se tiene una unidad IDE (o EIDE) y una o más unidades SCSI, /boot tiene que estar o en el disco IDE o en la SCSI en el ID 0. Otros IDs de SCSI no funcionarán.

En los dos primeros discos SCSI

Si se tiene sólo discos SCSI, /boot debe encontrarse en un disco en el ID 0 o ID 1. No habrá ningún otro ID de SCSI con el que funcione.

Partición completamente dentro del Cilindro 1023

No importa qué configuración descrita se utilice, la partición que contendrá /boot debe ser creada dentro del cilidro 1023. Si la partición que contiene /boot supera el cilindro 1023, GRUB y LILO funcionarán inicialmente (porque todas las informaciones necesarias se encuentran antes del cilidro 1023), sin embargo, no funcionarán si tiene que cargar un kernel nuevo y éste se encuentra más allá de este cilindro.

En general, LILO funciona de forma parecida a GRUB a excepción de tres diferencias:

- No posee ninguna interfaz del comando interactiva.
- Almacena información sobre la localización del kernel o de si otro sistema operativo se debe cargar en el MBR.
- No puede leer las particiones ext2.

El primer punto significa que el intérprete de comandos para LILO no es interactivo y permite tan sólo un comando con argumentos.

Los últimos dos puntos significan que si se cambia el archivo de configuración de LILO o se instala un kernel nuevo, debe reescribir el gestor de arranque LILO de la etapa 1 al MBR llevando a cabo el comando siguiente:

```
/sbin/lilo-v-v
```

Este método es más arriesgado que el de GRUB, porque un MBR que no haya sido configurado adecuadamente deja el sistema sin poder arrancar. Con GRUB, si el archivo de configuración está configurado de forma errónea, se disparará por defecto la interfaz de la línea de comandos de modo que el usuario pueda arrancar el sistema manualmente.

El grub cada día es más empleado en las distribuciones por ser más flexible y ofrecer más y mejores características que lilo.

Tema 102.3 Instalar programas desde los fuentes

Introducción

En este capítulo se verá como construir e instalar un programa desde sus ficheros fuentes, se verán algunas opciones en el proceso de compilado.

Los comandos que se verán en este tema son:

- gunzip
- gzip
- bzip2
- tar
- configure
- make

Este tema tiene un peso (importancia) de 5 de cara al examen final de la certificación LPI 101. El total de la suma de pesos de todos los temas es de 106.

Instalando Software desde el código fuente

Hace unos años instalar software desde el código fuente era de lo más habitual, con la aparición de nuevas distribuciones y los paquetes binarios se perdió un poco esta costumbre, distribuciones como Gentoo rescatan este modo de administración de paquetes, si bien puede ser empleado en cualquier otra distribución. Los beneficios de instalar los paquetes desde su código fuente son muy numerosos, mayor personalización en cada PC que se instale con unas determinadas características, portabilidad de sistemas y arquitecturas y un largo etcetera.

Porqué motivo bajarse el código fuente si se dispone de paquetes binarios que ahorran el tiempo de espera? Mucha gente lo hace porque le gusta bucear entre el código, otros porque son desarrolladores y les interesa revisar este código en busca de lineas extrañas, backdoors, agujeros de seguridad, fallos en la programación, etc. No toda la gente se para en estos detalles, bien por falta de tiempo, por desconocimiento o simplemente porque no le interesa, en la mayoría de los casos, el código fuente está disponible antes de que salgan los binarios del programa en si.

La gran mayoría de los proyectos con licencia libre, tienen disponible para descargar el código fuente así como las últimas versiones del CVS (trabajo en desarrollo), algunos de los sitios más famosos son http://sourceforge.net o www.freshmeat.net. Normalmente vienen empaquetados con la extensión tar.gz. Una vez descomprimido el fichero se procede a la instalación, normalmente vienen instrucciones en los ficheros REAME o INSTALL, algunos de los paquetes traen un Makefile preconfigurado, mientras que otros lo generan para cada sistema. El Makefile es un fichero de texto que le indica al compilador como ha de hacer la tarea, los pasos básicos para la instalación de un nuevo software son:

1. Obtener el código fuente
2. Descomprimirlo a un directorio temporal
3. Leer los ficheros REAME e INSTALL
4. Ejecutar el comando ./configure
5. Hacer cambios si se requieren al fichero Makefile
6. Compilar el programa con el comando make
7. Finalmente, instalar el programa con make install

Obteniendo el código fuente

Como se dijo anteriormente lo normal es que en la propia web del proyecto esté disponible el código fuente en diversos formatos (tar.gz, zip, tar, rar ...). En los cds de las distribuciones, los últimos cds (que no se usan en la instalación) traen el código fuente de todos, o la gran mayoría, de los paquetes que se incluyen con la distro, caso por ejemplo de la SUSE (cd's 6 y 7).

Descomprimiendo el tarball

Los ficheros fuentes que se bajan normalmente son tarballs, es decir, un conjunto de ficheros agrupados con la utilidad tar y comprimidos con gzip (extensión tar.gz) para descomprimirlos basta con ejecutar el comando:

gunzip fichero.tar.gz

... o bien:

gzip -d fichero.tar.gz

... lo que dará como resultado el fichero descomprimido fichero.tar.Para extraer los ficheros del tar, se ejecuta el comando:

tar xvf fichero.tar

... obteniendo todos los ficheros del tar en el directorio actual. Normalmente las distribuciones traen versiones del tar que ya soportan la descompresión, de modo que ejecutando un solo comando se descomprime y se extraen los ficheros del documento. La opción a añadir es la z, de modo que el comando quedaría:

tar zxvf fichero.tar.gz

Comando tar
El comando tar es uno de los más usados en todas las distribuciones, además de servir para realizar backups también es una forma excelente para la distribución de archivos ya que combina el empaquetado y la compresión en un solo archivo. Generalmente tienen extensión .tar (aunque no es obligatoria) si se comprimen pueden tener extensión .tar.gz o bien .tgz (man tar para más info)

Comando gzip
gzip se usa tanto para comprimir como para descomprimir, esta utilidad conserva los permisos y la hora de creación de los ficheros, normalmente tienen la extensión .gz (man gzip para más info)

Comando gunzip
gunzip se usa para la descompresión de los ficheros comprimidos con gzip (man gunzip más info)

Comando bzip2
bzip2 es otro modo de compresión, su extensión más común es .bz2 (man bzip2 más info)

Comando bunzip2
Esta orden se usa para descomprimir ficheros creados con la utilidad bzip2. (man bunzip2 más info)

Ejecutando el script de configuración

El script ./configure automatiza la tarea de creación del fichero Makefile, chequeando problemas con las dependencias de compiladores y componentes. Si se quiere más información mientras se ejecuta el configure, se puede usar la opción debug: ./configure -enable-debug

Haciendo cambios al fichero Makefile

Hoy día gracias al script configure es muy raro tener que realizar cambios en el fichero Makefile resultante, no obstante, bajo algunas circunstancias, puede ser recomendable o necesario realizar algunos cambios: modificar directorios de destino, paths de los programas, etc.

Compilando el software

El siguiente paso es la compilación del software, basta con ejecutar el comando make una vez creado el Makefile, el resultado será el programa compilado y listo para instalar.

Instalando el software

Como última tarea nos queda instalar el software recien compilado, esto se hace con el comando make install, el cual instalará todos los archivos, las páginas del manual, etc. Normalmente el software viene acompañado de un fichero README donde se indica el nombre del ejecutable y en que directorios se se instalaron los ficheros del programa, las distribuciones actuales ya generan iconos o accesos directos en los menús una vez instalados.

Tema 102.4 Administrando librerías compartidas

Introducción

En este capítulo se verán las librerías compartidas de las que dependen los programas, así como su instalación cuando sean necesarias.

Los comandos que se verán en este tema son:
ldd
ldconfig

Se verá la configuración del fichero:
/etc/ld.so.conf

... y el LD_LIBRARY_PATH

Este tema tiene un peso (importancia) de 3 de cara al examen final de la certificación LPI 101. El total de la suma de pesos de todos los temas es de 106.

Administrando librerías compartidas

Cuando se escribe el código fuente de los programas, los programadores no re-escriben código de operaciones básicas. Estas operaciones son escritas una vez y reusadas muchas veces por otros programadores, facilitando de este modo la reutilización de código y el ahorro de tiempo/esfuerzo. Estas operaciones rutinarias se guardan en lo que se conocen como librerías compartidas (shared libraries). Para que una aplicación que emplea estas librerías se pueda compilar e instalar, es necesario el acceso a las mismas. Algunos programas incluyen en el ejecutable final estas librerías, por lo que al usarlo no será necesario disponer de las mismas, estos programas responden al nombre de aplicaciones compiladas estáticamente. Sin embargo otros programas enlazan con las librerías en tiempo de compilación, son los programas compilados dinámicamente.

Las aplicaciones estáticas pueden parecer más ventajosas, sin embargo, la inclusión de la librerías hace que el programa sea mayor de lo que sería si fuese compilado dinámicamente. Las librerías compartidas en GNU/Linux normalmente se guardan en diversas carpetas, algunas de las más comunes:

- /lib -> librerías principales
- /usr/lib -> librerías supletorias
- /usr/X11R6/lib -> librerías de las X-window

Las librerías compartidas normalmente emplean un nombre estandarizado:
nombrelibreria-major-minor-patch.so

La extensión .so se refiere a “shared objet” (Objeto compartido). Por ejemplo: libcrypt-2-1-3.so hace referencia a la librería libcrypt, en su versión 2.1 y el patch 3.

En muchos casos los links simbólicos se crean para las librerías, estos se nombran de la siguiente manera: nombre.so , o bien, nombre.so.major

Ejemplos: libcrypt.so ó libcrypt.so.2

Estos enlaces permiten a los programas enlazar con estas librerías independientemente de la versión (menor) y el parche actual.

Viendo las librerías compartidas necesarias

Si se quieren ver las librerías que usa un programa, basta con introducir el comando:

```
$ldd fichero
```

Por ejemplo, para ver las librerías que usa el comando wget:

```
$ldd wget

linux-gate.so.1 =>  (0xffffe000)
libssl.so.0.9.7 => /usr/lib/libssl.so.0.9.7 (0x40038000)
libcrypto.so.0.9.7 => /usr/lib/libcrypto.so.0.9.7 (0x40067000)
libdl.so.2 => /lib/libdl.so.2 (0x4015b000)
libc.so.6 => /lib/libc.so.6 (0x4015e000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Esto significa que usa las librerías linux-gate (v1), libssl(v0.9.7), libcrypto(v.0.9.7), libdl(v2), libc (v6) y ld-linux(v2)

Administrando los paths de las librerías

Si una aplicación no puede encontrar una librearía compartida que necesite, dará un error y finalizará su ejecución. Si la librería no está en el path predefinido, se puede añadir a la variable de entorno LD_LIBRARY_PATH de la siguiente manera:

export LD_LIBRARY_PATH=/usr/nuevopath

De este modo se añadirá /usr/nuevopath al path y hará que el programa que anteriormente daba error se ejecute normalmente.

Configurando librerías compartidas

Si una librería compartida se instala manualmente, se debe informar al sistema de la existencia de la nueva librería. La configuración se guarda en el fichero /etc/ld.so.conf que contiene un listado de directorios donde se encuentran las diferentes librerías compartidas. Por ejemplo:

/usr/lib
/usr/X11R6/lib/Xaw3d
/usr/X11R6/lib

Notar que no existe el directorio /lib, este es incluido por defecto ya que las librerías requeridas por el sistema se encuentran ahí. Para mejorar el rendimiento se crea un fichero caché (/etc/ld.so.cache), el cual contiene todas las librerías de estos directorios. Cuando el fichero de configuración se cambia, el fichero caché debe de ser actualizado, esto se hace con el comando ldconfig.

Tema 102.5 Administrando paquetes Debian

Introducción

En este capítulo se verá como administrar un sistema usando el administrador de paquetes de Debian. Esto incluye comandos para instalar, actualizar y desinstalar programas, así como otras características, como saber la versión instalada, contenidos, dependencias, integridad del paquete, etc.

Los comandos que se verán en este tema son:

- dpkg
- dselect
- dpkg-reconfigure
- apt-get
- alien

Se verá la configuración del fichero:

- /etc/dpkg/dpkg.cfg
- /var/lib/dpkg/*
- /etc/apt/apt.conf
- /etc/apt/sources.list

Este tema tiene un peso (importancia) de 8 de cara al examen final de la certificación LPI 101. El total de la suma de pesos de todos los temas es de 106.

Administrando los paquetes de Debian

La distribución Debian y todas sus derivadas usan herramientas de empaquetado propias, diferentes de sistemas como RedHat (basado en RPMs) o Gentoo (basado en los fuentes). El sistema de Debian se basa en 4 comandos principalmente:

- dpkg
- deselect
- apt-get
- alien

Los paquetes de Debian, o paquetes .deb, por norma general contienen ficheros binarios para instalar así como otra información, conocida como metadata; este incluye información del paquete, scripts que serán ejecutados, la lista de dependencias y conflictos o sugerencias. Algunos paquetes traen el código fuente y pueden ser compilados a mano.

Se usa una convención en los nombres de los paquetes:

paquete_version-build_arquitectura.deb

- paquete es el nombre del programa o utilidad.
- Versión, es el número de versión de la aplicación.
- build es el número que indica la versión del paquete, cada vez que se hace un empaquetado se incrementa.
- Arquitectura, es la plataforma para la cual fue destinada la compilación del paquete.

Existe un tipo especial de paquete, conocido como “task package” (lista de tareas). Son paquetes vacios que incluyen una lista de programas a instalar, se usan para facilitar instalaciones “grandes” como las X-window (sistema gráfico) y Gnome o KDE (escritorios), que tienen muchas dependencias. Se instalan del mismo modo que cualquier paquete y su formato es:
helix-gnome-task

Usando dpkg

dpkg es el nucleo del sistema de empaquetado de Debian, la gran mayoría de herramientas usan el dpkg y lo hacen más sencillo o con más opciones. A veces es más rápido usar el dpkg que otras herramientas a priori más sencillas.

Instalando paquetes

Una vez se tiene el paquete (.deb) que se quiere instalar, se usa el siguiente comando para instalarlo:

dpkg --install paquete.deb

...o bien :

dpkg -i paquete.deb

Durante la instalación del paquete, dpkg revisará si existen las dependencias necesarias para la instalación e informará con un error si no están instaladas.

Por ejemplo, al instalar el paquete ethereal:

```
# dpkg -i ethereal_0_8.13-2_i386.deb
Selecting previously deselected package ethereal.
(Reading database ... 54478 files and directories currently
installed.)
Unpacking ethereal (from ethereal_0.8.13-2_i386.deb) ...
dpkg: dependency problems prevent configuration of ethereal:
 ethereal depends on libpcap0 (>= 0.4-1); however:
  Package libpcap0 is not installed.
dpkg: error processing ethereal (--install):
 dependency problems - leaving unconfigured
Errors were encountered while processing:
 Ethereal
```

Como se puede observar es necesario el paquete libpcap0, debemos por tanto instalarlo por separado o bien con el mismo comando como sigue:

```
#dpkg --install ethereal_0_8.13-2_i386.deb libpcap0_0.4a6-3_i386.deb
(Reading database ... 54499 files and directories currently
installed.)
Preparing to replace ethereal 0.8.13-2 (using
ethereal_0.8.13-2_i386.deb) ...
Unpacking replacement ethereal ...
Selecting previously deselected package libpcap0.
Unpacking libpcap0 (from libpcap0_0.4a6-3_i386.deb) ...
Setting up libpcap0 (0.4a6-3) ...
Setting up ethereal (0.8.13-2) ...
```

Opciones de forzado

En ocasiones es necesario, bien por gusto o por necesidad, sobreescribir un error cuando se instala o se borra un programa. El dpkg ofrece varias opciones para ignorar los errores, se listan en la tabla 5-1.

Tabla 5.1 Opciones de forzado del comando dpkg

Opción	***Uso***
configure-any	Configura otros paquetes que ayudarán al actual en su instalación
hold	Procesa otro paquete, incluso si está marcado como hold (fijado)
bad-path	Incluso con ficheros perdidos
not-root	Intenta eliminar o añadir paquetes aun cuando no se es root
overwrite	Sobreescribe un fichero de un nuevo paquete, incluso si corresponde a otro paquete
depends-version	Convierte un error por falta de una versión concreta en las dependencias en un warning, de ese modo puede continuar la instalación
depends	Convierte todos los errores de dependencias en warnings

Opción	***Uso***
confnew	Usa siempre el archivo de configuración más nuevo
confold	Usa siempre el archivo de configuración más viejo
conflicts	Permite que paquetes con conflictos sean instalados
overwrite-dir	Sobreescribe el directorio de otro paquete por el nuevo
remove-essential	Borra paquetes del sistema, peligroso

Por ejemplo, si se quiere instalar un programa que tiene conflictos con otro, se debe de teclear:

```
#dpkg -install new_package.deb –force-conflicts
```

Desinstalando programas

Para borrar programas se usa el siguiente comando:

dpkg --remove paquete

...o bien:

dpkg -r paquete

Estos comandos borran todos los ficheros del paquete excepto los ficheros de configuración, que pueden ser necesarios en una posterior re-instalación. Para quitar todos los ficheros (del programa y de configuración) se debe usar la siguiente opción:

dpkg --purgue paquete

...o bien:

dpkg -P paquete

Al igual que durante la instalación de un programa, al desinstalarlo, dpkg comprueba las dependencias.

```
#dpkg --remove libpcap0
dpkg: dependency problems prevent removal of libpcap0:
 ethereal depends on libpcap0 (>= 0.4-1).
dpkg: error processing libpcap0 (--remove):
 dependency problems - not removing
Errors were encountered while processing:
 libpcap0
```

Consultando la base de datos de los paquetes

Debian tiene una base de datos donde se recopilan todos los paquetes instalados en el sistema, la herramienta dpkg permite consultar esa base de datos.

Para visualizar información general de un paquete instalado:

dpkg --print-avail paquete

...o bien:

dpkg -p paquete

Por ejemplo para visualizar información del paquete ethereal teclearíamos:

dpkg -p ethereal

Esto nos dará bastante información: quien mantiene el paquete, su tamaño, versión, dependencias, descripción, la suma md5. Esto es útil para obtener información de un paquete que no sabemos para que sirve o para ponerse en contacto con su desarrollador.

Listando paquetes

Para obtener una lista de todos los paquetes instalados en el sistema basta con ejecutar la orden:

dpkg --list <patron>

...o bien:

dpkg -l <patron>

Para este comando, la opción <patron> es un parametro opcional de búsqueda, sin él, se listarán todos los paquetes instalados en el sistema.

Por ejemplo, para listar todos los paquetes que tengan que ver con apache se introduce el comando:

```
#dpkg -l apache*
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-
config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems
(Status,Err: uppercase=bad)
||/ Name          Version        Description
+++-==============-==============-
==========================================
pn apache        <none>         (no description available)
pn apache-common <none>           (no description available)
pn apache-dev    <none>         (no description available)
pn apache-doc    <none>         (no description available)
un apache-modules <none> (no description available)
```

Hay varios paquetes de apache listados, pero ninguno de ellos instalado, sin embargo si hubo alguna vez que estuvieron instalados. Hay 3 columnas a la izquierda, con el siguiente significado:

- p - significa que el paquete fue desinstalado
- n - significa que no está instalado
- u – desempaquetado y listo para instalar
- i – instalado
- h – medio instalado

Hay diversos estados para los paquetes, se listan a continuación

Estado de selección: se usa con el comando dselect, los posibles estados son:

- unknown – estado desconocido
- install – el paquete está marcado para su instalación
- remove – marcado para desinstalar
- purge – marcado para desinstalación completa
- hold – marcado como fijo, no será actualizado

Estado actual:
- not installed - no instalado
- installed – instalado
- config-files – no está instalado pero existen ficheros de configuración
- unpacked – desempaquetado y listo para instalar
- failed-config – ocurrió un problema al ejecutarse la configuración en la instalación
- half-installed – la instalación no se completó.

Errores
- None – no hay errores
- Hold – está marcado como estático, no puede ser borrado ni actualizado
- Reinstallation required – Se requiere reinstalación del paquete

Mostrando el estado de un paquete

Para mostrar el estado individual de cada paquete con todos los detalles del mismo, se usa el comando:

```
dpkg --status paquete
```

...o bien:

```
dpkg -s paquete
```

Por ejemplo:

```
#dpkg -s ethereal
Package: ethereal
Status: install ok installed
Priority: optional
Section: net
Installed-Size: 2996
Maintainer: Frederic Peters <fpeters@debian.org>
Version: 0.8.13-2
Depends: libc6 (>= 2.1.94), libglib1.2 (>= 1.2.0), libgtk1.2
(>= 1.2.8-1), libpc
ap0 (>= 0.4-1), libsnmp4.1, xlibs (>= 4.0.1-1), zlib1g (>=
1:1.1.3)
Description: Network traffic analyzer
 Ethereal is a network traffic analyzer, or "sniffer", for Unix
and
 Unix-like operating systems. It uses GTK+, a graphical user
interface
 library, and libpcap, a packet capture and filtering library.
```

Listando los ficheros de un paquete

Para listar los ficheros que contiene un paquete basta con ejecutar el comando:

dpkg --listfiles paquete

...o bien:

dpkg -L paquete

Mostrando el paquete propietario de un fichero

Para mostrar que paquete instaló un determinado archivo se usa el comando:

dpkg --search archivo

...o bien:

dpkg -S archivo

Por ejemplo:

```
#dpkg -S /etc/issue.net
base-files: /etc/issue.net
```

Lo cual indica que el fichero buscado fue instalado por el paquete base-files.

Observando los paquetes disponibles

La gran mayoría de información del comando dpkg está almacenada en el directorio /var/lib/dpkg , dos de los archivos más importantes son el available y status. El primero de ellos muestra los paquetes que están disponibles y el segundo muestra el estado de los paquetes.

Usando dselect

La herramienta deselect es una interface gráfica del comando dpkg, es decir, una alternativa visual al dpkg. Se ejecuta simplemente con el comando dselect en consola, nos aparecerán 7 opciones: access, update, select, install, config, remove y quit.

Access – La herramienta deselect permite acceder a una variedad de fuentes donde conseguir los programas, estas fuentes son configuradas a través de esta opción. En la tabla 5-2 se pueden ver las diferentes fuentes.

Tabla 5-2 Opciones de acceso del Dselect

Fuente	***Descripción***
cdrom	Instalación a través del cdrom
nfs	Instalación a través de un servidor nfs
harddisk	Instalación desde un disco duro no montado
mounted	Instalación desde un hdd montado
floppy	Instalación desde uno/varios diskette/s

Fuente	***Descripción***
ftp	Instalación desde un ftp
apt	Permite el acceso de diversas fuentes (mirar la sección del apt-get)

Update – Cuando se cambian los métodos de acceso o los paquetes tienen nuevas versiones, es necesaria esta opción para mantener la base de datos actualizada.

Select – Una vez la base de datos está actualizada se pueden seleccionar los paquetes a instalar. Las siguientes teclas son las básicas para moverse por este menú:

Tabla 5-3 Teclas de selección

Tecla	***Uso***
+	Añade el paquete para su instalación
-	Quita el paquete
=	Asigna el estado hold al paquete
:	Quita el estado hold
/<patron>	Buscar usando el patrón
\	Repite la última búsqueda
O	Recorre las opciones de ordenación
V	Cambia el estado del display
X	Quitar sin guardar los cambios
Q	Quitar guardando los cambios

Cuando un paquete se marca para su instalación o desinstalación, deselect revisa las dependencias que no están instaladas o que se pueden romper por su desinstalación, si esto sucede, se avisará con un mensaje de error recomendando no instalar/desinstalar el paquete seleccionado.

Install – Una vez seleccionados los paquetes lo siguiente es pulsar la opción de instalación desde el menú principal, esta opción obtiene los paquetes y los instala.

Config – Algunos paquetes requieren de la intervención del usuario para su instalación y configuración final, normalmente a través de unas opciones dadas para fijar unos parámetros determinados, para realizar este paso basta con seleccionar la opción del menú.

Remove – Luego de seleccionar los paquetes seleccionados para su desinstalación es necesario ir a esta opción para que los cambios tengan efecto

Quit – Con esta opción se sale de la aplicación.

Usando el apt-get

El apt-get es la herramienta por excelencia de Debian para la administración de paquetes, sin la necesidad de una interface como la de dselect y teniendo un abanico más amplio de opciones,

el apt-get al igual que el dselect, instalará automaticamente los paquetes así como sus dependencias.

Editando el fichero sources.list

Antes de que el apt-get pueda coger los paquetes para su instalación, tiene que saber de donde obtenerlos. El fichero sources.list tiene las direcciones de las ubicaciones desde donde obtener todos los paquetes. Este fichero también es usado por el dselect, está formado por un listado de fuentes con el siguiente formato para los binarios :

deb uri distribucion componente

...y para los fuentes:

deb-src uri distribución componente

URI (uniform resource identifier) es un superconjunto del familiar formato URL que la gran mayoría conoce, usa el siguiente formato:

protocolo://host/path

La sección //host del URI solamente se usa para los métodos HTTP y FTP, los cuatro tipos de acceso son:

- CD-ROM – Un cd-rom local
- File – Un directorio local
- FTP – Un servidor FTP
- HTTP – Un servidor WEB

Debian actualmente dispone de 3 ramas de desarrollo:

- Stable – Es la “oficial” y la que se recomienda para entornos de desarrollo/producción.
- Testing – Contiene los paquetes que están en cola para ser aceptados en la versión stable.
- Unstable – Paquetes que están siendo testados y probados. No se recomienda para entornos de producción, aunque para usuarios domésticos y los que quieran estar a la última esta es la mejor opción.

Dentro de estas ramas existen 3 componentes:

- Main – Los paquetes principales.
- Contrib – Paquetes secundarios.
- non-free – Paquetes que no son libres pero que pueden resultar interesantes/utiles. No son considerados parte de Debian.

Se recomienda poner los recursos más rápidos en la parte de arriba del fichero sources.list. Se pueden añadir comentarios a la lista usando el símbolo #, puede ser útil comentar los recursos por temática, paquetes concretos...

Un ejemplo del sources.list podría ser:

```
# See sources.list(5) for more information, especialy
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian unstable main contrib non-free
deb http://non-us.debian.org/debian-non-US unstable/non-US main contrib non-free
deb http://security.debian.org stable/updates main contrib non-free
#HelixCode
```

```
deb http://spidermonkey.helixcode.com/distributions/debian unstable main
# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable non-US
```

Actualizando los paquetes disponibles

La base de datos de Debian contiene una lista de todos los paquetes disponibles. Es útil, y casi necesario, actualizar esta lista a menudo, o cuando se realicen cambios al fichero sources.list. Para actualizar la base de datos se ejecuta el comando:

```
#apt-get update
```

Lo cual hará que el apt-get recorra los recursos del fichero sources.list y actualice la base de datos.

Instalando un paquete

Cuando se ordena la instalación de un paquete, el apt-get revisa primero si este ya fue descargado, si no lo fue, entonces el apt-get irá al primer recurso del sources.list a buscar la versión más nueva del programa, y si este tiene dependencias se añadirán a la lista de instalación.
Para instalar un programa se ejecuta el comando:

```
#apt-get install paquete
```

Por ejemplo:

```
#apt-get install ethereal
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  libpcap0
The following NEW packages will be installed:
  ethereal libpcap0
0 packages upgraded, 2 newly installed, 0 to remove and 202 not upgraded.
Need to get 1240kB of archives. After unpacking 3153kB will be used.
Do you want to continue? [Y/n]
Get:1 http://http.us.debian.org unstable/main ethereal 0.8.13-2 [1202kB]
Get:2 http://http.us.debian.org unstable/main libpcap0 0.4a6-3[38.6kB]
Fetched 1240kB in 16s (74.0kB/s)
Selecting previously deselected package libpcap0.
(Reading database ... 53531 files and directories currently installed.)
Unpacking libpcap0 (from .../libpcap0_0.4a6-3_i386.deb) ...
Selecting previously deselected package ethereal.
Unpacking ethereal (from .../ethereal_0.8.13-2_i386.deb) ...
Setting up libpcap0 (0.4a6-3) ...
Setting up ethereal (0.8.13-2) ...
```

Como se ve en el ejemplo, el ethereal requiere el paquete libpcap0, se pregunta al usuario si quiere que se instale el programa y sus dependencias, y acto seguido se procede a dicha instalación.

Actualizando paquetes

Una de las mejores características del sistema apt, es la posibilidad de actualizar todos los paquetes instalados en el sistema a la última versión disponible y en un solo paso. Para realizar esto, se introduce el comando:

```
#apt-get upgrade
```

Es necesario y muy conveniente, asegurarse de ejecutar el apt-get update antes de pasarlo para tener la base de datos actualizada. Dependiendo de la cantidad de programas instalados y de las novedades, el proceso llevará más o menos tiempo.

Borrando paquetes

Los paquetes pueden ser borrados con el apt-get al igual que con el dpkg, el comando para realizar esta operación:

```
#apt-get remove paquete
```

Actualizando la distribución

Aunque la instalación de Debian no es tan cómoda e intuitiva como otras distribuciones, tiene la ventaja de que sólo es necesaria la primera instalación, incluso cuando hay que cambiar a una versión superior del sistema. La versión actual puede ser actualizada a una nueva cuando sea lanzada con el apt-get. Para realizar esta operación se ejecuta el comando:

```
#apt-get dist-upgrade
```

La diferencia entre esta operación y la actualización normal es que el apt usa menos revisiones de dependencias y puede actualizar paquetes importantes a expensas de otros paquetes.

Limpiando los archivos de los paquetes

Cuando el apt-get instala un programa, guarda una copia del fichero deb en los directorios /var/cache/apt/archives y /var/cache/apt/archives/partial. Con el paso del tiempo estos directorios pueden llegar a ocupar un espacio importante e innecesario, para limpiar ambos se ejecuta el comando:

```
#apt-get clean
```

A veces puede ser útil guardar algunos archivos. Para limpiar solamente paquetes que no podrán ser instalados de nuevo porque no están disponibles, se puede usar el comando:

```
#apt-get autoclean
```

Las opciones del apt-get

Tabla 5-4 Opciones del apt-get

Opción	*Uso*
-h	Muestra la ayuda
-qq	Solamente muestra los errores en el proceso

Opción	***Uso***
-d	Sólo baja los ficheros, no los instala
-s	Simula la acción.Muestra la información como si realmente hiciera el proceso
-y	Responde "si" a todas las preguntas
-f	Continuar incluso si la revisión de integridad falla. A veces es útil para corregir problemas con las dependencias
-m	Continuar incluso si los paquetes no pueden ser localizados
-u	Muestra una lista de los paquetes actualizados
-b	Construye un paquete fuente después de descargarlo
-c=nombre_fichero	Lee el fichero de configuración especifico
-o=opcion	Hace uso de una opción especial

Usando Alien

Algunos paquetes están en otro formato diferente al deb de Debian, para solucionar este problema se creo la herramienta alien, que convierte estos formatos:

- Debian .deb
- Red Hat .rpm
- Slackware .tgz
- Stampede .slp

La sintaxis del comando es:

alien [opciones] paquete

Las opciones del comando están en la tabla 5-5.

Tabla 5-5 opciones del comando alien

Opción	Alternativa	Uso
-d	--to-deb	Por defecto, convierte en un paquete .deb
--patch=<filename>		Solamente usado con la opción -d. Especifica el fichero con el patch que debe ser usado
--nopatch		Sólo usado con la opción -d. Ningún patch se empleará
-r	--to-rpm	Crea un paquete rpm
-t	--to-tgz	Crea un paquete tgz
--to-slp		Crea un paquete stampede
-i	--install	Instala el programa tras la creación del paquete
-g	--generate	Desempaqueta el contenido del paquete pero no genera ninguno nuevo
-s	--single	Lo mismo que la opción -g pero no crea el directorio .orig

Opción	Alternativa	Uso
-c	--scripts	Incluye los scripts en el paquete
-k	--keep-version	No cambia la versión del nuevo paquete
--description=		Pone descripción al paquete creado
-h	--help	Muestra la ayuda
-v	--version	Muestra la versión

Por ejemplo para convertir el paquete wget (en rpm) a un deb, se usaría el comando:

```
# alien -d wget.rpm
```

Después se tendría disponible el wget.deb

Ejemplos prácticos

Ejemplo 1

Instalar un paquete usando dpkg –i con el nombre de un fichero disponible:

```
# dpkg -i ./hdparm_3.3-3.deb

(Reading database ... 54816 files and directories currently installed.)
Preparing to replace hdparm 3.3-3 (using hdparm_3.3-3.deb)
Unpacking replacement hdparm ...
Setting up hdparm (3.3-3) ...
```

Alternativamente, usar apt-install con el nombre del paquete. En este caso, el paquete procede de la localización o localizaciones configuradas en /etc/apt/sources.list.

Para este ejemplo, la localización es http://http.us.debian.org:

```
# apt-get install elvis
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
libncurses4 xlib6g
The following NEW packages will be installed:
elvis
2 packages upgraded, 1 newly installed, 0 to remove and 376 not upgraded.

Need to get 1678kB of archives. After unpacking 2544kB will be used.
Do you want to continue? [Y/n] y
Get:1 http://http.us.debian.org stable/main
  libncurses4 4.2-9 [180kB]
Get:2 http://http.us.debian.org stable/main
xlib6g 3.3.6-11 [993kB]
Get:3 http://http.us.debian.org stable/main
elvis 2.1.4-1 [505kB]
Fetched 1678kB in 4m11s (6663B/s)
(Reading database ... 54730 files and directories currently installed.)
Preparing to replace libncurses4 4.2-3 (using .../libncurses4_4.2-9_i386.deb) ...
Unpacking replacement libncurses4 ...
```

(la instalación continua...)

Ejemplo 2

Actualizar un paquete no es diferente de instalar uno. Sin embargo, se debería usar la opción -G cuando actualicemos con dpkg, para asegurarnos que no se efectue si una versión más reciente está realmente instalada.

Ejemplo 3

Usar dpkg –r o dpkg —purge para eliminar un paquete:

```
# dpkg --purge elvis
(Reading database ... 54816 files and directories
currently installed.)
Removing elvis ...
(purge continua...)
```

Ejemplo 4

Usar el comando dpkg –S para encontrar un paquete que contenga unos ficheros específicos. En este ejemplo, apt–get está contenido en el paquete apt:

```
# dpkg -S apt-get
apt: /usr/share/man/man8/apt-get.8.gz
apt: /usr/bin/apt-get
```

Ejemplo 5

Obtener información del estado del paquete, tales como versión, contenido, dependencias, integridad y estado de la instalación, usando dpkg –s:

```
# dpkg -s apt
Package: apt
Status: install ok installed
Priority: optional
Section: admin
Installed-Size: 1388
(listing continues...)
```

Ejemplo 6

Listar los ficheros de un paquete usando dpkg –L y procesar la salida usando grep o less:

```
# dpkg -L apt | grep '^/usr/bin'
/usr/bin
/usr/bin/apt-cache
/usr/bin/apt-cdrom
/usr/bin/apt-config
/usr/bin/apt-get
```

Ejemplo 7

Listar los paquetes instalados usando dpkg –l; si no se especifica todos los paquetes serán listados:

```
# dpkg -l xdm
ii xdm 3.3.2.3a-11 X display manager
```

Ejemplo 8

Usar dpkg –S para determinar desde que paquete se ha instalado un fichero en particular con su nombre:

```
# dpkg -S /usr/bin/nl
textutils: /usr/bin/nl
```

Tema 102.6 Administrando paquetes RPM

Introducción

En este capítulo se verá como administrar un sistema usando el administrador de paquetes RPM. Esto incluye comandos para instalar, actualizar y desinstalar programas, así como otras características, como saber la versión instalada, contenidos, dependencias, integridad del paquete, etc.

Los comandos que se verán en este tema son:

rpm
grep

Se verá la configuración del fichero:

/etc/rpmrc
/usr/lib/rpm/*

Este tema tiene un peso (importancia) de 8 de cara al examen final de la certificación LPI 101. El total de la suma de pesos de todos los temas es de 106.

Gestor de paquetes Red Hat

Hay que familiarizarse con las siguientes tareas: instalación del paquete, desinstalación del paquete, determinar la versión de dicho paquete y saber la versión del software que contiene, listar los archivos del paquete, listar los archivos de documentación de paquete, listar los archivos de configuración o los scripts de instalación o desinstalación, encontrar un determinado paquete que está instalado, saber que paquetes han sido instalados en el sistema (todos o un grupo determinado), averiguar en que paquete puede encontrarse un programa o archivo, verificar la integridad del paquete, saber la firma PGP o GPG del paquete y actualizarlo.
Requiere el uso de los comandos y programas: rpm, grep.

En ocasiones es difícil obtener el pedazo de código necesario para compilar correctamente el sistema, y a menos que se desee realizar cambios o leer el código, no existe una ventaja real en hacerlo. La mayoría del software está distribuido en formato binario, es decir, ya compilado.

El más popular gestor de paquetes que se utiliza con Linux es RPM, o Red Hat Package Manager. A pesar de ser creado por Red Hat, se utiliza en la mayoría de las distribuciones por defecto exceptuando Slackware, Debian, Gentoo... RPM es una de las contribuciones de Red Hat a la comunidad de software libre más conocidas y una de las que ahorraron a muchos administradores tiempo y esfuerzo.

Un sistema de gestión de paquetes mejora la distribución binaria gestionando el control de la versión, las dependencias con otros paquetes y su administración. Utilizando las herramientas del paquete, se puede comprobar la versión instalada, los archivos incluidos en el paquete, etc. RPM está compuesto por:

- Archivos del paquete (*`*.rpm`*)
- La base de datos RPM
- La herramienta *`rpm`*

Archivos del paquete (*.RPM)

Los archivos RPM se distribuyen para la mayoría de las aplicaciones. Un archivo RPM incluye las siguientes partes:

- Archivos de la aplicación comprimidos
- Nombre y versión del paquete
- Fecha de realización y fecha de publicación
- Descripción del paquete y de la aplicación
- Información de quién realizó el paquete
- MD5 “checksum” para verificar la integridad del paquete
- Otros paquetes requeridos (dependencias)

Como se puede observar, dentro de un paquete RPM se incluye mucha información. A través de los distintos archivos, se incluye toda la información necesaria para instalar y mantener el paquete. Los RPM siguen la siguiente tipología estándar:

package-version-patch.architecture.rpm

donde:

- package - Nombre de la aplicación instalada por el paquete.
- versión - Número de la versión de la aplicación.
- match - Número de "arreglo" del paquete. Si se produce un pequeño cambio o el administrador realiza una modificación en el paquete, este número se incrementa.
- architecture - la arquitectura del computador para la cual está realizado el paquete. Esto es muy importante ahora que Linux se ejecuta en tantas computadoras distintas. Algunos ejemplos: i386, i586, y i686 para Intel x86 y compatibles; sparc para Sun Sparc,; y alpha para Digital/Compaq Alpha.

Ejemplo:

ethereal -0.8.9-1.i386.rpm

Este paquete contiene la versión 0.8.9 de Ethereal, un paquete "sniffer" utilizado para reiniciar una red. Esta es la primera construcción de este paquete, y es para la plataforma i386 (Intel PC).

Un lugar para encontrar paquetes rpm de muchas aplicaciones es `www.rpmfind.net`

La base de datos RPM

La información sobre todos los paquetes instalados en el sistema se mantiene en una base de datos. Ésta se encuentra en el directorio `/var/lib/rpm`. Estos datos se utilizan para encontrar las dependencias, comprobar los ficheros que ya existen y verificar los paquetes instalados. Siempre que se utiliza el comando rpm se consulta la base de datos.

Ocasionalmente la base de datos tendrán inconsistencias y será necesario reconstruirla utilizando el comando `rpm` con `-rebuilddb`:

```
#rpm –rebuilddb
```

Cuando sucede una inconsistencia o se corrompe la base de datos, se reciben errores extraños al añadir o borrar paquetes.

La herramienta rpm

La herramienta rpm es una herramienta de la línea de comandos que se preocupa de RPM, por lo tanto no cabe confusión.

El comando `rpm` se utiliza para:

- Instalar paquetes
- Actualizar paquetes
- Borrar y desinstalar paquetes
- Consultar la base de datos RPM en busca de información
- Verificar el archivo del paquete
- Comprobar los archivos instalados
- Construir un paquete binario desde el código

La herramienta rpm actualiza errores del sistema realizando la instalación, desinstalación o actualización. Esto ayuda a proteger contra la instalación de un paquete inútil o dañino para otra aplicación. Estos diagnósticos incluyen la comprobación de:

- Suficiente espacio libre para el paquete
- Los archivos existentes no serán sobrescritos
- Todas las dependencias están instaladas.

Validando la integridad del paquete

RPM incluye funciones que permiten comprobar la integridad de un paquete para confirmar que ha sido correctamente descargado y no es falso. Eso se realiza utilizando el algoritmo MD5 y la herramienta GnuPG. MD5 es un algoritmo matemático que se utiliza para comprobar que un archivo no se ha modificado. Cuando se comprueba el archivo, el algoritmo extrae un número de comprobación (checksum), y si este número coincide con el generado por el archivo antes de la descarga, el archivo no está corrupto. La herramienta GnuPg es un paquete de encriptación de llave pública que puede ser usado para comprobar la autenticidad del código de un archivo o documento y encriptar las comunicaciones. GnuPG se instala por defecto en los sistemas Red Hat.

Las opciones `-k` o `--checksig` comprueban la integridad de un paquete utilizando MD5 y/o GnuPG.

```
rpm -k package_file.rpm
```

Para que funcione correctamente, se deben seguir los siguientes pasos:

1.- Instalación the la aplicación GnuPG si no está instalada. Se puede conseguir desde *www.gnupg.org*.

2.- Recuperación de la llave pública desde el administrador de la aplicación que se desee comprobar. Esto está generalmente disponible en los sitios Web o FTP. Por ejemplo, Red Hat está disponible en su FTP SITE y se denomina RPM-GPG-KEY.

3.- Añadir la llave pública apropiada al conjunto de llaves utilizando el comando gpg -import. Por ejemplo:

```
# gpg --import RPM-GPG-KEY
```

(se omite la salida por pantalla intencionadamente)

Si el paquete se valida correctamente, rpm escribirá un mensaje similar al siguiente:

```
# rpm -K wget-1.5.3-6.src.rpm
wget-1.5.3-6.src.rpm: md5 gpg OK
```

Si el paquete no se valida, el mensaje será el siguiente:

```
# rpm -K wget-1.5.3-10.i386.rpm
wget-1.5.3-10.i386.rpm: rpmReadSignature failed
```

Algunos paquetes utilizan PGP para comprobar la integridad mientras otros utilizan GnuPG. PGP se puede descargar desde *www.pgpi.com* .

Instalando Paquetes

El Gestor de Paquetes Red Hat (Red Hat Package Manager o RPM) simplifica muchísimo las instalaciones de software nuevo. Añadir un nuevo paquete al sistema puede ser tan simple como escribir el siguiente comando:

rpm -i *fichero_paquete*.rpm

o:

rpm --install *fichero_paquete*.rpm

CONSEJO PARA EL EXAMEN: Muchas de las opciones disponen de una versión corta y otra larga, por ejemplo, -i y --install. Debemos asegurarnos de conocer las dos versiones en el examen y recordar si van en mayúsculas o minúsculas.

La herramienta rpm también es capaz de instalar varios paquetes al mismo tiempo. Por ejemplo:

rpm -i *primer_fichero_paquete*.rpm *segundo_fichero_paquete*.rpm *tercer_fichero_paquete*.rpm
o:
rpm -i *.rpm

Se pueden utilizar comodines para instalar paquetes, pero no para eliminarlos.

La instalación simultánea de varios paquetes puede ser práctica cuando cada paquete va siendo instalado después de aquellos de los que depende. Para cada paquete instalado se realizan las verificaciones estándar de coherencia. Hay que tener en cuenta que el fallo en la instalación de un solo paquete podría abortar la ejecución de todo el comando y hacer que no se instalase ninguno.

La herramienta rpm también puede obtener paquetes desde Internet, lo que nos ahorraría uno o dos pasos en la instalación. El formato de la dirección del paquete es el mismo que utilizaríamos en un navegador web. La herramienta soporta tanto FTP anónimo como autorizado. Por ejemplo:

```
rpm -i ftp://rpmfind.net/linux/redhat/redhat-7.0/i386/en/RedHat/RPMS//libpcap-0.4-29.i386.rpm
```

o:

```
rpm -i http://rpmfind.net/linux/redhat/redhat-7.0/i386/en/RedHat/RPMS//libpcap-0.4-29.i386.rpm
```

Al instalar paquetes podemos añadir algunas opciones prácticas como -v y -h (--hash).
La opción -v muestra el nombre del paquete que está siendo instalado.

```
# rpm -iv libpcap-0.4-19.i386.rpm
libpcap-0.4-19
```

La opción -h va mostrando caracteres '#' (hash) mientras el paquete se está instalado para indicar la marcha de la instalación. Esto puede ser práctico cuando instalemos paquetes muy grandes.

```
# rpm -ivh libpcap-0.4-19.i386.rpm
```

```
libpcap
##################################################
```

Como se comentó anteriormente, rpm verifica todas las dependencias necesarias antes de instalar o eliminar un paquete. Si se encontrase algún problema, se mostraría un mensaje de error parecido al siguiente:

```
# rpm -ivh ethereal-0.8.9-1.i386.rpm
error: failed dependencies:
libpcap >= 0.4 is needed by ethereal-0.8.9-1
```

Esto significa que el paquete *ethereal-0.8.9-1.i386.rpm* necesita que esté instalado el paquete *libpcap* de versión 0.4 o mayor.

Si un paquete intentase sobreescribir un fichero existente, se mostraría un mensaje de error parecido a este:

```
# rpm -ivh libpcap.rpm
file /usr/lib/libpcap.a from install of libpcap-0.4a6-35 conflicts with file from package
libpcap-0.4-19
file /usr/man/man3/pcap.3.gz from install of libpcap-0.4a6-35 conflicts with file from
package libpcap-0.4-19
```

Por defecto, rpm no sobreescribirá ficheros de otros paquetes. En caso de que existiese una buena razón para continuar con la operación a pesar de los mensajes de advertencia, podrían utilizarse algunas de las opciones de sobreescritura que proporciona rpm y que se muestran a continuación:

--force—Fuerza a rpm a sobreescribir ficheros o paquetes existentes.
--nodeps—Se salta la verificación de dependencias. Esto es práctico si el paquete del que depende la instalación ha sido ya instalado mediante otro método como, por ejemplo, compilándolo a partir de los fuentes.
--replacefiles—Sobreescribe ficheros propiedad de otros paquetes.

Siguiendo el ejemplo anterior, asumamos que la librería *libpcap* ha sido compilada directamente desde los fuentes y no instalada a partir de un RPM. Para instalar el paquete Ethereal habría que utilizar el siguiente comando:

```
rpm -ivh --nodeps ethereal-0.8.9-1.i386.rpm
```

En esta ocasión, el paquete se instalará sin que aparezca ningún error. De todas formas, para que la aplicación funcione correctamente, la dependencia necesaria tiene que estar instalada correctamente. Saltarse los avisos es una práctica arriesgada que hay que tomar con mucha precaución considerando las posibles consecuencias, sobre todo en paquetes importantes.

Actualizando Paquetes

Si en el mundo de software hay algo seguro, es que continuamente nos encontraremos con

actualizaciones y nuevas versiones de los paquetes. RPM facilita esta labor realizando la eliminación de la versión antigua y la instalación de la nueva en un solo paso. Para actualizar un paquete simplemente utilizaremos la opción -U:

```
rpm -U fichero_paquete.rpm
```

o:

```
rpm --upgrade fichero_paquete.rpm
```

De la misma forma que con las opciones de instalación, en este caso también es recomendable el uso de los parámetros -v y -h.

Como la opción de actualización empieza eliminando la versión anterior, se guardarán los ficheros de configuración que contengan la extensión *.rpmsave*. Después, en la segunda fase se instalará la nueva versión realizándose en este paso las verificaciones estandar. Si no hubiese ninguna versión antigua del paquete en el sistema, rpm continuará adelante e instalará la nueva versión como si de una instalación nueva se tratase. Muchos administradores utilizan siempre la opción -U para cualquier tipo de instalación, de esta forma tienen los dos casos cubiertos ya que rpm intentará primero actualizar.

Otra opción práctica es -F (o --freshen), la cual actualiza un paquete sólo si ya existe una versión anterior instalada. De esta forma, se pueden actualizar un buen número de paquetes de una sola vez.

```
rpm -Fvh *.rpm
```

Este comando intentará actualizar cualquier paquete ya instalado en el sistema del que exista un fichero de paquete nuevo en el directorio actual. Combinando esto con el hecho de que la mayoría de las distribuciones, mantienen en sus sitios web un directorio con los paquetes que han sido actualizados desde la versión inicial, nos encontramos con una buena técnica para mantener nuestros sistemas actualizados.

Desinstalando Paquetes

Para eliminar un paquete del sistema se utilizan las opciones -e o –uninstall:

```
rpm -e nombre_paquete
```

o:

```
rpm --uninstall nombre_paquete
```

A la hora de desinstalar un paquete, debemos recordar que hemos de utilizar el nombre del paquete y no el nombre del fichero que contenía.
También hay que tener en cuenta que los comodines no funcionan en la eliminación de paquetes.
Las dos opciones realizan la misma función, simplemente, algunas personas encuentran que la opción -uninstall es más fácil de recordar.
Durante la desinstalación de un paquete, rpm realiza los chequeos de dependencias habituales y, por defecto, no permitirá desinstalar un paquete si existiese otro dependiente de éste.

```
# rpm -e libpcap
error: removing these packages would break dependencies:
libpcap >= 0.4 is needed by ethereal-0.8.9-1
```

Podemos utilizar la opción --nodeps para saltarnos estos avisos.
Cuando se desinstala un paquete, rpm guarda todos los ficheros de configuración modificados. De esta forma, si más adelante volviésemos a reinstalar el paquete no tendríamos que volverlo a configurar.

```
# ls /etc/pine*
/etc/pine.conf
# rpm -e pine
# ls /etc/pine*
/etc/pine.conf.rpmsave
```

Consultando la base de datos de RPM

La base de datos de RPM se guarda en /var/lib/rpm y contiene información sobre cada paquete instalado en el sistema. Estos datos pueden ser consultados para recopilar información práctica para la administración del sistema. Todas las opciones de consulta se realizan con el comando rpm y la opción -q.

Listando los paquetes instalados

La consulta más básica que podemos hacer es la de la versión de un paquete instalado:

rpm -q *nombre_paquete*

o:

rpm --query *nombre_paquete*

Por ejemplo:

```
# rpm -q kernel
kernel-2.2.14-5.0
```

Para listar todos los paquetes instalados en el sistema se utiliza la opción -a:

```
rpm -qa
```

Se puede combinar esta opción con *grep* para ver todos los paquetes instalados de un determinado grupo o familia.
El siguiente ejemplo utiliza rpm -qa para encontrar todos los paquetes y *grep* busca en esta lista todos los que contengan la palabra *kernel*:

```
# rpm -qa | grep kernel
kernel-headers-2.2.14-5.0
kernel-2.2.14-5.0
kernel-pcmcia-cs-2.2.14-5.0
kernel-utils-2.2.14-5.0
```

Averiguando que paquete instaló un determinado fichero

Hay ocasiones en las que no estamos seguros de que paquete instaló un determinado fichero y necesitamos averiguarlo. La opción -f nos proporciona ésta información.

rpm -qf *filename*

Por ejemplo:

```
# rpm -qf /etc/bashrc
bash-1.14.7-22
```

Esto nos indica que el fichero *bashrc* fue instalado por el paquete *bash-1.14.7-22*.

Listando los ficheros de un paquete

Para listar todos los ficheros instalados por un paquete se utiliza la opción -l.

rpm -ql *nombre_paquete*

Por ejemplo, para listar todos los ficheros del paquete openssh-clients teclearíamos lo siguiente:

```
# rpm -ql openssh-clients
/etc/ssh/ssh_config
/usr/bin/slogin
/usr/bin/ssh
/usr/bin/ssh-add
/usr/bin/ssh-agent
/usr/man/man1/slogin.1.gz
/usr/man/man1/ssh-add.1.gz
/usr/man/man1/ssh-agent.1.gz
/usr/man/man1/ssh.1.gz
```

Para listar los ficheros que serán instalados con el paquete se utiliza adicionalmente la opción -p.

rpm -qlp *fichero_paquete.rpm*

Mostrando información de un paquete

Para ver la descripción y otras informaciones sobre un determinado paquete se utiliza la opción -i.

rpm -qi *nombre_paquete*

Por ejemplo, para obtener información sobre el kernel Linux instalado en nuestro sistema, escribiríamos lo siguiente:

```
# rpm -qi kernel
```

(... a continuación se muestra una información muy completa sobre el paquete ...)

Si quisiéramos obtener ésta misma información pero de un paquete que aun no ha sido instalado, utilizaríamos la opción -p:

rpm -qip *fichero_paquete*.rpm

Mostrando los Scripts de un paquete

Algunos paquetes incluyen scripts (ficheros de comandos shell) que se ejecutan antes o después de la instalación de los mismos. Para ver estos scripts se utiliza el siguiente comando:

rpm -qp --scripts *fichero_paquete*.rpm

Por ejemplo, para ver los scripts que se ejecutarán cuando el paquete *kernel-2.2.14-5.0.i686.rpm* sea instalado, escribiríamos lo siguiente:

```
# rpm -qp –scripts /mnt/cdrom/RedHat/RPMS/kernel-2.2.14-5.0.i386.rpm
```

Verificando ficheros de paquetes

A veces es necesario verificar si un fichero ha cambiado desde la instalación de un paquete. Podríamos haber hecho algunos cambios en algún fichero de configuración y ahora necesitar averiguar cuales fueron los ficheros modificados. También podría interesarnos verificar si los ficheros extraídos de determinados paquetes han sido modificados por hackers o por algún virus. La herramienta rpm nos proporciona esta funcionalidad a través de la opción de verificación -V.

rpm -V *nombre_paquete*

Cuando realiza una verificación, rpm realiza una serie de comprobaciones en los ficheros instalados y muestra aquellos que han cambiado desde la instalación. Cada fichero tiene una entrada asociada en la base de datos RPM donde se almacena una firma MD5, el tamaño del mismo, un puntero a un enlace simbólico (symbolic link), la fecha y hora de modificación, el usuario y grupo propietarios y los permisos y tipo del fichero. Si el fichero fuese un dispositivo, también se comprueban los números de dispositivo mayor y menor. La siguiente tabla muestra la salida del comando según que característica haya sido cambiada.

Tabla 6-1 Características de verificación de paquetes

Ítem	*Significado*
.	No se ha encontrado ningún cambio en esta característica
5	Cambio en la firma MD5
S	Cambio en el tamaño del fichero
L	Cambio en el enlace simbólico (Symbolic link)

Ítem	*Significado*
T	Cambio en fecha u hora de modificación
G	Cambio en grupo propietario
U	Cambio en usuario propietario
D	Cambio en los números mayor/menor de dispositivo
M	Cambio en permisos y/o tipo de fichero

Por ejemplo, para ver que ficheros de configuración instalados por el paquete han sido modificados desde la instalación, escribiríamos lo siguiente:

```
# rpm -V setup
S.5....T c /etc/hosts.allow
......G. c /etc/profile
S.5....T c /etc/services
```

Esta salida nos muestra que han cambiado el tamaño, firma MD5 y hora de modificación en los ficheros *hosts.allow* y *services*. En cambio, al fichero *profile* sólo se le ha cambiado el grupo propietario.

Para verificar un paquete a partir del nombre de fichero del paquete, se utiliza la opción -p.

rpm -Vp *fichero_paquete*.rpm

Se pueden verificar todos los paquetes instalados en el sistema mediante la opción -a. Esta es una forma rápida de averiguar que ficheros han cambiado desde que se instaló el sistema.

rpm -Va

Creando paquetes binarios a partir de paquetes de fuentes

No todos los paquetes RPM distribuyen ficheros binarios. Algunos distribuyen el código fuente y los scripts de instalación que permiten la creación de paquetes RPM binarios personalizados, de esta forma podemos cambiar u optimizar un determinado paquete para nuestras necesidades o hardware particular.
Estos paquetes utilizan un sistema de nombres ligeramente diferente ya que no dependen de una arquitectura de sistema en particular.

paquete-version-patch.src.rpm

Los RPMs de fuentes se instalan, igual que los binarios, con el comando *rpm -i*.
La instalación coloca los diferentes componentes del paquete dentro de la jerarquía de directorios /usr/src/redhat.
La siguiente tabla muestra la función de los directorios de este árbol.

Tabla 6-2 Subdirectorios en el árbol /usr/src/redhat

Directorio	***Función***
/usr/src/redhat/SOURCES	Contiene el código fuente de la aplicación
/usr/src/redhat/SPECS	Contiene el fichero spec de la aplicación
/usr/src/redhat/BUILD	Donde se compila en código fuente
/usr/src/redhat/RPMS	Contiene el RPM binario final
/usr/src/redhat/SRPMS	Contiene el RPM fuente creado por el proceso

El fichero spec de un paquete controla como se compila el mismo y que scripts se ejecutarán cuando sea instalado o desinstalado.
El nombre de este fichero es /usr/src/redhat/SPECS/nombre_paqute.spec.
El fichero spec tiene ocho secciones como se muestra en la siguiente tabla:

Tabla 6-3 Secciones del fichero spec

Sección	***Descripción***
Header (Cabecera)	Información general, nombre, versión, etc
Prep (Preparación)	Scripts que realizarán las tareas necesarias antes del proceso de compilación
Buid (Construcción)	Comandos para construir el fichero spec y para compilar el código fuente
Install (Instalación)	Comandos necesarios para instalar el software en un sistema
Clean (Limpieza)	-Opcional- Comandos para limpiar el entorno de compilación en el caso de que se volviese a generar éste paquete
Optional Install Uninstall Scripts (Scripts opcionales de instalación y desinstalación)	Scripts opcionales que se pueden ejecutar en el proceso de instalación y desinstalación del paquete
File List (Lista de ficheros)	Lista de ficheros del paquete
Changelog (Registro de cambios)	Registro de los cambios que se hagan en el paquete

CONSEJO PARA EL EXAMEN: Debemos saber como construir un paquete binario a partir de un paquete de fuentes, incluyendo las modificaciones del fichero *spec*.

Esta es una sección **build** de ejemplo sacada del fichero spec de la aplicación *wget*:

```
%build
#./configure --prefix=/usr --sysconfdir=/etc
%configure --sysconfdir=/etc
make
```

Si quisiéramos hacer algunos cambios en el proceso de compilación deberíamos hacerlos aquí. Esto es lo que nos permite personalizar el paquete de forma que se adapte a nuestras necesidades. Una vez que hemos realizado todas las modificaciones oportunas hay que construir el paquete binario. Esto se hace con *rpm* y la opción -ba.

```
rpm -ba package.spec
```

Una vez completado el proceso, el paquete binario quedará en el directorio /usr/src/redhat/RPMS.

Ejemplos prácticos

Ejemplo 1
Para instalar un paquete nuevo, simplemente se utiliza el comando *rpm -i* seguido del nombre de un archivo de paquete. Si el nuevo paquete dependiese de otro paquete, la instalación daría un error como el siguiente:

```
# rpm -iv netscape-communicator-4.72-3.i386.rpm
error: failed dependencies:
netscape-common = 4.72 is needed by
netscape-communicator-4.72-3
```

Para corregir este tipo problemas, debemos instalar primero aquellos paquetes que satisfagan esas dependencias, en este ejemplo, *netscape-communicator* depende de *netscape-common*, por tanto debemos instalar primero este último paquete:

```
# rpm -iv netscape-common-4.72-3.i386.rpm
netscape-common

# rpm -iv netscape-communicator-4.72-3.i386.rpm
netscape-communicator
```

Ejemplo 2
Se puede actualizar un paquete existente a una nueva versión mediante la opción –U. El modo de actualización es realmente es un caso especial del modo de instalación, donde los paquetes existentes pueden ser reemplazados por las nuevas versiones. La opción -U instalará un paquete incluso si este no estuviese ya instalado, en este caso se comportaría de la misma forma que la opción -i:

```
# rpm -U netscape-common-4.72-3.i386.rpm
```

Ejemplo 3:
La eliminación de un paquete es lo contrario de la desinstalación del mismo y, por tanto, tiene las mismas restricciones de dependencia:

```
# rpm -e netscape-common
error: removing these packages would break dependencies (la eliminación de estos paquetes
        rompería algunas dependencias):
netscape-common = 4.72 is needed by
netscape-communicator-4.72-3
```

Ejemplo 4
Para determinar la versión del software contenido en un fichero RPM se utilizan las opciones de consulta e información del paquete:

```
# rpm -qpi xv-3.10a-13.i386.rpm | grep Version
Version : 3.10a Vendor: Red Hat Software
```

Cuando los paquetes estén ya instalados, se omite la opción -p y se especifica el nombre del paquete en lugar del nombre de fichero del paquete:

```
# rpm -qi kernel-source | grep Version
Version : 2.2.5 Vendor: Red Hat Software
```

Ejemplo 5

Utilizar el modo de consulta y listar los ficheros contenidos en un paquete:

```
# rpm -qlp gnucash-1.3.0-1.i386.rpm
/usr/bin/gnc-prices
/usr/bin/gnucash
/usr/bin/gnucash.gnome
/usr/doc/gnucash
/usr/doc/gnucash/CHANGES
(...el listado continúa ...)
```

Para un paquete ya instalado, entrar en modo consulta y utilizar la opción -l además del nombre del paquete:

```
# rpm -ql kernel-source
/usr/src/linux-2.2.5/COPYING
/usr/src/linux-2.2.5/CREDITS
/usr/src/linux-2.2.5/Documentation
/usr/src/linux-2.2.5/Documentation/00-INDEX
/usr/src/linux-2.2.5/Documentation/ARM-README
(...el listado continúa ...)
```

Ejemplo 6
Listar los ficheros de documentación en un paquete:

```
# rpm -qd at
/usr/doc/at-3.1.7/ChangeLog
/usr/doc/at-3.1.7/Copyright
/usr/doc/at-3.1.7/Problems
/usr/doc/at-3.1.7/README
/usr/doc/at-3.1.7/timespec
/usr/man/man1/at.1
/usr/man/man1/atq.1
/usr/man/man1/atrm.1
/usr/man/man1/batch.1
/usr/man/man8/atd.8
/usr/man/man8/atrun.8
```

Para indicar el nombre de fichero del paquete se usa la opción –p.

Ejemplo 7

Listar los scripts o ficheros de configuración de un paquete:

```
# rpm -qc at
/etc/at.deny
/etc/rc.d/init.d/atd
```

Ejemplo 8

Determinar desde que paquete en particular fue instalado un determinado fichero. Por supuesto, no todos los ficheros proceden de paquete:

```
# rpm -qf /etc/issue
file /etc/issue is not owned by any package (este mensaje indica que ese fichero no es
        propiedad de ningún paquete)
```

Para aquellos ficheros que son miembros de algún paquete, el resultado será parecido al siguiente:

```
# rpm -qf /etc/aliases
sendmail-8.9.3-10
```

Ejemplo 9

Listar los paquetes que han sido instalados en el sistema (todos o un subconjunto):

```
# rpm -qa
```
(... a continuación se listarán centenares de paquetes ...)

PREGUNTAS TEST

1. **¿**Cual es el sistema de ficheros por defecto en las particiones Linux?

2. ¿Que tipo de partición puede ser bootable –capaz de cargar el sistema operativo al arranque del equipo-?

3. ¿Cual es el nombre de dispositivo utilizado por la segunda partición del segundo disco IDE?

4. ¿Donde se almacena la información sobre etiquetas, bloques y tablas de inodos?

5. ¿Que tipo de sistema de ficheros se utiliza para acceder a sistemas remotos?
   A. ext2
   B. hpfs
   C. swap
   D. nfs

6. ¿Cual de los dispositivos siguientes representaría la tercera partición del segundo disco IDE?
   A. /dev/hdb3
   B. /dev/sdc2
   C. /dev/hdc2
   D. /dev/hda5

7. ¿Cual de los siguientes elementos es el puntero que señala la ubicación de los datos en los ficheros?
   A. Superbloque
   B. Inodo
   C. Partición
   D. Sistema de Fichero

8. ¿Que tipo de partición puede contener discos lógicos?
   A. Primaria
   B. Extendida
   C. Swap
   D. Root

9. ¿Que tipo de Sistema de Ficheros se usa en los sistemas Linux?
   A. ext2
   B. hpfs
   C. swap
   D. nfs

10. ¿Que tipo de Sistema de Ficheros proporciona memoria virtual en los sistemas Linux?
    A. ext2
    B. hpfs
    C. swap
    D. nfs

11.¿Que tipo de Sistema de Ficheros se utiliza en los sistemas OS/2?
A. ext2
B. hpfs
C. swap
D. nfs

12.¿Cuantas particiones primarias y extendidas pueden crearse en un disco duro?
A. 1
B. 2
C. 3
D. 4

13.¿Cuantas particiones primarias pueden crearse en un disco duro?
A. 1
B. 2
C. 3
D. 4

14. Que comando genera el fichero Makefile para tu sistema ?
A. ./gen
B. ./genmake
C. ./configure
D. ./config

15.Que comando instala software compilado ?
A. make
B. ./install
C. make setup
D. make install

16.Que comando se usa para generar el fichero /etc/ld.so.cache?
A. ldcache
B. ldupdate
C. ldconf
D. ldconfig

17.Que método de compilado crea ejecutables de menor tamaño ?
A. dinámico
B. unlinked
C. varible
D. Estático

18.Qué herramienta se usa para convertir los paquetes de un sistema a otro?
A. alien
B. dpkg
C. apt
D. Pconvert

19.Qué comando se usa para instalar un paquete de Debian?
A. apt --install <packagename>

B. dpkg --install <packagename>
C. apt-get -I <packagename>
D. rpm -i <packagename>

20.Qué comando desinstala un paquete Debian, incluyendo los ficheros de configuración?
A. dpkg --remove <packagename>
B. dpkg -e <packagename>
C. apt-get purge <packagename>
D. dpkg -P <packagename>

21.Qué herramienta provee una interfaz facil de usar para acceder al manejo de paquetes Debian?
A. dselect
B. apt-get
C. dpkg
D. Gnorpm

22.Para cambiar los sources para apt-get, se edita el fichero:
A. sources.list
B. apt.sources
C. sources.apt
D. dpkg.sources

23.Qué parametro de apt-get actualiza la base de datos de los paquetes disponibles?
A. upgrade
B. refresh
C. reload
D. Update

24.La herramienta apt-get que fuentes de las siguiente soporta? (seleccionar todas las que sean de aplicación)
A. FTP
B. HTTP
C. NFS
D. CD-ROM

25.Qué comando borra los paquetes antiguos de los archivos Debian?
A. dpkg -clean
B. apt-get autoclean
C. dpkg -autoclean
D. Dselect

26.Cuál de los siguientes formatos de paquetes soporta alien?
A. RPM
B. .deb
C. BSD
D. .tgz

27. Qué comando convierte un paquete RPM en formato Debian?
A. alien -r package.rpm
B. alien -t package.rpm

C. alien -d package.deb
D. alien -d package.rpm

28.¿Que sistema de paquetización utiliza Red Hat?
A. rpm
B. deb
C. tgz
D. rhp

29.¿Que opción de *rpm* se utiliza cuando se reciben mensajes extraños que hacen suponer que la base de datos rpm está corrupta?
A. rpm –fixdb
B. rpm –rebuilddb
C. rpm –updatedb
D. rpm –regendb

30.¿Que métodos de control de integridad de paquetes soporta RPM? (Selecciona todos los que procedan)
A. MD5
B. 3DES
C. PGP
D. GnuPG

31.Hay que utilizar el comando _____ para instalar el paquete llamado *processor-4.2.i386.rpm*.

32.¿Que comando o comandos se utilizan para eliminar un paquete RPM?
A. rpm --uninstall <nombre_de_paquete>
B. rpm --remove <nombre_de_paquete >
C. rpm -e <nombre_de_paquete >
D. rpm -u <nombre_de_paquete >

33.¿Cual de estos ficheros indica como fue compilado un paquete fuente RPM?
A. Makefile
B. fichero spec
C. fichero config
D. fichero .conf

RESPUESTAS TEST

1. Las particiones Linux usan por defecto el Sistema de Ficheros ext2.

2. Solo las particiones primarias son bootables –capaces de cargar un sistema-.

3. El nombre de dispositivo /dev/hdb2 es el utilizado para la segunda partición del segundo disco IDE.

4. La información sobre el disco se almacena en el superbloque del sistema.

5. **D.** Para acceder a sistemas remotos se utiliza el Sistema de Ficheros nfs (network file system). Para más información, mira la sección "Tipos de Sistemas de Ficheros".

6. **A.** La tercera partición del segundo disco es /dev/hdb3. La b especifica el segundo disco y el 3 se refiere a la tercera partición. Para más información, mira la sección "Tipos de Sistemas de Ficheros".

7. **B.** El inodo es un puntero que identifica la situación de los datos en el Sistema de Ficheros. Para más información, mira la sección "Consideraciones durante la creación de un Sistema de Ficheros".

8. **B.** Los discos lógicos existen dentro de las particiones extendidas. Para más información, mira la sección "Tipos de particiones".

9. **A.** El sistema de ficheros ext2 es utilizado por los sistemas Linux. Para más información, mira la sección "Tipos de Sistemas de Ficheros".

10. **C.** Los Sistemas de Ficheros Swap proporcionan memoria virtual en los sistemas Linux. Para más información, mira la sección "Tipos de Sistemas de Ficheros".

11. **B.** El Sistema de Ficheros hpfs se utiliza en los sistemas OS/2. Para más información, mira la sección "Tipos de Sistemas de Ficheros".

12. **D.** Solo pueden existir cuatro particiones en total –primarias y extendidas- en un solo disco. Para más información, mira la sección "Tipos de particiones".

13. **D.** Solo pueden existir cuatro particiones primarias en un solo disco. Para más información, mira la sección "Tipos de particiones".

14. C

15. D

16. D

17. A

18. A. La herramienta alien convierte paquetes de ficheros. La herramienta dpkg se usa para manipular paquetes en Debian.

19.B. El comando dpkg --install instala paquetes .deb. La herramienta rpm se usa para paquetes RPM. No hay ninguna opción -I para apt-get.

20.D. dpkg -P purga el paquete, desinstalando todos los ficheros incluyendo los de configuración.

21.A. La herramienta dselect tiene toda la funcinalidad de dpkg, pero usa una interfaz gráfica basada en caracteres en lugar de línea de comandos. apt-get se usa para recuperar e instalar paquetes. gnorpm es un frente gráfico para la herramienta rpm.

22.A. Los sources para apt-get están almacenados en el fichero sources.list. Las otras opciones no son válidas.

23.D. El comando update chequea todos los sources en el fichero sources.list y actualiza la base de datos de paquetes. El comando upgrade dice a apt-get que baje e instale todos los paquetes que son nuevos y aquellos instalados en el sistema. Las otras opciones no son válidas.

24.A, B, C y D. La herramienta apt-get puede obtener paquetes desde NFS local y de la unidad de CD-ROM y por medio de internet (FTP y HTTP)

25.B. El parámetro autoclean elimina sólo los viejos paquetes que no se pueden recuperar. Las otras opciones no son válidas.

26.A, B y D. alien soporta formatos de paquetes de Red Hat, Debian, y Slackware, pero no de BSD.

27.D. La opción -d indica a alien que cree un paquete debian. La opción -r especifica RPM, y la opcion -t especifica .tgz.

28.A. El sistema de paquetización RPM fue creado por Red Hat. Debian utiliza los paquetes .deb.

29.**B.** En ocasiones puede corromperse la base de datos RPM, en estos casos el comando *rpm --rebuilddb* intentará reconstruir la base de datos. El resto de las opciones son incorrectas.

30.**A, C, y D.** RPM soporta éstos tres métodos.

31.Tanto rpm -i processor-4.2-i386.rpm como rpm --install processor-4.2-i386.rpm son correctas.

32.**A y C.** Tanto *rpm --uninstall* como *rpm -e* eliminan paquetes RPM. El resto de las opciones son incorrectas.

33.**10. B.** El fichero *spec* es el que contiene las opciones de compilación. Se utiliza un Makefile para compilar código que no está en formato RPM.

Bibliografia y enlaces recomendados

LPIC 1 Certification Bible (Bible) by Angie Nash, Jason Nash
John Wiley & Sons; Bk&CD-Rom edition (July 1, 2001) ISBN: 0764547720

LPI Linux Certification in a Nutshell by Jeffrey Dean
O'Reilly & Associates; 1st ed edition (May 15, 2001) ISBN: 1565927486

CramSession's LPI General Linux Part 1 : Certification Study Guide
CramSession.com; ISBN: B000079Y0V; (August 17, 2000)

Referencias Unix Reviews
http://www.unixreview.com/documents/s=7459/uni1038932969999/

Página LPI: www.lpi.org

Apuntes IBM: http://www-106.ibm.com/developerworks/edu/l-dw-linux-lpir21-i.html

Manuales GPL: http://www.nongnu.org/lpi-manuals/