

Administración/Configuración de Linux - Gestionando el software

Cubre el objetivo del Linux Professional Institute (LPI) 1.102.4

Tabla de Contenidos

- 1 Introducción
- 2 Paquetes de software
 - 2.1 Conceptos importantes sobre sistemas de paquetes
 - 2.2 Depósitos o almacenes de software
- 3 El sistema de paquetes Debian
 - 3.1 Depósitos de software .deb
 - 3.2 Herramientas del sistema de paquetes Debian
 - 3.3 Comprobando el software instalado
 - 3.4 Instalando paquetes
 - 3.5 Eliminando paquetes
 - 3.6 Actualizando el sistema
 - 3.7 Ejemplo paso a paso

1 Introducción

[Ocultar conocimiento avanzado](#)

Por defecto, un sistema Linux viene con cientos de programas instalados, sin embargo, es frecuente que, como usuario de Linux, tengas que instalar nuevos programas.

En este guión de prácticas se **explica cómo instalar nuevos programas**.

Para ello primero hay que familiarizarse con el concepto de **paquete de software** (*software package*).

2 Paquetes de software

Los programas en Linux son muy **modulares**. Eso significa que raramente un sólo programa contiene todo lo que necesita para ejecutarse. La mayoría de los programas tienen **dependencias**.

Las **dependencias** son un conjunto de software, que tiene que estar instalado también en el sistema, para que un programa dado pueda funcionar bien.

Gestionar bien las dependencias de los programas es uno de los grandes retos de administrar un sistema Linux. Al instalar un programa, se pueden obtener mensajes de error indicando que para instalar el programa A es necesario tener instalado el programa B o las bibliotecas C y D. Esto se conoce como el **infierno de las dependencias** (*dependency hell*), y es una de las causas por las que en el pasado, los usuarios noveles no usaban Linux.

Hoy en día, prácticamente todas las distribuciones de Linux incluyen algún mecanismo para gestionar las dependencias. Todos ellos se basan en los conceptos de **paquete de software** (*software package*), y **depósito o almacén de software** (*software repository*).

Al mecanismo para gestionar el software instalado en una distribución Linux se le denomina **sistema de paquetes**.

2.1 Conceptos importantes sobre sistemas de paquetes

Antes de aprender a **manejar los sistemas de paquetes de software**, es muy importante estudiar algunos conceptos relacionados con ellos:

- **Paquete.** Un paquete de software es una **colección de archivos que se instalan juntos en el ordenador**. Incluye programas, archivos auxiliares, archivos de configuración, etc. Por lo general, se distribuyen como un único archivo que es parecido a un archivo tar (*tarball*) o zip.

Cuando se instala un paquete de software, decenas e incluso cientos de archivos se copian al ordenador y después se ejecuta un *script* o programa de configuración para realizar de forma automática la configuración del programa.

El **sistema de paquetes** lleva cuenta de todos los **archivos instalados**, el **paquete al que pertenecen** y el **directorio** en el que se encuentran.

- **Base de datos de archivos instalados.** El sistema de paquetes mantiene una base de datos de todos los archivos instalados. La base de datos incluye información sobre cada fichero instalado a través del sistema de paquetes, el nombre del paquete al que pertenece, e información asociada adicional.
- **Dependencias.** Uno de los tipos de información más importante, mantenido por el sistema de paquetes, es la información sobre dependencias, es decir, las necesidades que tiene un paquete de software de que esté instalado otro paquete de software para funcionar correctamente.

Por ejemplo, si MiPrograma necesita a MiBibliotecaDeArbolesB para realizar su trabajo, esta información está en la base de datos de paquetes. Así, si se intenta instalar MiPrograma sin que esté instalada MiBibliotecaDeArbolesB, el sistema de paquetes lo avisará e impedirá (también puede hacerse que el propio sistema de paquetes resuelva el problema). De forma similar, si se intenta desinstalar MiBibliotecaDeArbolesB estando todavía instalado MiPrograma, el sistema de paquetes lo avisará e impedirá.

- **Checksums.** Los *checksums* permiten verificar la validez del software instalado. Un *checksum* es un número que se obtiene a partir de un archivo. Si debido a un error de disco un archivo ha sido dañado o modificado, su *checksum* no coincidirá con el almacenado en el sistema de paquetes que lo avisará para que pueda ser reinstalado.
- **Actualización y desinstalación de software.** El sistema de paquetes lleva cuenta de los archivos y sus dependencias, por lo que permite una actualización a una nueva versión o una desinstalación de forma sencilla.

Por ejemplo, al desinstalar un paquete, se desinstalan todos los archivos asociados, sin que el administrador tenga que preocuparse de cuáles son o dónde se encuentran. Asimismo, si los archivos de un paquete son necesarios para que otro funcione correctamente, el sistema no permitirá su desinstalación.

Los dos **sistemas de paquetes** más usuales son **Red hat Package Manager (RPM)** y **Debian Package System**. Ambos son sistemas de paquetes, por tanto proporcionan la misma funcionalidad, sin embargo **son incompatibles entre sí**, ya que usan distinto formato para los paquetes (*.rpm* y *.deb* respectivamente) y la base de datos de archivos instalados. **No se puede instalar directamente un paquete rpm en un sistema debian y viceversa.**

Ambos sistemas de paquetes pueden instalarse en un mismo sistema Linux, pero **es desaconsejable**, ya que tienen bases de datos de paquetes instalados separadas. Si se instala un programa usando RPM y a continuación se instala ese mismo programa usando Debian Package System, como las bases de datos son distintas, Debian no se dará cuenta de que ese programa ya se ha instalado y viceversa. Eso puede dejar el sistema en una situación inestable o errónea.

Esta es la razón por la que la mayoría de las distribuciones de Linux optan por instalar uno sólo de los dos sistemas de paquetes. La distribución **Red Hat** y las que se basan en ella como Mandriva y Yellow Dog entre otras, usan el sistema de paquetes **rpm**. La distribución **Debian** y las que se basan en ella como **Ubuntu**, Libranet o Xandros entre otras, usan el sistema de paquetes **debian**.

2.2 Depósitos o almacenes de software

Un **depósito o almacén de software** (*software repository*) es una lista de paquetes instalables. Permite a un sistema de paquetes saber cual es el software disponible para instalar o actualizar. Normalmente se encuentra en un servidor de confianza de la empresa o grupo que realiza la distribución de Linux que se está usando, aunque también puede estar en un CD o DVD de instalación.

Así por ejemplo, en la dirección <http://security.ubuntu.com> puede encontrarse el depósito de software que contiene todo el software disponible para la distribución Ubuntu Linux.

Los depósitos de software también permiten descargar directamente los distintos paquetes que se quieran instalar.

La **gestión de los depósitos de software** la realiza directamente **el sistema de paquetes** y por tanto el administrador no accede a ellos directamente sino a través de las herramientas que proporciona el sistema de paquetes.

3 El sistema de paquetes Debian

3.1 Depósitos de software .deb

En las **distribuciones basadas en Debian como Ubuntu**, el sistema de paquetes que se usa es el **Debian Package System**. Este sistema de paquetes mantiene una lista de todos los depósitos de software disponibles en el archivo `/etc/apt/sources.list`. Los depósitos de software más importantes se añaden automáticamente a este archivo durante la instalación del sistema operativo, sin embargo también puede editarse para añadir manualmente otros depósitos de software para que estén disponibles para el sistema de paquetes de Debian.

En Ubuntu, los paquetes de software se clasifican en **categorías** que se refieren al estado del paquete, por ejemplo, si el paquete se considera seguro o si tiene una licencia de uso que no corresponde con la *common open source standards*.

Las categorías que se usan en Ubuntu son:

- **Main**. Esta categoría contiene todo el software que es soportado oficialmente por *Canonical*, la compañía que está detrás de Ubuntu. El software que se instala normalmente en una distribución Ubuntu pertenece a esta categoría. Si sólo se instala software de esta categoría, nos aseguramos de tener un sistema estable y, lo más importante para las empresas, que va a tener soporte técnico.
- **Restricted**. Esta categoría contiene el software que, si bien es soportado oficialmente, usa una licencia de uso que no es libre. Incluye los controladores (*drivers*) de algunos dispositivos (como las tarjetas gráficas de NVidia), o el software que hay que pagar.

Por lo general, el software de esta categoría se encuentra en un subdirectorio específico del depósito de software.

- **Universe**. Esta categoría contiene software libre que no es soportado oficialmente. Se puede usar, y por lo general no causa problemas, pero no es posible recibir soporte técnico por parte de *Canonical* para esta categoría de software.
- **Multiverse**. Esta categoría contiene software no libre que no es soportado oficialmente.
- **Backports**. En esta categoría se puede encontrar software que es tan novedoso que no tiene soporte oficial y todavía no se encuentra en un estado maduro de desarrollo. Si se quiere trabajar con el último software disponible esta es la categoría adecuada, pero puede dar lugar a un sistema inestable o con problemas.

Cuando se instala software usando las herramientas del sistema de paquetes, éstas buscan la información de los paquetes disponibles y sus dependencias en las fuentes de instalación (los depósitos de software) que se listan en el archivo `/etc/apt/sources.list`.

En el siguiente listado pueden verse tres de las líneas de este archivo en la distribución Ubuntu Linux 12.04 LTS:

```
deb cdrom:[Ubuntu 12.04 LTS _Precise Pangolin_ - Release amd64 (20120425)]/ precise
main restricted
deb http://es.archive.ubuntu.com/ubuntu/ precise main restricted
deb-src http://es.archive.ubuntu.com/ubuntu/ precise main restricted
```

Cada línea tiene cuatro o más campos.

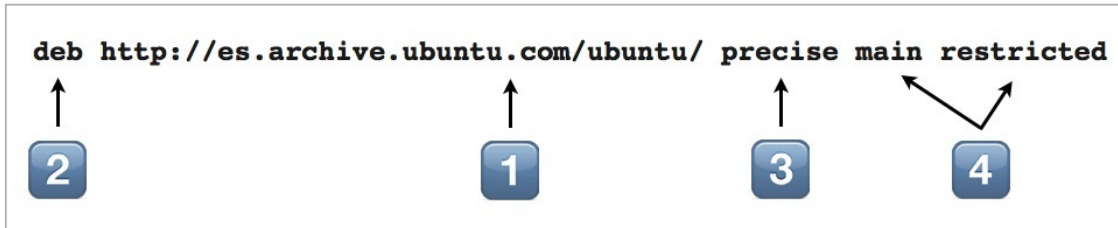
- El primero indica el **formato de los paquetes** que se encuentran en ese depósito de software. Hay dos tipos **deb** y **deb-src**. El primero indica **paquetes binarios** de software mientras que el segundo **paquetes de código fuente**.

Los **paquetes binarios** incluyen los programas ya compilados y enlazados, listos para ejecutarse en la distribución de Linux donde se instalan. Por el contrario, los **paquetes de código fuente** incluyen los archivos de código fuente a partir de los cuales se pueden construir los binarios de los programas compilándolos y enlazándolos. La primera opción es mucho más cómoda, pero los paquetes ocupan más espacio en disco y son más lentos cuando viajan por la red.

- El segundo campo es **la dirección donde se encuentra el depósito de software**. Normalmente es la URL de un servidor en internet (como en las dos últimas líneas del ejemplo), pero también puede hacer referencia al CD de instalación (como en la primera línea) o incluso a un directorio de nuestro equipo en el que se haya descargado una copia del depósito de software.

- El tercer campo hace referencia al **nombre de la distribución** (en el ejemplo la distribución es la 12.04 LTS que tiene como seudónimo: *precise pangolin* o más abreviadamente *precise*).
 - Los campos cuarto y siguientes hacen referencia a la **categoría o categorías de paquetes** que pueden encontrarse en el depósito de software.
- Por ejemplo, en el CD de instalación sólo se encuentran paquetes de las categorías *main* y *restricted*.

La forma correcta de interpretar cada entrada del archivo `/etc/apt/sources.list` puede verse en el siguiente ejemplo:



1. En el depósito de software cuya URL es <http://es.archive.ubuntu.com/ubuntu/>
2. pueden encontrarse paquetes binarios
3. pertenecientes a la distribución *precise pangolin* (12.04 LTS)
4. pertenecientes a las categorías *main* y *restricted*.

Nombres de los Paquetes

Los nombres de los paquetes de Debian se forman de la siguiente manera:
 nombre_version-numero_arquitectura.deb

por ejemplo
 xxchat_2.8.6-4ubuntu5_amd64.deb

El número dice la versión y el número de revisión del paquete software Debian que contiene, mientras que el nombre de la arquitectura especifica la arquitectura de computadores (i386, sparc, todos).

El ejemplo anterior dice que el paquete deb se llama xchat y la revisión 4 de la versión 2.8.6 para el sistema operativo ubuntu5 y la arquitectura AMD64.

3.2 Herramientas del sistema de paquetes Debian

El sistema de paquetes Debian proporciona dos herramientas para la gestión del software instalado en el ordenador: la suite de herramientas **APT** y la orden **dpkg** (Debian PacKaGe tool). Estas son herramientas de la línea de órdenes que permiten una gestión simple del software instalado, aunque también existen herramientas con una interfaz más elaborada tales como *dselect*, *aptitude* y *synaptic*.

De entre todas las herramientas que proporciona la suite **APT**, la más utilizada es **apt-get**^[1]. Esta orden, usa la base de datos de paquetes instalados que se encuentra en `/etc/apt/sources.list`. Debe usarse con permisos de root, por tanto siempre deberá usarse con el siguiente formato:

```
sudo apt-get [opciones][accion] [nombres de paquetes]
```

La orden **dpkg** toma el siguiente formato:

```
dpkg [opciones][acción] [archivo de paquete | nombre de paquete]
```

Un **archivo de paquete** es un archivo en disco con extensión `.deb` o `.deb-src`, que contiene un paquete binario o de código fuente para ser instalado en el sistema. Un nombre de paquete es un nombre simbólico que corresponde total

o parcialmente con el nombre de algún paquete ya instalado en el sistema o bien con el nombre de algún paquete del que el sistema de paquetes tiene conocimiento.

[1] A partir de Ubuntu 14.04 se puede sustituir "**apt-get**" por simplemente "**apt**".

3.3 Comprobando el software instalado

La forma de **comprobar el software instalado** en el ordenador es con la orden `dpkg -l` que genera un **largo listado con todos los paquetes instalados actualmente en el ordenador**.

El siguiente listado es una parte del resultado mostrado por la orden `dpkg -l`:

```
fconde@iMac21:~$ dpkg -l
||/ Nombre                      Versión                      Descripción
+++-----
=====
ii accountsservice              0.6.15-2ubuntu9.4          query and manipulate user account
information
ii acl                          2.2.51-5ubuntu1            Access control list utilities
ii acpi-support                 0.140                      scripts for handling many ACPI
events
```

La primera columna se refiere al **estado** del paquete. Consta de tres caracteres:

1. El primero indica el **estado deseado** del paquete, es decir, el estado en el que debería encontrarse el paquete dada su historia de instalación/desinstalación.
2. El segundo indica el **estado actual** del paquete, es decir, el estado real en el que se encuentra.
3. Y el tercero indica cualquier **estado de error** conocido asociado con el paquete. Por lo general es el espacio en blanco, lo que indica que no hay errores.

El primer carácter indicador de estado "**estado deseado**" puede tomar cualquiera de los siguientes valores:

- **i: (*install*)** Indica que el paquete **debería estar instalado**. Es el indicador que se muestra en la mayoría de los casos.
- **h: (*hold*)** Indica que el paquete **no puede ser modificado**.
- **p: (*purge*)** Indica que el paquete **debería haberse eliminado por completo**, incluyendo sus archivos de configuración.
- **r: (*remove*)** Indica que el paquete **debería haberse eliminado, sin eliminar los archivos de configuración asociados**.
- **u: (*unknown*)** Indica que **no hay un estado deseado** conocido para este paquete.

El segundo carácter indicador de estado "**estado actual**" puede tomar cualquiera de los siguientes valores:

- **i: (*Installed*)** Indica que el paquete está **realmente instalado**.
- **c: (*Config-files*)** Indica que **los archivos de configuración del paquete están instalados**, pero el resto de archivos del paquete no.
- **f: (*Failed-config*)** Indica que el paquete está instalado pero **no hay garantías de que esté correctamente instalado**.
- **h: (*Half-installed*)** Indica que el paquete está **parcialmente instalado**.
- **n: (*Not-installed*)** Indica que el paquete **no está instalado** en este momento.
- **u: (*Unpacked*)** Indica que el paquete se instaló, pero **la instalación no finalizó** ya que el *script* de configuración que se debe ejecutar tras la ejecución del paquete, no se completó con éxito.

El tercer carácter indicador de estado indica cualquier **estado de error** conocido asociado con el paquete. En la mayoría de los casos se muestra un espacio en blanco, lo que indica que no hay ningún error. Otras opciones para este carácter son:

- **H: (*Hold*)** Indica que el **paquete está retenido** por el sistema de paquetes. Esto significa que se han encontrado problemas de dependencias, es decir, que algunos paquetes que este paquete necesita no han sido instalados.
- **R: (*Reinst-required*)** Indica que es necesario **reinstalar el paquete**.
- **X: (*X=both-problems*)** Mezcla de las dos anteriores. El paquete requiere reinstalación y además ha sido

retenido.

La orden `dpkg -l` puede llevar como argumento parte del nombre de un paquete, en cuyo caso muestra sólo los datos de instalación de los paquetes cuyo nombre coincida con el argumento.

Por ejemplo, la orden `dpkg -l zi*` muestra el siguiente listado (en una instalación básica de Ubuntu Linux 12.04 LTS):

```
fconde@iMac21:~$ dpkg -l zi*
||/ Nombre          Versión          Descripción
+++=====
ii  zip                3.0-4           Archiver for .zip files
un  zip-crypt           ninguna         (no hay ninguna descripción disponible)
```

Otra orden interesante para conocer los archivos instalados de un paquete es `dpkg -L nombrePaquete`. Muestra **un listado con todos los archivos instalados por ese paquete**.

Por ejemplo, la orden `dpkg -L zip` muestra el siguiente listado:

```
fconde@iMac21:~$ dpkg -L zip
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/zip
/usr/share/doc/zip/copyright
/usr/share/doc/zip/TODO
/usr/share/doc/zip/changelog.Debian.gz
/usr/share/doc/zip/WHATSNEW
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/zipsplit.1.gz
/usr/share/man/man1/zipcloak.1.gz
/usr/share/man/man1/zip.1.gz
/usr/share/man/man1/zipnote.1.gz
/usr/bin
/usr/bin/zipsplit
/usr/bin/zipcloak
/usr/bin/zip
/usr/bin/zipnote
/usr/share/doc/zip/changelog.gz
```

Como se puede ver, al instalar un paquete de un programa, no sólo se instalan los binarios de ese programa en el ejemplo: `/usr/bin/zip`, sino que también se instalan archivos adicionales como las páginas de manual para el programa: `/usr/share/man/man1/zip.1.gz` o información sobre los cambios en la nueva versión: `/usr/share/doc/zip/WHATSNEW`.

Una última orden muy útil para **mostrar información de un paquete que ya está instalado en el sistema** es `dpkg -p nombrePaquete`.

Por ejemplo: la orden `dpkg -p zip` muestra el siguiente listado:

```
fconde@iMac21:~$ dpkg -p zip
Package: zip
Priority: optional
Section: utils
Installed-Size: 636
Maintainer: Ubuntu Developers
Architecture: amd64
Version: 3.0-4
Replaces: zip-crypt (<= 2.30-2)
Depends: libbz2-1.0, libc6 (>= 2.7)
Recommends: unzip
Conflicts: zip-crypt (<= 2.30-2)
Size: 262186
Description: Archiver for .zip files
This is InfoZIP's zip program. It produces files that are fully
compatible with the popular PKZIP program; however, the command line
```


options are not identical. In other words, the end result is the same, but the methods differ. :-)

This version supports encryption.
Original-Maintainer: Santiago Vila
Homepage: <http://www.info-zip.org/Zip.html>

Como puede verse, esta orden incluye mucha información interesante como **dependencias**, otros **paquetes recomendados** (paquetes que se deberían instalar junto con este), etc.

3.4 Instalando paquetes

Para instalar nuevos paquetes en el sistema se usa la orden `apt-get` con la acción `install` seguida del nombre del paquete que se quiere instalar. Como se trata de una orden que debe ejecutarse con permisos de root, debe precederse de `sudo`. Por tanto el formato de la orden para instalar nuevos paquetes queda así:

```
sudo apt-get install nombrePaquete
```

Una de las ventajas de `apt-get` es que no sólo instala el paquete sino que lo busca en los depósitos de software disponibles y lo descarga al directorio `/var/cache/apt/archives/`. Una vez descargado lo instala y ejecuta el *script* de post-instalación, con lo que el paquete queda perfectamente instalado y configurado.

Además, **si el paquete tiene dependencias, `apt-get install` descarga e instala también aquellos otros paquetes que sean necesarios para que el paquete que queremos instalar funcione correctamente.**

Por ejemplo, para instalar el paquete de software `nmap`, se debe ejecutar la orden: `sudo apt-get install nmap` lo que produce la siguiente salida por pantalla:

```
fconde@iMac21:~$ sudo apt-get install nmap
1 Leyendo lista de paquetes... Hecho
   Creando árbol de dependencias
2 Leyendo la información de estado... Hecho
   El paquete indicado a continuación se instaló de forma automática y ya no es necesarios.
   tdb-tools
   Utilice «apt-get autoremove» para eliminarlos.
   Se instalarán los siguientes paquetes NUEVOS:
   nmap
3 0 actualizados, 1 se instalarán, 0 para eliminar y 6 no actualizados.
   Se necesita descargar 0 B/1.643 kB de archivos.
   Se utilizarán 6.913 kB de espacio de disco adicional después de esta operación.
   Seleccionando paquete nmap previamente no seleccionado
   (Leyendo la base de datos ... 170276 ficheros o directorios instalados actualmente.)
   Desempaquetando nmap (de .../nmap_5.21-1.1ubuntu1_amd64.deb) ...
4 Procesando disparadores para man-db ...
   Configurando nmap (5.21-1.1ubuntu1) ...
```

Como se puede ver, existen cuatro fases en el proceso de instalación:

1. **Leer la información de los depósitos de software para construir la lista de paquetes disponibles, crear el árbol de dependencias y leer la información de estado actual del paquete que se ha pedido instalar.** Con todo eso, la orden sabe si el paquete está disponible para su instalación y si las dependencias se satisfacen. Si todo está correcto la orden puede continuar.
2. Ya que se han leído la lista de paquetes y demás información relacionada, la orden `apt-get` **aprovecha para dar algunos mensajes relativos al sistema de paquetes aunque no estén relacionados con el paquete que se está instalando.** En este ejemplo, la orden aprovecha que estamos instalando el paquete `nmap`, y nos indica que deberíamos eliminar el paquete `tdb-tools`.

Si no hubiese ningún paquete que necesitase nuestra atención, esta fase no produciría ningún resultado en la pantalla.

3. En esta fase se **descarga el paquete al disco**, en este caso el archivo de paquete binario: `nmap_5.21-1.1ubuntu1_amd64.deb`, **se descomprime y se instala.**
4. Por último **se ejecuta el script de post-instalación que configura `nmap`.** Esto hace que el paquete quede perfectamente configurado y listo para usar.

Una opción interesante para la acción `install` de la orden `apt-get` es `-d o --download-only`, que sólo descarga el paquete de software al directorio `/var/cache/apt/archives/`, pero no lo instala ni configura. Esta opción es útil, por ejemplo, cuando se quiere instalar un paquete de software en otro ordenador sin conexión a la red. Se descarga el paquete en otro ordenador y de allí se copia en un *pendrive* para su uso en el ordenador sin conexión.

La orden `sudo apt-get install -d nmap` descarga el archivo de paquete binario: `nmap_5.21-1.1ubuntu1_amd64.deb` en el directorio `/var/cache/apt/archives/`

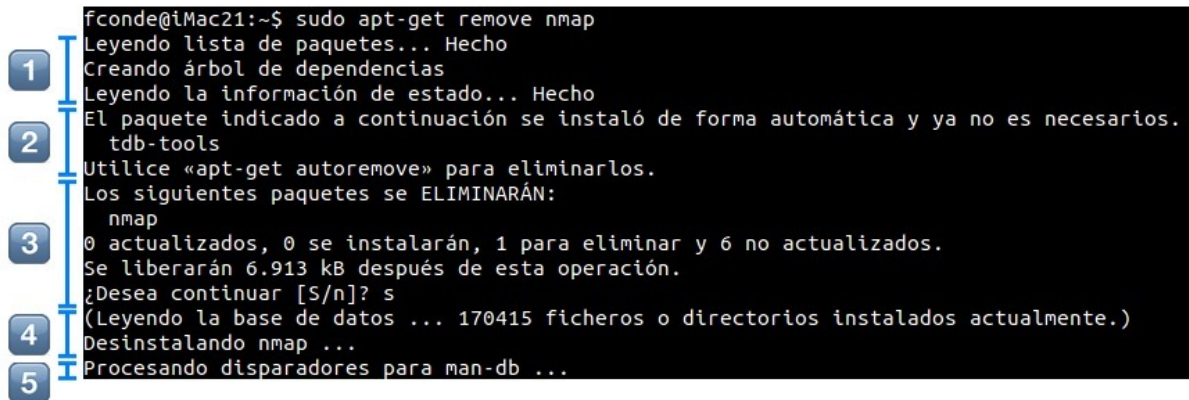
3.5 Eliminando paquetes

Para eliminar paquetes ya instalados en el sistema, se usa la orden `apt-get` con la acción `remove` seguida del nombre del paquete que se quiere eliminar. Como se trata de una orden que debe ejecutarse con permisos de root, debe precederse de `sudo`. Por tanto el formato de la orden para eliminar paquetes ya instalados queda así:

```
sudo apt-get remove nombrePaquete
```

IMPORTANTE: La orden `apt-get remove` **NO** elimina los archivos de configuración asociados al paquete de software que se desinstala. Para realizar una desinstalación completa, que incluya también los archivos de configuración es necesario usar la opción `--purge`.

Por ejemplo, para eliminar el paquete de software `nmap`, se debe ejecutar la orden: `sudo apt-get remove nmap` lo que produce la siguiente salida por pantalla:



```
fconde@iMac21:~$ sudo apt-get remove nmap
1 | Leyendo lista de paquetes... Hecho
   | Creando árbol de dependencias
   | Leyendo la información de estado... Hecho
2 | El paquete indicado a continuación se instaló de forma automática y ya no es necesario.
   | tdb-tools
   | Utilice «apt-get autoremove» para eliminarlos.
   | Los siguientes paquetes se ELIMINARÁN:
   | nmap
3 | 0 actualizados, 0 se instalarán, 1 para eliminar y 6 no actualizados.
   | Se liberarán 6.913 kB después de esta operación.
   | ¿Desea continuar [S/n]? s
4 | (Leyendo la base de datos ... 170415 ficheros o directorios instalados actualmente.)
   | Desinstalando nmap ...
5 | Procesando disparadores para man-db ...
```

Como puede verse, existen cinco fases en el proceso de desinstalación:

1. **Leer la información de los depósitos de software para construir la lista de paquetes disponibles, crear el árbol de dependencias y leer la información de estado actual del paquete que se ha pedido desinstalar.** Con todo eso, la orden sabe si el paquete puede desinstalarse sin que ello afecte a otros paquetes instalados en el sistema. Si todo está correcto la orden puede continuar.
2. Ya que se han leído la lista de paquetes y demás información relacionada, la orden `apt-get` aprovecha para dar algunos **mensajes relativos al sistema de paquetes aunque no estén relacionados con el paquete que se está desinstalando**. En este ejemplo, la orden aprovecha que estamos desinstalando el paquete `nmap`, y nos indica que deberíamos eliminar el paquete `tdb-tools`.

Si no hubiese ningún paquete que necesitase nuestra atención, esta fase no produciría ningún resultado en la pantalla.
3. En esta fase se pide **confirmación al usuario de que realmente quiere desinstalar los paquetes que se listan**. El usuario debe pulsar la tecla `s` para continuar con la desinstalación.
4. **Se eliminan del disco los archivos que componen el paquete de software.** Esto no incluye los archivos de configuración.
5. Por último se **ejecuta el script de post-desinstalación** que completa la eliminación del paquete `nmap`. En este ejemplo, se rehace la base de datos de páginas de manual (`man-db`) para que ya no estén disponible las del paquete `nmap`.

Una opción interesante para la acción `remove` de la orden `apt-get` es `-y o --yes o --assume-yes`, que produce automáticamente la respuesta "sí", a cualquier pregunta que la orden `apt-get` haga al usuario durante su ejecución. En el caso de la acción `remove`, la opción `-y` hace que no se pregunte al usuario si realmente quiere desinstalar el paquete de software.

Si se quisiera una desinstalación completa del paquete `nmap`, la orden correcta es:

```
sudo apt-get remove --purge nmap
```

3.6 Actualizando el sistema

Una de las aplicaciones más usadas de la orden `apt-get` es actualizar el sistema operativo para mantener la última versión de los distintos programas que lo forman.

Para ello se usan las acciones `update` y `dist-upgrade` ejecutadas en secuencia.

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
```

La primera acción, `update`, **actualiza la lista de paquetes disponibles** a partir de la información contenida en los depósitos de software que se guardan en el archivo `/etc/apt/sources.list`. Es muy importante ejecutar esta orden antes de actualizar el sistema operativo, para que el sistema de paquetes cuente con la información más actualizada sobre paquetes y dependencias.

La segunda acción, `dist-upgrade`, **comprueba aquellos paquetes instalados para los que existen versiones más actualizadas** en los depósitos de software, se asegura de que las actualizaciones no generan problemas de dependencias, descarga los archivos de paquete que se necesiten y los ejecuta en el mejor orden para evitar conflictos.

Dependiendo de la velocidad de la conexión a internet y de si hace mucho tiempo desde la última actualización, esta orden puede tardar bastantes minutos, pero cuando termina tenemos el sistema operativo completamente actualizado.

3.7 Ejemplo paso a paso

Para terminar el estudio de los sistemas de paquetes veamos un ejemplo paso a paso de cómo gestionar la **instalación / desinstalación de un paquete de software**.

Para ello vamos a instalar la shell **zsh** que por defecto no se instala en la distribución Ubuntu Linux 12.04 LTS.

1. En primer lugar comprobamos que la shell **zsh** no está ya instalada en el sistema. Para ello pueden usarse o bien la orden:

```
fconde@iMac21:~$ dpkg -l zsh
No se ha encontrado ningún paquete que corresponda con zsh.
```

O bien la orden:

```
fconde@iMac21:~$ dpkg -L zsh
El paquete `zsh' no está instalado.
```

En ambos casos, si el paquete que contiene la shell **zsh** no está instalado, se recibe un mensaje indicándolo.

2. Una vez comprobado que el paquete no está ya instalado, podemos instalarlo. Para ello usaremos la orden:

```
fconde@iMac21:~$ sudo apt-get install zsh
```

3. Ahora podemos comprobar su estado con la orden:

```
fconde@iMac21:~$ dpkg -l zsh
||/ Nombre          Versión          Descripción
+++-----
ii  zsh                4.3.17-1ubuntu  shell with lots of features
```

El estado es `ii` lo que indica que no hay errores, que el paquete debería estar instalado y que realmente lo está.

4. Ahora podemos comprobar los archivos que se han instalado en el disco al instalar el paquete `zsh`. Para ello usamos la orden:

```
dpkg -L zsh
```

que produce un listado muy largo de archivos, entre ellos `/bin/zsh4` que es el binario mediante el cual se puede ejecutar la shell recién instalada.

Ejecutando la orden `ls -al /bin/zsh*` obtenemos como resultado:

```
lrwxrwxrwx 1 root root      21 nov  2 11:19 /bin/zsh -> /etc/alternatives/zsh
-rwxr-xr-x 1 root root 688656 mar 30 2012 /bin/zsh4
```

Y ejecutando la orden `ls -al /etc/alternatives/zsh` obtenemos como resultado:

```
lrwxrwxrwx 1 root root 9 nov  2 11:19 /etc/alternatives/zsh -> /bin/zsh4
```

Es decir, que también se puede ejecutar la shell mediante la orden `zsh`, ya que existen enlaces simbólicos al binario `/bin/zsh4`. Esos enlaces simbólicos los ha creado el *script* de post-instalación del paquete **zsh**, como parte del proceso de configuración.

5. Vamos a probar ahora la nueva shell instalada. Para ello usamos la orden `zsh4` o `zsh`. Ambas producen el mismo resultado ya que `zsh` es un enlace simbólico al programa `zsh4`.

El programa muestra un menú para realizar la configuración de usuario inicial. Debemos pulsar 0 (cero) y a continuación muestra el prompt de la nueva shell:

```
iMac21%
```

Podemos teclear algunas órdenes para probar el funcionamiento de la nueva shell instalada y para salir la orden interna `exit`.

6. Ahora vamos a desinstalar la shell **zsh**. Para ello ejecutamos la orden:

```
sudo apt-get remove zsh
```

o la orden:

```
sudo apt-get -y remove zsh
```

si no queremos que el sistema de paquetes nos pregunte si realmente queremos desinstalar el paquete.

7. Si intentamos ejecutar la shell **zsh** el sistema nos responde con el siguiente mensaje:

```
fconde@iMac21:~$ zsh
bash: /usr/bin/zsh: No existe el archivo o el directorio
```

8. Y por último, si comprobamos el estado mediante la orden:

```
fconde@iMac21:~$ dpkg -l zsh
||/ Nombre          Versión          Descripción
+++-----
rc  zsh              4.3.17-1ubuntu shell with lots of features
```

Obtenemos el estado `rc`. El primer carácter (`r`) indica que el paquete debería haberse eliminado sin eliminar sus archivos de configuración. El segundo carácter (`c`) indica que los archivos de configuración del paquete existen, pero el paquete en sí no está instalado.

Si volvemos a instalar el paquete **zsh** y ejecutamos la orden `zsh` o `zsh4`, esta vez, no nos va a mostrar el menú