

GESTIÓN DE SOFTWARE Y PAQUETES

Instalación de SW en GNU/linux

La instalación de programas en GNU/Linux es diferente a la de otros sistemas operativos. Siguen un estándar para el desarrollo de su software y asimismo para su instalador. Este estándar implica que cada programa a ser instalado se representa como un paquete de software.

1.

Paquete de Software

Conjunto de
archivos



Es un conjunto de archivos necesarios para instalar, desinstalar, configurar y ejecutar un programa, utilizando herramientas de gestión de software del sistema operativo.

Un programa consta de uno o más paquetes.

Tipos de paquetes

Paquetes Binarios

Contienen los archivos para reconstruir una aplicación dentro del sistema. Es software precompilado. Se reconocen por ser `.deb` (dpkg) y `.rpm` (yum)



Paquetes de Código

contienen los códigos fuente y los archivos necesarios para compilar e instalar en forma manual los programas. Son comprimidos y se reconocen por ser `.tgz`, `.tar.gz` o `.tar.bz2`



Relación entre paquetes

- Un programa puede estar contenido en un solo paquete o consistir en varios paquetes relacionados entre ellos.
- Varios programas pequeños y relacionados entre sí pueden estar en el mismo paquete (ej: paquete fileutils que contiene varios comandos como ls, cp, etc.)
- Algunos paquetes requieren de otros para funcionar. En Debian, algunos paquetes pueden **dependen** de otro, recomendar, sugerir, romper, o entrar en conflicto con otros paquetes.

Dependencia de paquetes

- Si un paquete **A depende** de otro paquete **B**, entonces B es necesario para que A funcione correctamente. Ej: gimp depende de gimp-data para que el editor gráfico GIMP acceda a sus ficheros críticos de datos.
- Si un paquete **A recomienda** otro paquete **B**, entonces B ofrece una funcionalidad adicional para A. Ej: mozilla-browser recomienda el mozilla-psm, que añade la capacidad para la transferencia segura de datos al navegador web de Mozilla.
- Si un paquete **A sugiere** otro paquete **B**, entonces el paquete B ofrece a A una funcionalidad que puede que mejore A, pero que no es necesaria en la mayoría de los casos. Ej: el kmail sugiere el gnupg, que contiene software de cifrado que KMail puede emplear.
- Si un paquete **A entra en conflicto** con otro paquete **B**, los dos paquetes no se pueden instalar a la vez. Ej: fb-music-hi entra en conflicto con fb-music-low porque ofrecen conjuntos alternativos de sonidos para el juego Frozen Bubble.

Instalación de un paquete

Para la instalación de un programa se utiliza un **Gestor de Paquetes** que asiste al usuario en la tarea de administrar el conjunto de paquetes.

Instalación de un paquete

Para la instalación de un programa se utiliza un **Gestor de Paquetes** que asiste al usuario en la tarea de administrar el conjunto de paquetes.

2.

Gestor de Paquetes de Software

Programa para
administrar
paquetes

Un Gestor de Paquetes

- Es una colección de herramientas (comandos) para automatizar el proceso de instalación, actualización, configuración y eliminación de paquetes de software. Debian y derivados utilizan las herramientas de alto nivel **apt** y **aptitude**.
- Mantiene un registro del SW está instalado lo que permite una gestión de sencilla y centralizada de los programas.
- Forma parte del sistema operativo y posee un único formato de paquetes (estándar).
- Utiliza la información incluida en los paquetes (nombre completo, descripción de funcionalidad, versión, y paquetes requeridos para su funcionamiento).

Instalación de un programa con el Gestor de Paquetes apt

1º Paso: el usuario solicita instalación de un programa por medio de un comando del Gestor (por ejemplo: `#apt-get install tree` es el pedido para instalar el comando tree)



2º Paso: el Gestor busca el paquete de software correspondiente al programa solicitado en el **repositorio** (url, discos) definido en el archivo de texto: `/etc/apt/sources.list`



3º Paso: el Gestor ubica los paquetes necesarios, los descarga, los desempaqueta y copia sus archivos binarios, archivos de configuración y manuales a sus respectivos directorios dejando a los mismos listos para su ejecución. Realiza sugerencias sobre la instalación y el programa queda listo para ser utilizado.

Repositorios

- Son directorios en servidores especiales que únicamente mantienen paquetes e información de estos, en una estructura similar a una base de datos.
- El software está a disposición en estos repositorios, a fin de proporcionar un sencillo control sobre los diferentes tipos de software que van a instalar en su sistema.
- Cada distribución mantiene organizados los paquetes en repositorios, lo que permite actualizar e instalar por red todo el sistema desde una localización confiable (repositorios oficiales).
- Se utilizan para actualizar los programas instalados o instalar programas que no se encuentren en nuestra distribución.
- Pueden ser de acceso público, o pueden estar protegidos y necesitar de una autenticación previa. Los más conocidos son los académicos e institucionales.

Repositorios



- Son directorios en servidores especiales que únicamente mantienen paquetes e información de estos, en una estructura similar a una base de datos.
- El software está a disposición en estos repositorios, a fin de proporcionar un sencillo control sobre los diferentes tipos de software que van a instalar en su sistema.
- Cada distribución mantiene organizados los paquetes en repositorios, lo que permite actualizar e instalar por red todo el sistema desde una localización confiable (repositorios oficiales).
- Se utilizan para actualizar los programas instalados o instalar programas que no se encuentren en nuestra distribución.
- Pueden ser de acceso público, o pueden estar protegidos y necesitar de una autenticación previa. Los más conocidos son los académicos e institucionales.

/etc/apt/sources.list

- Es un archivo de texto que contiene una lista de direcciones (fuentes) con las ubicaciones de los repositorios de paquetes de nuestra distribución.
- Antes de instalar programas debemos chequear nuestro archivo sources.list y actualizarlo de ser necesario.
- Se puede editar y agregar otras fuentes diferentes a las configuradas durante la instalación con cualquier editor de texto. Por ejemplo:

```
root@mipc:~#nano /etc/apt/sources.list
```

- Luego de editar sources.list, siempre actualizar con la línea de comandos:

```
root@mipc:~#apt-get update ó root@mipc:~#aptitude update
```

#

```
#deb cdrom:[Debian GNU/Linux 9.0.0 _Stretch_ - Official amd64 DVD Binary-1 20170617-13:08]/ stretch main
#deb cdrom:[Debian GNU/Linux 9.0.0 _Stretch_ - Official amd64 DVD Binary-1 20170617-13:08]/ stretch main
```

```
deb http://deb.debian.org/debian stretch main contrib non-free
deb-src http://deb.debian.org/debian stretch main contrib non-free
```

```
deb http://deb.debian.org/debian-security/ stretch/updates main contrib non-free
deb-src http://deb.debian.org/debian-security/ stretch/updates main contrib non-free
```

```
deb http://deb.debian.org/debian stretch-updates main contrib non-free
deb-src http://deb.debian.org/debian stretch-updates main contrib non-free
```

```
# stretch-oficiales
```

```
deb http://ftp.us.debian.org/debian/ stretch main contrib non-free
deb-src http://ftp.us.debian.org/debian/ stretch main contrib non-free
```

```
# debian-multimedia
```

```
deb http://www.debian-multimedia.org stretch main non-free
```

```
#deb http://download.virtualbox.org/virtualbox/debian stretch contrib
```

- ▣ Ubicaciones de repositorios
- ▣ Secciones
- ▣ # comentarios

Secciones en `/etc/apt/sources.list`

```
deb http://deb.debian.org/debian stretch main
```

- **tipo:** 2 tipos: **deb** (típico paquete binario de Debian) y **deb-src** (código fuente oficial del paquete)
- **uri:** Identificador Universal de Recursos, tipo de recurso de la cual se obtienen los paquetes.
 - *CD-ROM*: APT usa la unidad de CD-ROM local. Con apt-cdrom se añade entradas en consola.
 - *FTP*: Especifica un servidor FTP como archivo.
 - *HTTP*: Especifica un servidor HTTP como archivo.
 - *FILE*: Permite considerar como archivo a cualquier fichero en el sistema de ficheros.
- **distribución:** Distribución instalada. Cada distribución cuenta con sus paquetes.
- **componentes:** Los componentes son los tipos de repositorios clasificados según las licencias de los paquetes que contienen. Dentro de los componentes tenemos **main**, **contrib** y **non-free**.

Componentes o tipo de repositorio

■ **main**

Directorio con paquetes 100% libres; esto quiere decir que cumplen o están de acuerdo con las directivas de Debian, en donde marcan cuando un paquete se le puede considerar que es 100% software libre.

■ **non-free**

Se encuentran paquetes que no pueden considerarse software libre según las directivas de Debian. Ej: hay software que puede ser distribuido e instalado, pero no se tiene acceso a su código fuente (No todos de esta sección son así hay software que si se proporciona su código fuente), simplemente por la licencia que trae el software de este paquete no cuadra con las directivas de Debian.

■ **contrib**

Se puede encontrar software libre, pero depende de alguna forma de un paquete que no es 100% libre

Gestor de Paquetes de Debian y derivados

- ▣ Debian cuenta con la herramienta de alto nivel **apt** como gestor de paquetes.
- ▣ **Advanced Packaging Tool** (Herramienta Avanzada de Empaquetado) es un Gestor de Paquetes creado por el proyecto Debian.
- ▣ Contiene un grupo de comandos que permite:
 - ▣ instalar o eliminar un paquete,
 - ▣ actualizar el sistema,
 - ▣ listar paquetes disponibles, etc.
- ▣ Los comandos de apt descargan los paquetes desde repositorios **resolviendo e instalando automáticamente** todas las dependencias de cada paquete a ser instalado, recomendando la instalación de otros posiblemente relacionados.

Algunas operaciones con apt (1)

- ▣ `#apt-get install paquete`

Descarga el paquete, junto con todas sus dependencias y los instala o actualiza; los paquetes bajados se descargan en `/var/cache/apt/archives`.

- ▣ `#apt-get -f install`

Repara e instala dependencias pendientes que necesita el sistema.

- ▣ `#apt-get remove paquete`

Desinstala el paquete del sistema.

- ▣ `#apt-get update`

Actualiza la lista de paquetes disponibles y sus versiones a partir de los repositorios disponibles. Debería usarse si hay que instalar algún paquete nuevo o cada vez que se modifique `/etc/apt/sources.list`.

Algunas operaciones con apt (2)

- ▣ `#apt-get clean`

Borra todos los paquetes en `/var/cache/apt/archives`.

- ▣ `#apt-cache search expresion_regular`

Busca un patrón en los nombres de paquetes y sus descripciones. Donde `expresion_regular` es una cadena de caracteres que representa al paquete buscado.

- ▣ `#apt-cache show paquete`

Muestra la descripción completa del paquete.

Cuando no podemos usar apt

- Las herramientas apt y aptitude sólo pueden instalar paquetes .deb desde un repositorio.
- No todos los programas se pueden instalar con apt. Generalmente se trata de paquetes de programas (por ejemplo el navegador Chrome, Skype, Dropbox y otros) que no se encuentran en los repositorios pero sí están disponibles para descargar de la web, en un DVD-ROM o una unidad de almacenamiento. Para manipular estos paquetes en las distribuciones Debian y derivados se recurre al comando **dpkg**.

Comando dpkg

- Es un programa de bajo nivel que gestiona paquetes .deb, permite la instalación, desinstalación y consulta de información de los paquetes instalados.
- A diferencia de apt no instala automáticamente las dependencias. Se limita a indicarlo durante el proceso de instalación.
- Una vez instalado un paquete .deb con dpkg, se puede ejecutar la línea de comandos para completar las dependencias faltantes o instalarlas de a una con dpkg si se dispone de los paquetes de las dependencias.

Operaciones con `dpkg`

- ▣ `#dpkg -l` (letra ele minúscula)

Comprueba los paquetes instalados en la máquina y ofrece un listado completo. Si queremos información relacionada con un solo paquete, se puede utilizar `grep`.

Ejemplo: `dpkg -l |grep tree`

- ▣ `#dpkg -L nombrePaquete`

Informa sobre el contenido (los ficheros) que forman un paquete.

- ▣ `#dpkg -i nombrePaquete`

Para instalar paquetes que tenemos localmente y no necesitamos descargar

- ▣ `#dpkg -r nombrePaquete`

Desinstala el paquete

Ejemplo de instalación con `dpkg`

- Abrimos una terminal y nos logueamos como superusuario o usuario con privilegios de root (sudo). Nos ubicamos en el directorio en el cual descargamos el paquete .deb

```
#cd /home/usuario/Descargas
```

- Ejecutamos la instalación:

```
# dpkg -i paquete.deb
```

- El paquete se instalará y puede suceder que muestre un error si hay dependencias no satisfechas, en este caso, se resuelven con:

```
# apt-get -f install
```

- Este comando descarga las dependencias requeridas, las instalará y concluirá la instalación del paquete .deb que quedó interrumpida.

3.

Instalación desde Código fuente

Compilar y crear un
ejecutable
compatible con
nuestra
distribución

Instalación desde el código fuente

- Si no hay paquetes hechos para nuestra distribución o no se encuentren actualizados en los repositorios podemos descargar la versión de ese programa en CÓDIGO FUENTE (Source Code) desde la página oficial del programa o de desarrollo del mismo .
- Es el programa escrito por el programador. Entonces para poder usarlo programa debemos compilarlo y crear un ejecutable compatible con nuestra distribución. Generalmente estos códigos vienen en lenguaje C y C++, aunque pueden aparecer en Python, Ruby, etc.
- Suelen presentarse en formato .tar.gz o tar.bz2 (o sea compactado con tar y comprimido con gzip o bzip).
- Generalmente tienen información en el archivo README o INSTALL

Ejemplo de Instalación desde el código fuente



- **1º PASO:** Descargar el código fuente `miprograma.tar.gz` en un directorio (`/opt`) y situarse en el directorio.

```
root@mipc:~#cd /opt
```

- **2º PASO:** Descomprimir el archivo:

```
root@mipc:/opt#tar -xzvf miprograma.tar.gz
```

- **3º PASO:** Al descomprimir, se creará un directorio con el nombre del archivo comprimido. Situar en el directorio producto de la descompresión.

```
root@mipc:/opt#cd miprograma
```

- **4º PASO:** Ejecutar el script `configure` el cual prepara al sistema para compilar el programa.

```
root@mipc:/opt/miprograma#./configure
```

- **5º PASO:** Compilar el código fuente con el comando `make` para generar el código máquina o binarios ejecutables del programa.

```
root@mipc:/opt/miprograma#make
```

- **6º PASO:** Copiar los binarios generados y archivos necesarios a los respectivos directorios con `make install`.

```
root@mipc:/opt/miprograma#make install
```

Gracias!

Preguntas?