

A tope de  rpm



1. Introducción a la gestión de paquetes

Instalación tradicional de software



Pulse *Siguiente* para continuar con esta charla

- Al principio, para instalar un programa, simplemente **se copiaban** los archivos al disco duro (*MS-DOS, Unix,...*)
- Más tarde aparecieron los **instaladores**, que realizaban esta tarea de una forma más sencilla
- Los instaladores aún se siguen utilizando en **Windows**

Instalación tradicional de software

Pocos programas: ok

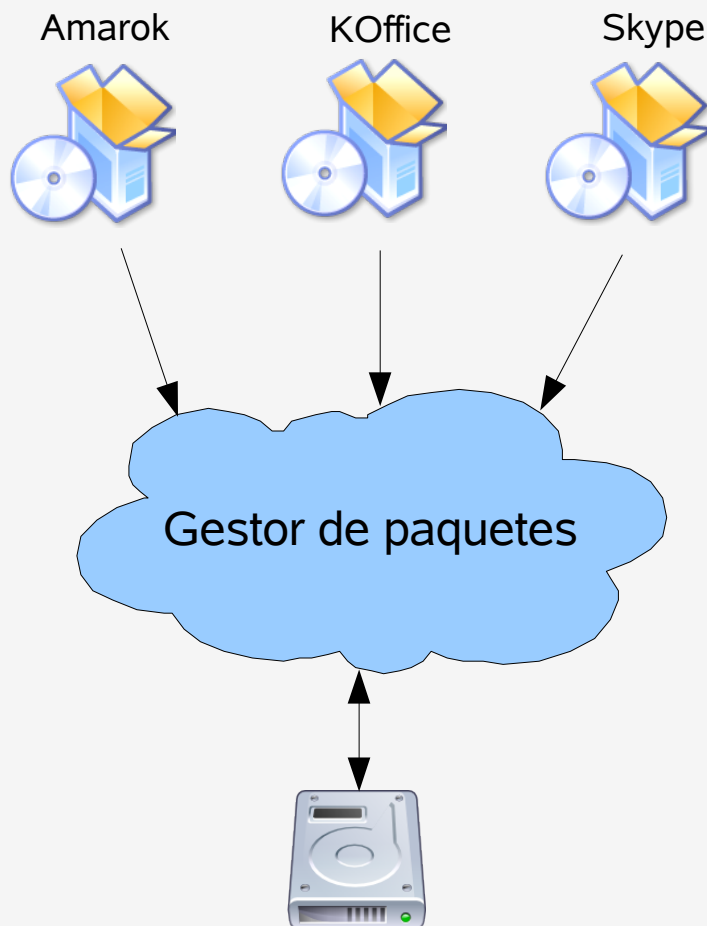
Muchos programas: **Problemas**

- Problemas para saber **dónde se ha instalado** cada programa y para desinstalarlo completamente
- Todos los componentes del programa tienen que ir incluidos juntos, lo cual **desperdicia espacio** al no reutilizarlos
- No existe un **sistema estándar** para instalar los programas, que sea exactamente igual para todos
- No hay una forma estándar de dar **información** sobre el programa



Sistemas de gestión de paquetes

Para solucionar estos problemas, surgieron los sistemas de gestión de paquetes.



- Organizan el software de forma **centralizada** para todo el sistema
- Ofrecen un **mayor control** sobre el software instalado, sabiendo en todo momento dónde están los ficheros de cada programa. Esto permite desinstalar completamente un programa sin que afecte a los demás.
- Permiten **ahorrar espacio** al reutilizar librerías y datos gracias al concepto de dependencias.

Sistemas de gestión de paquetes

Hay **muchos** sistemas de gestión de paquetes para Linux:

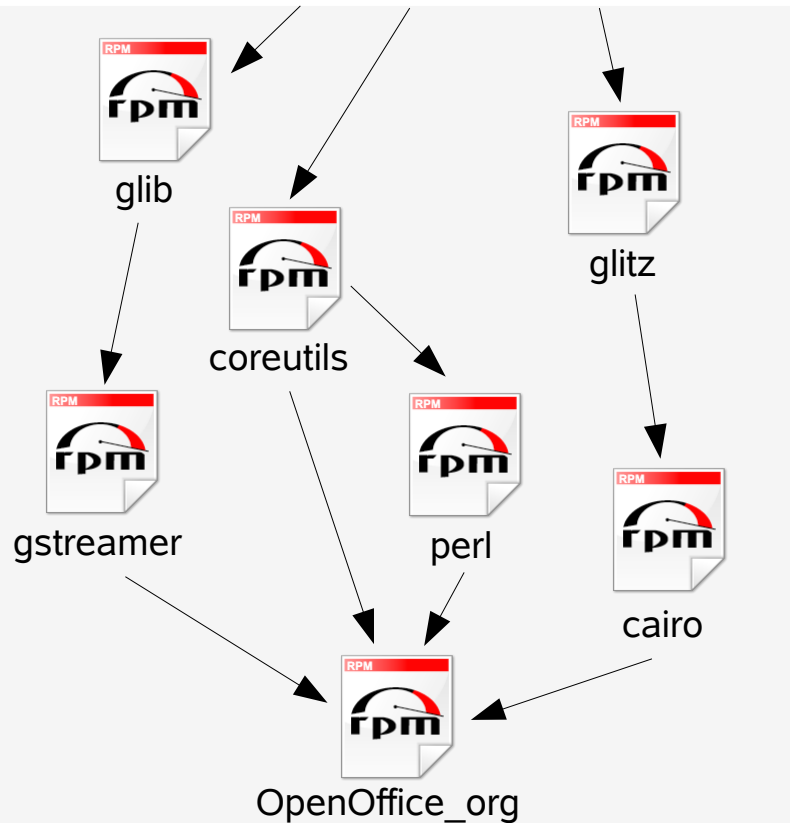
- RPM (*openSUSE, Mandriva,...*)
- Dpkg (*Debian, Kubuntu,...*)
- Portage (*Gentoo*)
- Tgz (*Slackware*)
- Pacman (*Arch Linux*)
- Autopackage
- Klik
- ...

Por desgracia, **no hay un sistema único**, lo cual es una fuente de dolores de cabeza para los desarrolladores



Defectos de los sistemas de gestión de paquetes

A pesar de sus ventajas, aún siguen sin resolver algunos problemas:



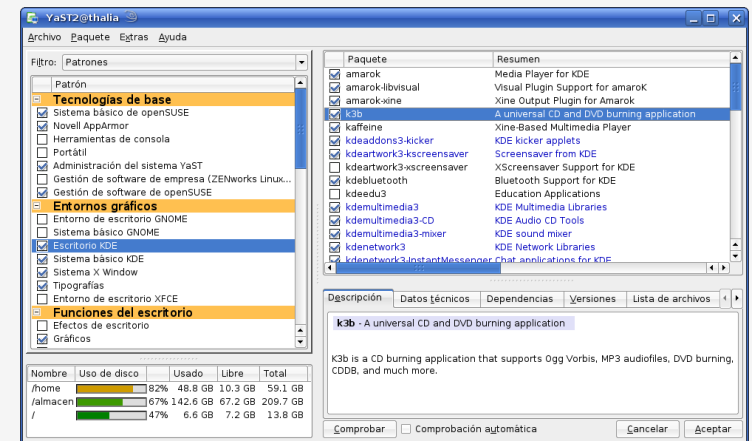
¡Oiga, que yo sólo quiero el OpenOffice!

- Resolución automática de **dependencias**
- **Obtención** centralizada de los paquetes (hay que buscar cada uno por separado)
- **Actualización** automática de los paquetes instalados

Herramientas de gestión de repositorios

Por ello, no tardaron en aparecer herramientas encima de los sistemas de gestión de paquetes que **gestionan repositorios**.

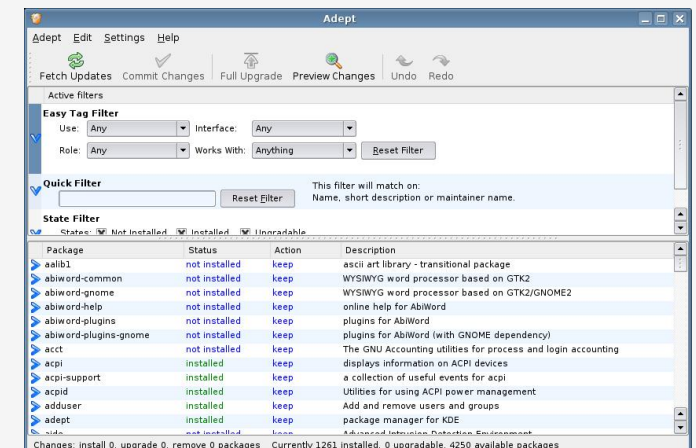
- APT (*Debian y derivados*)
- YaST/libzypp (*openSUSE*)
- YUM
- urpmi (*Mandriva, PCLinuxOS*)
- swaret (*Slackware*)
- Smart
- ...



YaST en openSUSE 10.2

También aparecieron **interfaces gráficas** para estas herramientas


- Adept (*para APT*)
- YaST
- Smart GUI
- Synaptic (*para APT*)
- ...



Adept

2. Cómo es RPM

Sistema de gestión de paquetes RPM



What is RPM?

The RPM Package Manager (RPM) is a powerful command line driven package management system capable of installing, uninstalling, verifying, querying, and updating computer software packages. Each software package consists of an archive of files along with information about the package like its version, a description, and the like. There is also a library API, permitting advanced developers to manage such transactions from programming languages such as C or Python.

RPM is **free software**, released under the **GNU GPL**.

RPM is a core component of many Linux distributions, such as **Red Hat Enterprise Linux**, the **Fedora Project**, **SUSE Linux Enterprise**, **openSUSE**, **CentOS**, **Mandriva Linux**, and many others. It is also used on many other operating systems as well, and the RPM format is part of the **Linux Standard Base**.

Current status

We're relaunching rpm.org, with a new direction for

Download

Get the RPM source [here](#).

Contribute

There are several ways to contribute to RPM.

- Join the **development community**.
- Write **documentation**.
- File **bugs**.

Contact

- **Mailing lists**
- **IRC channel**

<http://www.rpm.org>

- RPM es uno de los sistemas de gestión de paquetes **más populares** en Linux.
- Forma parte de la Linux Standard Base (**LSB**).
- La herramienta básica que se encarga de la gestión de los paquetes es el **comando “rpm”**.
- Los paquetes en formato RPM tienen la **extensión “.rpm”**.

Tipos de paquetes RPM

Hay 3 tipos de paquetes RPM:

- Paquetes **binarios**

Contienen software **ya compilado** y listo para instalar. Su extensión **depende de la plataforma**, por ejemplo “.i586.rpm”, “.x86_64.rpm”,...

- **Delta RPMs**

Contienen las **diferencias entre dos versiones** de un paquete. Son útiles para actualizar un paquete con una descarga de menor tamaño.

- Paquetes de **código fuente**

Contienen el código fuente de un programa, parches, así como los ficheros necesarios **para construir un paquete binario** (specfiles)

Tienen la extensión “.src.rpm”

Nomenclatura de los paquetes RPM

El **nombre** de un paquete RPM se forma así:

nombre del programa – versión – revisión . arquitectura . rpm

Ejemplos:

ulogd-1.23-48.i586.rpm

alsa-firmware-1.0.13-25.noarch.rpm

autofs-4.1.4-53_62.3.x86_64.delta.rpm

FlightGear-0.9.10-45.src.rpm

Instalación de paquetes RPM

Instalar un paquete:

```
rpm -i aircrack-ng-0.6.1-19.i586.rpm
```

Desinstalar un paquete:

```
rpm -e aircrack-ng
```

Actualizar un paquete:

```
rpm -U aircrack-ng-0.7-5.i586.rpm
```

Consultas a RPM

Podemos **obtener mucha información** sobre un paquete RPM con la opción “-q” y a continuación el parámetro que queramos.



rpm -q

- Si el paquete ya **está instalado**:

```
rpm -q[parámetros] paquete
```

- Si tenemos el paquete en el **directorio actual**:

```
rpm -qp[parámetros] paquete.rpm
```

Algunas consultas a RPM útiles

-qa	Lista de todos los paquetes instalados
-qi <i>paquete</i>	Detalles de un paquete (nombre, fecha de creación, página web, descripción,...)
-q --requires <i>paquete</i>	Dependencias de un paquete
-ql <i>paquete</i>	Lista de ficheros que contiene un paquete
-qf <i>fichero</i>	Nombre del paquete que contiene un fichero
-q --whatrequires <i>paquete</i>	Lista de paquetes que tienen como dependencia a un paquete

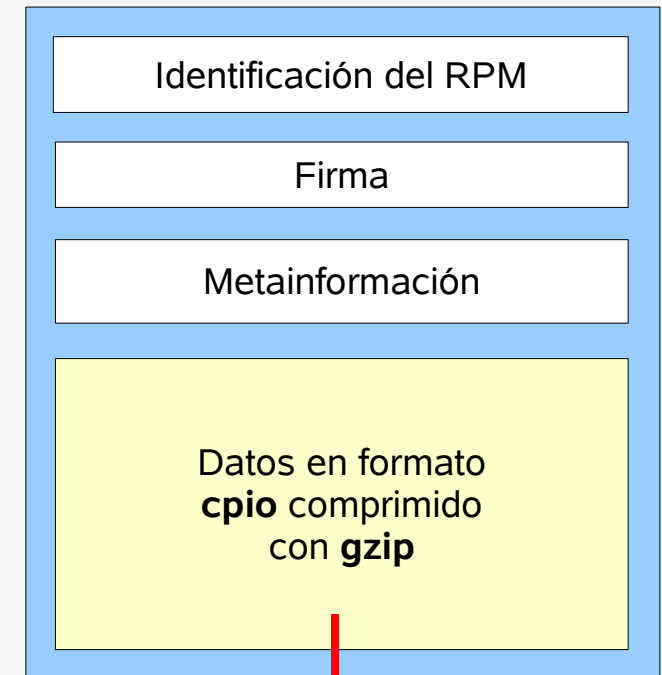
Extraer el contenido de un paquete RPM

Los ficheros de programas y datos que contiene un paquete RPM están en formato **cpio comprimido**.

El formato cpio es **similar** en funcionalidades a un **tar**.

Podemos extraerlos en el directorio actual con el comando “**rpm2cpio**”:

```
rpm2cpio paquete | cpio -ivd
```



Paquete RPM

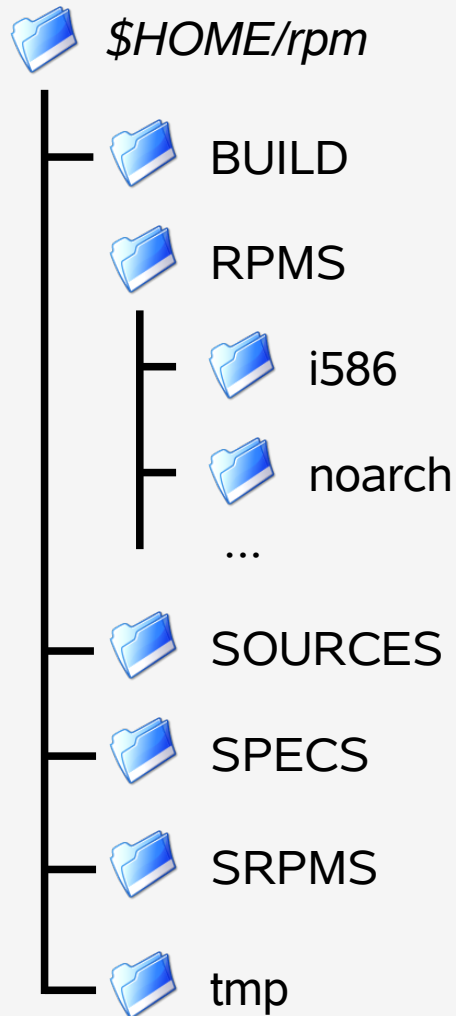
rpm2cpio



.cpio

3. Creando nuestros propios RPM

Entorno de desarrollo de paquetes RPM



Entorno de desarrollo de paquetes RPM

- Antes de empezar a crear nuestros propios RPM tenemos que crearnos un **entorno de desarrollo** de paquetes RPM.
- Este entorno de desarrollo no es más que una **estructura de directorios**, con un directorio padre y los subdirectorios “BUILD”, “RPMS”, “SOURCES”, “SPECS”, “SRPMS” y “tmp”.
- En el directorio RPMS, creamos un subdirectorio **por cada plataforma**.

~/rpmmacros

Una vez creado el entorno tenemos que crear el fichero **.rpmmacros** dentro de nuestro \$HOME para que RPM lo utilice.

```
%_topdir      /home/usuario/rpm
%_tmppath     /home/usuario/rpm/tmp
```

Podemos añadir **macros adicionales**, como por ejemplo la distribución o el nombre del empaquetador:

```
%distribution  openSUSE 10.2
%packager      Víctor Fernández <vfernandez@polinux.upv.es>
```

rpmbuild

- Para construir paquetes RPM utilizamos el comando “**rpmbuild**”.
- Hay dos formas de construir un **paquete RPM binario** (.i586.rpm, .x86_64.rpm,...):
 - A partir de un paquete RPM de código fuente (.src.rpm)
 - Desde cero (a partir del código fuente del programa)
- En el primer caso, basta con:

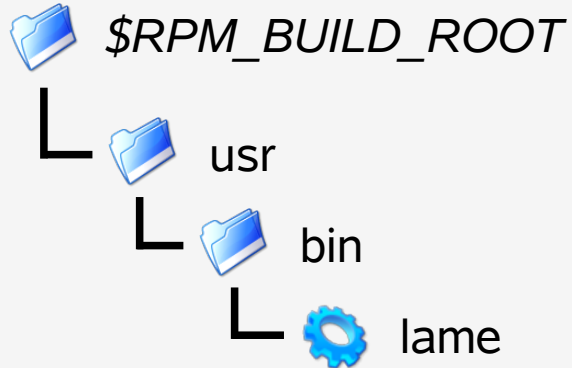
```
rpmbuild --rebuild paquete.src.rpm
```

El proceso de construcción de paquetes RPM

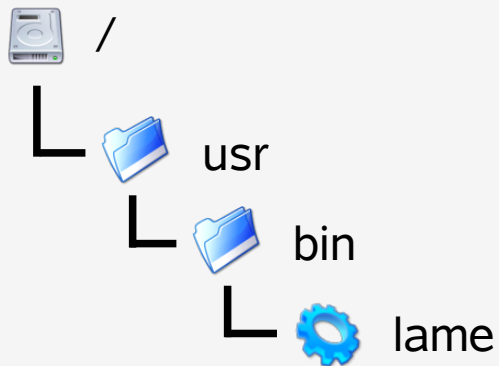
El proceso de construcción de un paquete RPM desde 0 sigue los siguientes pasos:

1. Obtener el **código fuente** del programa y colocarlo en ***SOURCES***
2. **Escribir un specfile** en el directorio ***SPECS***
3. Estando en el directorio ***SPECS***, **ejecutar `rpmbuild -bb`** con el nombre del specfile
4. Si todo ha ido bien, el paquete binario estará en ***RPMS/arquitectura*** y el paquete de código fuente en ***RPMS***
5. Si hay algún problema, modificamos el specfile y volvemos al paso 3

Directorio raíz falso



Nada más compilar el paquete

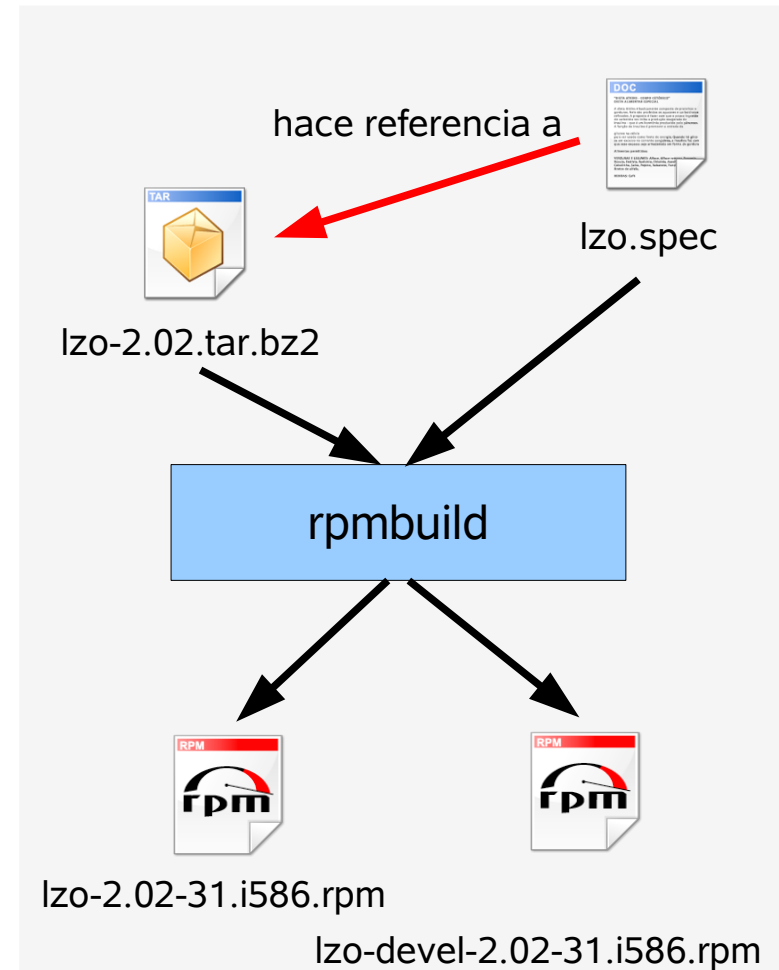


Tras instalar el paquete

- Durante la compilación de un paquete RPM, los archivos del programa se instalan en un **directorio raíz falso** (fake root).
- Los directorios y archivos que haya en este directorio **serán instalados en el directorio raíz** cuando instalemos el paquete.
- Tendremos la ruta a este directorio en la variable de entorno **`$RPM_BUILD_ROOT`**.

El specfile

- Un specfile es un fichero que contiene información sobre un programa, así como las **instrucciones para compilarlo**.
- En él también se indica el/los **ficheros que contienen el código fuente** del programa y los parches a aplicar.
- Los specfiles tienen la extensión **.spec** y son ficheros de texto plano que se pueden editar con un editor de texto normal.
- Un specfile puede generar uno o **varios paquetes**.



Partes de un specfile

Un specfile típico tiene las siguientes partes:

```

BuildRequires: kdelibs3-devel

Name: knutclient
Summary: Graphical client for UPSes supported by nut
Summary(es): Cliente gráfico para SAIs soportados por nut
Version: 0.9
Release: lpolinux
URL: http://www.alo.cz/knutclient/
Source: %{name}-%{version}.tar.gz
License: GPL
Group: Hardware/UPS
Buildroot: %{_tmppath}/%{name}-%{version}-root
Requires: kdelibs3 >= 3.5
Requires: nut
Packager: Victor Fernández <vfernandez@polinux.upv.es>

%description
KNutClient is a visual KDE client for UPS systems using NUT.

Author
-----
Daniel Prynych <Daniel.Prynych@alo.cz>

%prep
%setup -q
. /etc/opt/kde3/common_options

%build
. /etc/opt/kde3/common_options
./configure --prefix=/opt/kde3
make

%install
. /etc/opt/kde3/common_options
make DESTDIR=$RPM_BUILD_ROOT $INSTALL_TARGET
kde_post_install

%clean
rm -rf $RPM_BUILD_ROOT

%files
%defattr(-,root,root)
/opt/kde3/*
%doc ChangeLog COPYING NEWS README TODO AUTHORS

%changelog
* Thu Oct 26 2006 - vfernandez@polinux.upv.es
- first build

```

Cabecera con información sobre el programa

Descripción sobre para qué sirve el programa

Instrucciones para compilar el programa

Ficheros a incluir en el paquete

Lista de cambios (change log)

Firmar los paquetes con GPG

- Por seguridad, es conveniente **firmar** los paquetes **con GPG**.
- Para ello, tras generar nuestro par de claves, indicamos a rpmbuild dónde están con las siguientes macros en `~/.rpmmacros`:

```
%_signature      gpg
%_gpg_name        Nombre <email@servidor>
%_gpg_path        ~/.gnupg
```

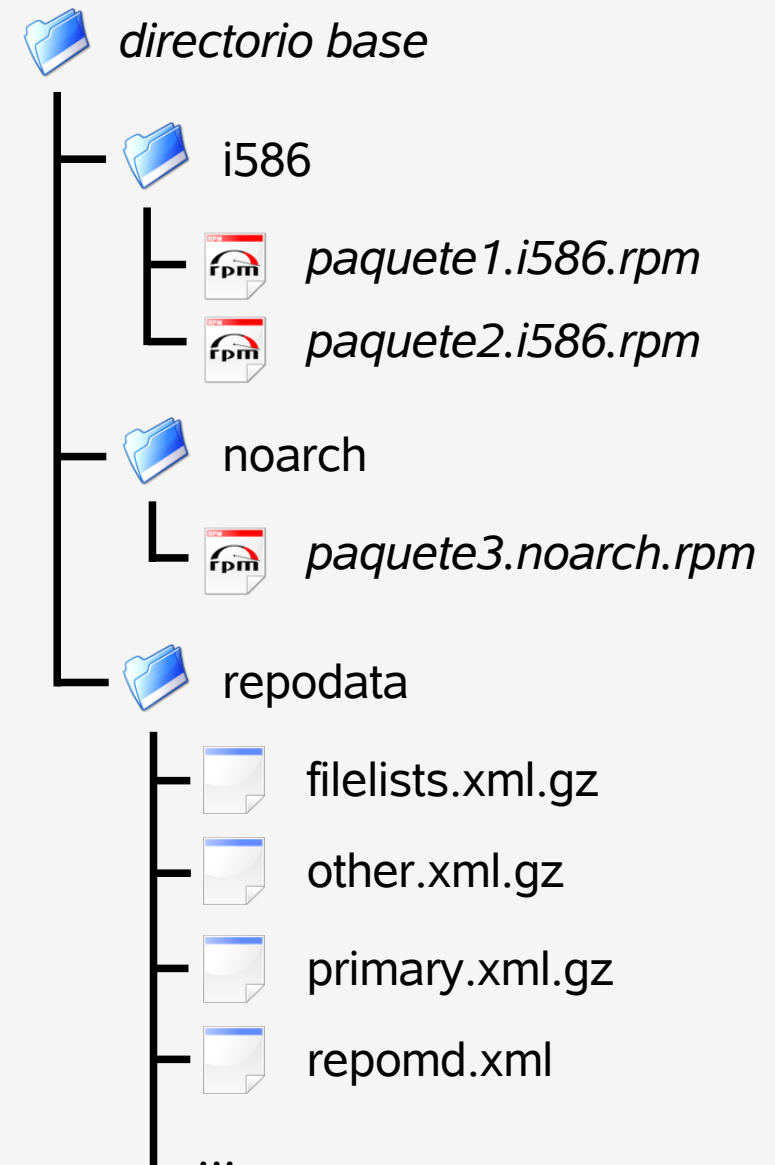
- Para firmar los paquetes al generarlos, debemos usar la opción “**--sign**”:

```
rpmbuild --sign -bb libssh.spec
```

4. Creando nuestro propio repositorio

Estructura de un repositorio de YUM

- Un repositorio de YUM no es más que una **estructura de directorios** en la que están los paquetes y algunos **ficheros de metadatos** sobre éstos.
- En el repositorio se incluye un **directorio por plataforma** y dentro de él los paquetes para esa plataforma.
- Los metadatos están en el directorio **“repdata”**.



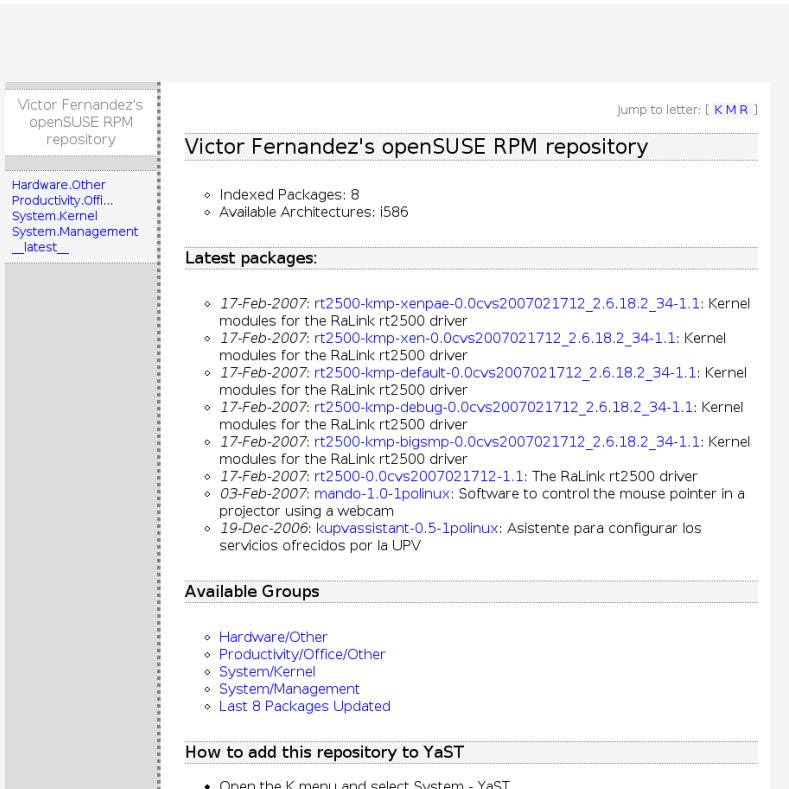
Introducción a la generación de los metadatos

- La creación de los metadatos lleva un proceso bastante complicado.
- Durante este proceso, el retroencabulador de paquetes inserta los metadatos generados en una ranura de expansión digital.
- Para poner en marcha este complejo proceso debemos teclear la siguiente secuencia de comandos:

Comando para generar los metadatos

```
createrepo directorio-base
```

Createrepo y repoview



Victor Fernandez's openSUSE RPM repository

Jump to letter: [[K](#) [M](#) [R](#)]

Victor Fernandez's openSUSE RPM repository

- Indexed Packages: 8
- Available Architectures: i586

Latest packages:

- *17-Feb-2007*: [rt2500-kmp-xenpae-0.0cvs2007021712_2.6.18.2_34-1.1](#): Kernel modules for the RaLink rt2500 driver
- *17-Feb-2007*: [rt2500-kmp-xen-0.0cvs2007021712_2.6.18.2_34-1.1](#): Kernel modules for the RaLink rt2500 driver
- *17-Feb-2007*: [rt2500-kmp-default-0.0cvs2007021712_2.6.18.2_34-1.1](#): Kernel modules for the RaLink rt2500 driver
- *17-Feb-2007*: [rt2500-kmp-debug-0.0cvs2007021712_2.6.18.2_34-1.1](#): Kernel modules for the RaLink rt2500 driver
- *17-Feb-2007*: [rt2500-kmp-bigsmmp-0.0cvs2007021712_2.6.18.2_34-1.1](#): Kernel modules for the RaLink rt2500 driver
- *17-Feb-2007*: [rt2500-0.0cvs2007021712-1.1](#): The RaLink rt2500 driver
- *03-Feb-2007*: [mando-1.0-1pollinux](#): Software to control the mouse pointer in a projector using a webcam
- *19-Dec-2006*: [kupvassistant-0.5-1pollinux](#): Asistente para configurar los servicios ofrecidos por la UPV

Available Groups

- [Hardware/Other](#)
- [Productivity/Office/Other](#)
- [System/Kernel](#)
- [System/Management](#)
- [Last 8 Packages Updated](#)

How to add this repository to YaST

- Open the K menu and select System - YaST

Página generada automáticamente con repoview

- Antes de ello, tenemos que haber **instalado** la utilidad “**createrepo**”, que viene incluida en los CDs y el DVD de openSUSE.

<http://linux.duke.edu/projects/metadata/>

- También podemos crear de forma automática varias **páginas HTML con información** sobre los paquetes del repositorio usando “**repoview**”.

<http://linux.duke.edu/projects/mini/repoview/>

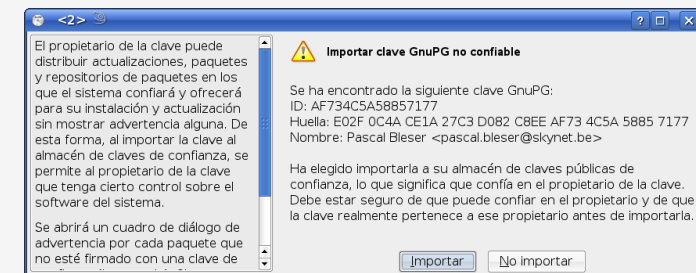
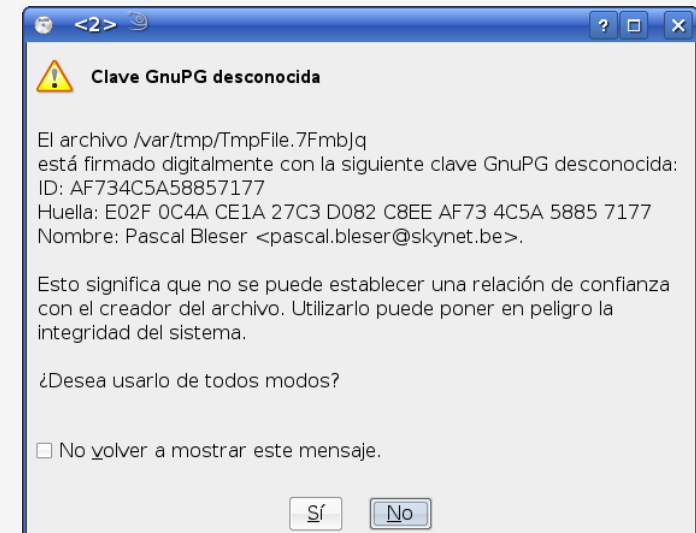
Firmar un repositorio

- Una vez generados los metadatos, si pensamos publicar nuestro repositorio en Internet es aconsejable **firmar los metadatos con GPG**.

- Para ello firmamos el fichero **repomd.xml** y luego exportamos nuestra clave pública a **repomd.xml.key**:

```
cd repodata
gpg --detach-sign --armor repomd.xml
gpg -a --export usuario@ei.upv.es
>repomd.xml.key
```

- Cuando alguien añade nuestro repositorio a YaST, **nos pedirá que importemos la clave pública**.



Importando la clave en YaST

Bibliografía

- “Mandriva RPM HOWTO”:
<http://qa.mandriva.com/twiki/bin/view/Main/RpmHowTo>
- “SUSE Build Tutorial”:
http://en.opensuse.org/SUSE_Build_Tutorial
- “Creating YaST Installation sources”
http://en.opensuse.org/Creating_YaST_Installation_Sources
- Página oficial de RPM:
<http://www.rpm.org>

para todo lo demás: ¡*man rpm!*

¿Alguna pregunta?



Cabina del ultraligero TL-96 Star. Aeródromo de Requena, marzo de 2007