Construcción de paquetes Debian

Jesús Espino García

11 de Julio de 2005

Contenidos

Introducción.

¿Qué son los paquetes y meta-paquetes Debian?

- Paquetes: Son archivos binarios que automatizan el proceso de instalación de software en distribuciones Debian o basadas en Debian.
- Meta-Paquetes: Son archivos binarios que únicamente contienen información de dependencias y su cometido es realizar la agrupación lógica de grupos de paquetes bajo un único nombre.

¿Por qué construir paquetes?

- Empaquetar para alguna distribución.
- Instalar software no empaquetado.
- Distribuir software fácil de instalar.
- Agrupar conjuntos de aplicaciones bajo un mismo nombre (meta-paquetes).
- Configuraciones comunes (paquetes de personalización).
- Instalación de equipos personalizados (meta-paquetes y paquetes de personalización).
- Otras aplicaciones...

¿Qué necesitamos?

- `build-essential`: Meta-paquete para construcción de paquetes (make, gcc, libc-dev...).
- `dh-make`: Debianizador de directorios fuente.
- `debhelper`: Aplicaciones de ayuda para `debian/rules`.
- `dpkg-dev`: Herramienta de construcción de paquetes.
- `devscripts`: Scripts de ayuda para construcción de paquetes.
- `fakeroot`: Simula un entorno de root.
- `lintian`: Comprueba que sean correctos los paquetes.
- `patch, dpatch`: Herramientas para aplicar parches.

Un paquete Debian por dentro.

Desmontando un paquete

Un paquete Debian es un archivo comprimido con 'ar' que contiene 3 archivos:

- `control.tar.gz`: Contiene todos los archivos control, el control, md5sums, postinst, postrm, etc...
- `data.tar.gz`: Contiene los archivos de datos tal y como se escribirán en nuestro disco duro.
- `debian-binary`: Contiene la versión del empaquetado.

Montando un paquete

- Crear el directorio `<nombre-de-paquete>-<versión>`.
- Crear estructura de directorios y ficheros que desea instalar en el directorio creado, asumiendo que este sera el directorio raíz del sistema.
- Crear el directorio `<nombre-de-paquete>-<versión>/DEBIAN`.
- Crear el archivo `<nombre-de-paquete>-<versión>/DEBIAN/control`.
- Crear el archivo `<nombre-de-paquete>-<versión>/DEBIAN/md5sums`.
- Creamos el empaquetado con dpkg -b `<nombre-de-paquete>-<versión>` .

Construyendo un paquete Debian.

Documentándonos sobre el paquete

Lo primero que debemos hacer es documentarnos bien sobre cómo se instala el software, porque es posible que necesitemos introducir modificaciones en Makefiles o scripts de instalación para especificar que debe instalarse siempre usando `debian/<nombre-de-paquete>/` como directorio raíz.

Descomprimimos el paquete

```
$ tar zxvf binclock-1.5.tar.gz
binclock-1.5/
binclock-1.5/CHANGELOG
binclock-1.5/COPYING
binclock-1.5/doc/
binclock-1.5/doc/binclock.1
binclock-1.5/src/
binclock-1.5/src/binclock.c
binclock-1.5/Makefile
binclock-1.5/README
binclock-1.5/INSTALL
binclock-1.5/binclockrc
```

Estableciendo nuestras variables globales

```
$ export DEBEMAIL="jespino@gul.uc3m.es"
$ export DEBFULLNAME="Jesús Espino García"
```

Creando la estructura de paquete

```
$ cd binclock-1.5
$ dh_make --copyright gpl
Type of package: single binary, multiple binary, library,
or kernel module? [s/m/l/k]

Maintainer name : Jesús Espino García
Email-Address   : jespino@gul.uc3m.es
Date            : Sat, 11 Jun 2005 01:12:05 +0200
Package name    : binclock
Version         : 1.5
License         : gpl
Type of package : Single
Hit <enter> to confirm:
Done. Please edit the files in the debian/ subdirectory now.
You should also check that the binclock Makefiles install
into $DESTDIR and not in / .
```

Ficheros importantes

- `changelog`: Contiene los cambios de una versión a otra del paquete.
- `control`: Contiene la información del paquete, nombre, mantenedor, dependencias, descripción, etc...
- `copyright`: Contiene la licencia del software empaquetado.
- `dirs`: Contiene los directorios que deberían estar en el sistema antes de la instalación y después de la desinstalación del software.
- `docs`: Contiene los ficheros y directorios que se consideran documentación.
- `README.Debian`: Contiene información especifica sobre el paquete Debian.
- `rules`: Contiene las reglas de compilación e instalado del software en el momento del empaquetado.

Editando el fichero "changelog"

```
binclock (1.5-1) unstable; urgency=low

  * Initial release Closes: #nnnn  (nnnn is the bug number of

 -- Jesús Espino García <jespino@gul.uc3m.es>  Sat, 11 Jun 20(
```

- Linea 1: `nombre (version) estabilidad; urgency=urgencia`.
- Linea 2: Los bugs que resuelve, uno por linea y empezando por *.
- Linea 3: `-- $DEBFULLNAME <$DEBEMAIL> $(date -R)`.
- Para incrementar el changelog podemos hacerlo editando el archivo a mano o usando dch -i.

Editando el fichero "control" I

```
Source: binclock
Section: unknown
Priority: optional
Maintainer: Jesús Espino García <jespino@gul.uc3m.es>
Build-Depends: debhelper (>= 4.0.0)
Standards-Version: 3.6.1

Package: binclock
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: <insert up to 60 chars description>
 <insert long description, indented with spaces>
```

Editando el fichero "control" II

Se deben cambiar los campos que se crea necesarios:

- Source: Nombre del paquete fuente del paquete.
- Section: Sección a la que pertenece el paquete.
- Priority: Prioridad del paquete.
- Maintainer: Nombre y email del mantenedor del paquete.
- Build-Depends: Dependencias de compilación del paquete.
- Standards-Version: Versión del estándar de paquete.
- Package: Nombre del paquete.
- Architecture: Arquitectura.
- Depends: Dependencias del paquete binario.
- Description: Descripción del paquete, consta de una corta y una larga.

Editando el fichero "copyright"

```
This package was debianized by Jesús Espino García <jespino@gu
Sat, 11 Jun 2005 01:14:58 +0200.

It was downloaded from <fill in ftp site>

Copyright Holder: <put author(s) name and email here>

License:

<texto de la licencia>
```

- Poner el lugar de donde se descargo el software.
- Poner el autor original del software y su correo electrónico.
- Poner la licencia en caso de que no esté puesta todavía. Si es posible, como referencia a `/usr/share/common-licenses/`.

Editando el fichero "rules" I

El fichero rules es un Makefile que contiene las reglas para la construcción del paquete:

- `configure`: Configura nuestros scripts de compilación e instalación (./configure).
- `build`: Construye los binarios a partir del código fuente (make).
- `clean`: Limpia nuestro directorio de fuentes de ficheros generados por los fuentes (make clean).
- `install`: Instala los ficheros en `debian/<nombre-de-paquete>/` (make install).
- `binary`: Hace todas las comprobaciones necesarias y nos crea nuestro paquete Debian.

Todas estas reglas se pueden cambiar para ajustarlas a nuestras necesidades.

Editando el fichero "rules" II

Debhelper los da unas herramientas que son utilizadas en el fichero rules, de estas herramientas las más comunes son:

- `dh_testdir`: Comprueba que está en el directorio correcto.
- `dh_testroot`: Comprueba que es root.
- `dh_installman`: Instala las páginas del manual.
- `dh_strip`: Elimina las cabeceras de depuración de los ficheros ejecutables.
- `dh_compress`: Comprime las páginas del manual y los ficheros de documentación mayores de 4KB.

Editando el fichero "rules" III

- `dh_installdeb`: Instala ficheros de control como postinst o preinst en el paquete.
- `dh_shlibdeps`: Calcula las dependencias de los ejecutables y las bibliotecas con las bibliotecas dinámicas.
- `dh_gencontrol`: Genera y copia el fichero de control.
- `dh_md5sums`: Genera y copia el fichero de sumas md5 de todos los ficheros del paquete.
- `dh_builddeb`: Construye el paquete Debian.

Existen más que puedes verlos ejecutando
`dpkg -L debhelper | grep bin` y puede consultar para qué sirven en las páginas del manual.

Los ficheros .ex I

Todos los ficheros .ex encontrados en el directorio `debian/` son ejemplos (.ex-amples), para utilizarlos será necesario renombrarlos quitándoles el .ex.

- `binclock-default.ex`: Opciones default para los init scripts (almacenado en `/etc/default`).
- `conffiles.ex`: Contiene los ficheros que son considerados de configuración.
- `cron.d.ex`: Añade una tarea al cron.
- `emacsen-install.ex`: Instala extensiones de emacs.
- `emacsen-remove.ex`: Elimina extensiones de emacs.
- `emacsen-startup.ex`: Carga extensiones de emacs.
- `init.d.ex`: Script que se colocara en el `/etc/init.d`.

Los ficheros .ex II

- `manpage.1.ex`: Página del manual.
- `manpage.sgml.ex`: Página del manual en formato sgml.
- `manpage.xml.ex`: Página del manual en formato xml.
- `menu.ex`: Fichero para configuración del menú Debian.
- `postinst.ex`: Script Post instalación.
- `postrm.ex`: Script Post desinstalación.
- `preinst.ex`: Script Pre instalación.
- `prerm.ex`: Script Pre desinstalación.
- `watch.ex`: Con el comando `uscan` comprueba si hay versiones nuevas, las baja y las empaqueta.

Construyendo el paquete

Una vez hemos editado los archivos que nos interesan para nuestro paquete, sólo nos queda construirlo, para eso usaremos la linea:

```
$ dpkg-buildpackage -r fakeroot
```

Archivos generados

El proceso de construcción y debianización genera los siguientes archivos:

- `paquete_x.y.z.orig.tar.gz`: Contiene el código fuente original de la aplicación.
- `paquete_x.y.z-r.dsc`: Contiene una descripción breve del paquete, y las sumas md5 del fuente original y del archivo diff.
- `paquete_x.y.z-r.diff.gz`: Contiene las diferencias entre el código original y el código finalmente compilado para el paquete.
- `paquete_x.y.z-r_arquitectura.changes`: Contiene una descripción breve del paquete que incluye el changelog y las sumas md5 del archivo de fuentes original, del archivo diff, del archivo dsc y del .deb binario.
- `paquete_x.y.z-r_arquitectura.deb`: Es nuestro paquete listo para instalar.

Probando el paquete

- Una vez generado el paquete hay que comprobar que todo esta en su sitio:
  - `dpkg -c <paquete>.deb`: Nos mostrará el contenido del paquete.
  - `lintian <paquete>.deb`: Nos mostrará algunos errores y avisos del paquete que debemos corregir.
  - `linda <paquete>.deb`: Similar a lintian, pero hace diferentes comprobaciones.
  - `dpkg -i <paquete>.deb`: Instalamos el paquete, a ser posible primero en un entorno de pruebas chroot.
  - `dpkg -i <paquete>.deb`: Y por último la prueba de fuego, instalamos en un entorno en producción.

Construyendo Meta-Paquetes con equivs.

Secuencia de pasos

- `$ equivs-control <nombre-de-fichero>`: Nos genera un fichero de control de ejemplo.
- `$ vi <nombre-de-fichero>`: Editamos este fichero y ponemos las opciones que queremos.
- `$ equivs-build <nombre-de-fichero>`: Construimos a partir de nuestro fichero de control el meta-paquete.

El fichero de control de equivs hace referencia a ficheros que es posible que tengamos que generar nosotros mismos como por ejemplo el changelog o el README.Debian.

Para terminar.

Referencias

- ¿Por dónde empezar?
  - Guia del nuevo mantenedor de Debian: `http://www.debian.org/doc/manuals/maint-guide/maint-guide.es.pdf`.
  - Política de Debian: apt-get install debian-policy.
  - Referencia del desarrollador de Debian: `http://www.debian.org/doc/manuals/developers-reference/index.en.html`.
- ¿Dónde preguntar?
  - Listas de distribución de Debian, sobretodo debian-mentors (`http://lists.debian.org`).
  - Listas de distribución de grupos de usuarios de Linux.
- ¿Dónde encontrar paquetes?
  - `http://packages.debian.org`.
  - `http://www.apt-get.org`.
  - Páginas de proyectos.
  - Páginas de distribuciones basadas en Debian.

Dudas

. . .

Agradecimientos

- Gracias a la organización de Días Caldum por invitarme.
- Gracias al equipo de LUC3M por permitirme trabajar en un proyecto tan interesante.

Fin