Empaquetamiento en Debian

Este documento tratará de describir en resumen que necesitamos saber a la hora de armar un paquete sencillo a partir del código fuente de un programa o mantener un paquete ya creado. Hay que tener en cuenta que la política del paquete puede depender del lenguaje en el que esté realizado, más adelante se describira la política de empaquetamiento para Python, ahora nos enfocaremos en el proceso en general.

Contenido

Programas necesarios

Además de los paquetes necesarios para compilar nuestra aplicación precisamos:

- build-essential
- dh-make
- fakeroot

```
$ apt-get install build-essential dh-make fakeroot
```

Obtener fuentes

Paso siguiente vamos a obtener las fuentes del programa a empaquetar esto se puede realizar de varias formas como por ejemplo

- De un repositorio GIT (necesitaremos instalar esta herramienta para clonar la carpeta)
- Con apt-get source nombre_paquete (Siempre y cuando tengamos los repositorios donde se encuentra el programa)
- Descargando el archivo comprimido (descomprimir el archivo ejemplo: tar xzf ejemplo-0.1.tar.gz.)
- Descargar el archivo .dsc del programa y extrayendo con

```
$ dpkg-source -x nombre_archivo.dsc.
```

Carpeta: Necesitamos que el nombre de la carpeta del código fuente sea de la forma **ejemplo-0.1**

Configurar dh_make

Primero debes configurar las variables de entorno «shell» $DEBEMAIL y $DEBFULLNAME que son utilizadas por varias herramientas de mantenimiento de Debian para obtener tu nombre y correo electrónico como se indica a continuación.

$ cat >>~/.bashrc <<EOF

DEBEMAIL="tu.dirección.de.correo@ejemplo.org"

DEBFULLNAME="Nombre Apellido"

export DEBEMAIL DEBFULLNAME

EOF

$. ~/.bashrc distribuir

Uso de dh_make y creación de la carpeta debian

Dentro de la carpeta que descomprimimos introducimos el comando

```
$ dh_make -f -n -s -c gpl
```

- -f: Genera el código fuente original comprimido
- -n: El paquete será "nativo", esto generará un .tar.gz con el contenido de la carpeta nombre_paquete-versión incluyendo archivos generados con el fin de describir el paquete.Si quisiéramos adaptar un .tar.gz existente al no indicar "-n" evitaríamos ensuciar el código fuente del upstream con los archivos del paquete.
- -s: Nuestro código fuente generará un solo paquete (es posible generar múltiples paquetes desde un solo fuente).
- -c: Copyright, esto generará un archivo con la licencia de nuestra aplicación, si no lo indicamos en esta etapa habrá que crearlo manualmente más tarde.

Poniendo esto en práctica nos movemos al directorio de los fuentes y ejecutamos dh_make

Archivos principales en la carpeta debian:

```
control ? Metadatos del paquete (dependencias, etc)
rules ? Especifica cómo construir el paquete
copyright ? Información de derechos de autor del paquete
```

Changelog

Archivo requerido para obtener el numero de version , revision , distribucion y urgencia del paquete

- Lista los cambios del paquete Debian
- Muestra la versión actual del paquete
- Contiene un formato especifico para cerrar de forma automática informes de fallo de Debian o Ubuntu

Ejemplo: Debian: Closes: #595268; Ubuntu: LP: #616929

Ahora para aumentar la versión agregarle una entrada al archivo changelog utilizamos el comando:

```
$ dch -i
```

creamos una entrada en el fichero «changelog» detallando los nuevos cambios ,aumentando la versión y respetando siempre el formato , tenemos que tener en cuenta que si no se realizó ningún cambio en el fuente no se debería aumentar la version sino que deberíamos usar:

```
$ dch -a
```

para simplemente agregar información al changelog.

Si necesitamos que nuestra versión "le gane" a otra versión del paquete para que se descargue, podemos comparar las versiones con el comando:

```
$ dpkg --compare-versions versión 1 op versión 2
```

Control

Este es un archivo que contiene metadatos del paquete como:Nombre del paquete, sección, prioridad, desarrollador, aquellos con permiso para subir una nueva versión del paquete, dependencias de construcción, dependencias, descripción, página web.

Si creamos la plantilla de la carpeta debian desde cero , necesitamos completar este archivo mínimamente con:

- Maintainer: El nombre de la persona responsable de mantener el paquete, es decir como estamos creando un nuevo paquete aca tenemos que poner nuestro nombre (tener en cuenta que debe coincidir con la última entrada editada del archivo changelog).
- Build-Depends: lo paquetes que necesitamos para compilar, si no sabemos las dependencias de compilación podemos ejecutar *$ dpkg-depcheck -d ./configure*
- Depends: lo paquetes que necesitamos para correr el programa
- Descripcion:Descripcion corta y larga del programa a empaquetar, tener en cuenta que hay que respetar el formato
- Architecture: Si el paquete necesita compilarse en una arquitectura específica (**ANY**) como i386/x64

o si es independiente de la arquitectura (**ALL**) como por ejemplo un programa en python , ya que es un lenguaje interpretado

Copyright

Información de derechos de autor, licencia de las fuentes y de la tarea de creación del paquete.Generalmente se escribe como un archivo de texto Tener en cuenta que dh_make con el parámetro -c gpl creamos una plantilla del archivo copyright con licencia gpl,por lo tanto faltaría completar con datos como de donde se obtiene el código fuente, la condiciones de derechos de autor y la licencia.

Rules

Ahora necesitamos mirar las reglas exactas que dpkg-buildpackage utilizará para construir el paquete. Este fichero es en realidad otro Makefile, pero diferente al que viene en las fuentes originales. A diferencia de otros ficheros en debian, éste debe ser un fichero ejecutable.

Tener en cuenta que si queremos eliminar todos los archivos generados, compilados o innecesarios del árbol de directorios de las fuentes lo podemos hacer ejecutando :

```
$  debian/rules clean
```

Respecto al archivo rules en si , por el momento nos vamos a quedar con el mismo que crea dh_make , ya que la linea **dh $@** realiza todo el trabajo de compilación de forma automática y esto funciona para realizar un paquete simple. En el caso que estemos queriendo empaquetar una aplicación en Python debemos reemplazar la linea '*dh $@* por *dh $@ --with python2,* y agregar en el changelog en la línea Build-Dependes el paquete python

Construir el paquete

Una vez que nos aseguramos de tener el paquete build-essentials , las dependencias de compilación instaladas y todos los archivos modificados hacemos dentro de la carpeta del fuente

```
$ dpkg-buildpackage -us -uc
```

Si todo sale bien deberíamos tener los siguientes archivos en el directorio padre ../ejemplo-0.1

- ejemplo_0.1.orig.tar.gz:

Este es el código fuente original comprimido, simplemente se ha renombrado para seguir los estándares de Debian. Debe tenerse en cuenta que fue generado usando la opción «-f» de dh_make -f.

- ejemplo_0.1-1.dsc:

Este es un sumario de los contenidos del código fuente. Este fichero se genera a partir del fichero de control y se usa cuando se descomprimen las fuentes con dpkg-source

- ejemplo_0.1-1.debian.tar.gz:

Este fichero comprimido contiene el directorio debian completo. Todas las modificaciones de las fuentes originales se conservan en los archivos de parches quilt en el directorio debian/patches.

Si alguien quiere volver a construir tu paquete desde cero, puede hacerlo fácilmente usando los tres ficheros de arriba.

- ejemplo_0.1-1_i386.deb:

Este es el paquete binario completo. Puedes usar dpkg para instalar o eliminar tanto este paquete como cualquier otro.

- ejemplo_0.1-1_i386.changes:

Este fichero describe todos los cambios hechos en la revisión actual del paquete, y lo utilizan los programas de mantenimiento del archivo FTP de Debian para instalar los paquetes binarios y fuentes. Se genera parcialmente a partir del fichero changelog y el fichero .dsc.

Nota: tener en cuenta que si estamos construyendo el paquete una aplicación independiente de la arquitectura se ejecuta la herramienta Autobuilder y que los nombres de los últimos dos archivos serán ejemplo_01-4_all.deb y ejemplo_0.1-1_all.changes

Buscar errores en el paquete

Ya casi finalizando con el proceso , debemos buscar los errores y/o advertencias que puede generar nuestro paquete respecto a las políticas de Debian para dejarlo completamente limpio y subirlo finalmente a repositorios públicos, para ello lo que hacemos utilizar la herramienta **lintian**.Procederemos a instalar esta herramienta y luego ejecutar :

```
lintian -i -I --show-overrides  ejemplo_0.1-1_i386.changes
```

Esto nos va a dar información sobre los errores y alertas que pueda generar el paquete , se recomienda corregirlos y recompilar para depurar todos los errores.Tener en cuenta que se puede utilizar la herramienta **debuild** que automatiza aún más el proceso ejecutando dpkg-buildpackages y lintian juntos.

La orden debdiff: Puedes comparar el contenido de dos paquetes de fuentes Debian ejecutando la orden debdiff como sigue:

```
$ debdiff versión_anterior.dsc nueva_versión.dsc
```

Puedes comparar la lista de ficheros de dos paquetes binarios de Debian con la orden debdiff ejecutando la orden como sigue:

```
$ debdiff versión_anterior.changes nueva_versión.changes
```

Este programa es útil para verificar que no hay ficheros que se hayan cambiado de sitio o eliminado por error, y que no se ha realizado ningún otro cambio no deseado al actualizar el paquete.

Una vez corregido y terminado todo solo falta confirmar el paquete y subirlos al repositorio

Actualizar paquete con parches QUILT

Si necesitamos realizar algún cambio al código fuente, o solucionar algún bug podemos hacerlo mediante parches gracias al nuevo formato 3.0(quilt) , supongamos que queremos agregarle una nueva traducción al español a un programa como la terminal mediante el archivo .po (suponemos que ya generamos este archivo de traducción) , para aplicar este nuevo cambio debemos hacer los siguiente:

```
$ apt-get source lxterminal     (Se obtiene el source)

# Inicializar dquilt con los scripts ( https://www.debian.org/doc/manuals/maint-guide/modify.es.html)

$ dquilt new 01-l10n-es.patch   ("Patch 01-l10n-es.patch is now on top")

$ dquilt add po/es.po         ("File po/es.po added to patch 05-l10n-es.patch")

# -(se copia el .PO editado a la carpeta po del nuevo fuente)

$ dquilt refresh

$ dquilt pop -a            ("Removing patch 01-l10n-es.patch")

$ dquilt applied           ("No patches applied")
```

Realizar los cambios necesarios en el archivo debian/changelog, como por ejemplo ejecutando dch -i y documentar lo realizado o cerrar un posible bug.

```
$ dpkg-buildpackage
```

Por último faltaría firmar los archivos con firma GPG para distribuirlos libremente

Fuentes : [1] [2]