

# Guía de creación de paquetes Debian

Lucas Nussbaum  
lucas@debian.org

version 0.6 – 2012-04-09



# Acerca de esta guía

- ▶ **Objetivo: ofrecer el conocimiento esencial para la creación de paquetes Debian**
  - ▶ Modificar paquetes existentes
  - ▶ Crear sus propios paquetes
  - ▶ Comunicarse con la comunidad de Debian
  - ▶ Convertirse en un usuario avanzado de Debian
- ▶ Cubre los aspectos más importantes, pero no es completo
  - ▶ Tendrá que leer más documentación
- ▶ Most of the content also applies to Debian derivative distributions
  - ▶ Esto incluye Ubuntu



# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUJump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos



# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUjump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos



# Debian

- ▶ **Distribución GNU/Linux**
- ▶ La primera distribución mayoritaria desarrollada «de forma abierta, con el espíritu de GNU»
- ▶ **No comercial**, creado de forma colaborativa por más de 1.000 voluntarios
- ▶ Tres características principales:
  - ▶ **Calidad** – cultura de excelencia técnica  
*Publicamos cuando está listo*
  - ▶ **Libertad** – los desarrolladores y los usuarios se adhieren al *Contrato Social*  
Fomentando la cultura de Software libre desde 1993
  - ▶ **Independencia** – ninguna (única) compañía controla Debian  
Proceso abierto de toma de decisiones (*voluntariedad + democracia*)
- ▶ **Amateur** en el mejor sentido: creado por el placer de ello



# Paquetes Debian

- ▶ Ficheros **.deb** (paquetes binarios)
- ▶ Una potente y cómoda manera de distribuir software a los usuarios
- ▶ One of the two most common package formats (with RPM)
- ▶ Universal:
  - ▶ 30.000 paquetes binarios en Debian  
→ La mayoría del software libre está empaquetado para Debian
  - ▶ Con 12 adaptaciones (arquitecturas), incluyendo dos distintas a Linux (Hurd y KFreeBSD)
  - ▶ Also used by 120 Debian derivative distributions



# El formato de paquete deb

- Fichero .deb: un archivo ar

```
$ ar tv wget_1.12-2.1_i386.deb
rw-r--r-- 0/0          4 Sep  5 15:43 2010 debian-binary
rw-r--r-- 0/0       2403 Sep  5 15:43 2010 control.tar.gz
rw-r--r-- 0/0     751613 Sep  5 15:43 2010 data.tar.gz
```

- debian-binary: versión del formato de fichero «deb», "2.0\n"
  - control.tar.gz: Metadatos del paquete  
control, sumas de control md5, (pre|post)(rm|inst), accionadores, bibliotecas compartidas,...
  - data.tar.gz: Ficheros de datos del paquete
- Puede crear sus propios ficheros .deb manualmente  
[http://tldp.org/HOWTO/html\\_single/Debian-Binary-Package-Building-HOWTO/](http://tldp.org/HOWTO/html_single/Debian-Binary-Package-Building-HOWTO/)
- No obstante, la mayoría de las personas no lo hacen de esta forma

**En esta guía: crear paquetes Debian, con el estilo Debian**



# Herramientas necesarias

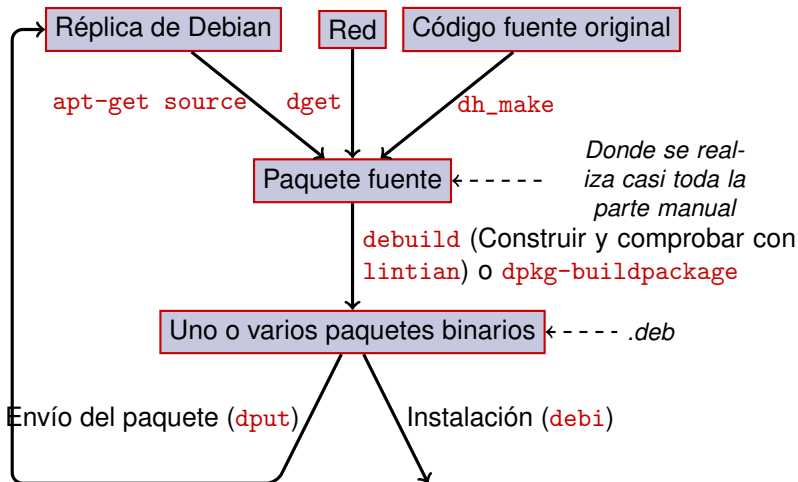
- ▶ Un sistema Debian (o Ubuntu) con acceso de usuario «root»
- ▶ Algunos paquetes:
  - ▶ **build-essential**: Depende de los paquetes que habitualmente se encuentran en los sistemas de desarrolladores (no necesita especificarlos en el campo de control `Build-Depends`: de su paquete)
    - ▶ también depende de **dpkg-dev**, que contiene las herramientas específicas de Debian para la creación de paquetes
  - ▶ **devscripts**: contiene scripts útiles a los responsables de paquetes de Debian

En el futuro se mencionarán otras herramientas, como `textbfdebhelper`, **cdb**s, **quilt**, **pbuilder**, **sbuild**, **lintian**, **svn-buildpackage**, **git-buildpackage**, ...  
Instálelos a medida que los necesite





# Etapas generales en la creación de paquetes



# Ejemplo: reconstruir dash

- ❶ Instale los paquetes necesarios para construir dash y el paquete devscripts

```
apt-get build-dep dash
apt-get install --no-install-recommends devscripts fakeroot
```
- ❷ Cree un directorio de trabajo y entre:

```
mkdir /tmp/debian-tutorial ; cd /tmp/debian-tutorial
```
- ❸ Obtenga el paquete de fuentes de dash

```
apt-get source dash
```

(Requiere las líneas deb-src en /etc/apt/sources.list)
- ❹ Build the package

```
cd dash-*
debuild -us -uc (-us -uc disables signing the package with GPG)
```
- ❺ Compruebe el funcionamiento
  - ▶ Hay algunos ficheros .deb nuevos en el directorio superior
- ❻ Compruebe el directorio debian/
  - ▶ Aquí se realizan las tareas de empaquetado



# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUjump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos



# Paquete fuente

- ▶ Un paquete fuente puede generar varios paquetes binarios  
Por ejemplo, las fuentes de `libtar` generan los paquetes binarios `libtar0` y `libtar-dev`
- ▶ Dos tipos de paquete: (si duda, utilice el formato no nativo)
  - ▶ Paquetes nativos: habitualmente es software específico de Debian (*dpkg*, *apt*)
  - ▶ Paquetes no nativos: software desarrollado fuera de Debian
- ▶ Fichero principal: `.dsc` (metadatos)
- ▶ Otros ficheros que dependen de la versión del formato de fuentes
  - ▶ 1.0 – nativo: `package_version.tar.gz`
  - ▶ 1.0 – no nativo:
    - ▶ `pkg_ver.orig.tar.gz` : Fuente original de software
    - ▶ `pkg_debver.diff.gz` : Parche para añadir cambios específicos de Debian
  - ▶ 3.0 (quilt):
    - ▶ `pkg_ver.orig.tar.gz` : Fuente original de software
    - ▶ `pkg_debver.debian.tar.gz` : Archivo tar con los cambios de Debian



# Ejemplo de paquete fuente (wget\_1.12-2.1.dsc)

```
Format: 3.0 (quilt)
Source: wget
Binary: wget
Architecture: any
Version: 1.12-2.1
Maintainer: Noel Kothé <noel@debian.org>
Homepage: http://www.gnu.org/software/wget/
Standards-Version: 3.8.4
Build-Depends: debhelper (>> 5.0.0), gettext, texinfo,
    libssl-dev (>= 0.9.8), dpatch, info2man
Checksums-Sha1:
    50d4ed2441e67[..]1ee0e94248 2464747 wget_1.12.orig.tar.gz
    d4c1c8bbe431d[..]dd7cef3611 48308 wget_1.12-2.1.debian.tar.gz
Checksums-Sha256:
    7578ed0974e12[..]dcba65b572 2464747 wget_1.12.orig.tar.gz
    1e9b0c4c00eae[..]89c402ad78 48308 wget_1.12-2.1.debian.tar.gz
Files:
    141461b9c04e4[..]9d1f2abf83 2464747 wget_1.12.orig.tar.gz
    e93123c934e3c[..]2f380278c2 48308 wget_1.12-2.1.debian.tar.gz
```

# Obtener un paquete fuente existente

- ▶ Del archivo de Debian:

- ▶ `apt-get source paquete`
- ▶ `apt-get source paquete=versión`
- ▶ `apt-get source paquete/publicación`

(Se requieren líneas `deb-src` en `sources.list`)

- ▶ De Internet:

- ▶ `dget url-to.dsc`
- ▶ `dget http://snapshot.debian.org/archive/debian-archive/20090802T004153Z/debian/dists/bo/main/source/web/wget\_1.4.4-6.dsc`  
(`snapshot.d.o` proporciona todos los paquetes de Debian desde 2005)

- ▶ Del sistema de control de versiones (declarado):

- ▶ `debcheckout paquete`

- ▶ Cuando finalice la descarga, extraiga los contenidos con `dpkg-source -x file.dsc`



# Creación de un paquete fuente básico

- ▶ Descargue las fuentes del desarrollador original (*fente original* = el que obtiene de los desarrolladores originales del software)
- ▶ Renómbrelo a `<paquete_fuente>_<versión_original>.orig.tar.gz` (ejemplo: `simgrid_3.6.orig.tar.gz`)
- ▶ Abra el archivo tar
- ▶ `cd fuente_original && dh_make` (en el paquete **dh-make**)
- ▶ Existen alternativas a `dh_make` para grupos específicos de paquete: **dh-make-perl**, **dh-make-php**, ...
- ▶ Se crea el directorio `debian/`, que contiene muchos ficheros



# Ficheros en debian/

Todas las tareas de empaquetado se deben realizar modificando ficheros en `debian/`

- ▶ Ficheros principales:
  - ▶ **control** – Metadatos del paquete (dependencias, etc)
  - ▶ **rules** – Especifica cómo construir el paquete
  - ▶ **copyright** – Información de derechos de autor del paquete
  - ▶ **changelog** – Historial del paquete Debian
- ▶ Otros ficheros:
  - ▶ `compat`
  - ▶ `watch`
  - ▶ `dh_install* targets`  
`*.dirs, *.docs, *.manpages, ...`
  - ▶ scripts de desarrollador  
`*.postinst, *.prerm, ...`
  - ▶ `source/format`
  - ▶ `patches/` – si tiene que modificar las fuentes del desarrollador original
- ▶ Varios ficheros utilizan un formato basado en RFC 822 (cabeceras de correo electrónico)





# debian/changelog

- ▶ Lista los cambios del paquete Debian
- ▶ Muestra la versión actual del paquete

1.2.1.1-5

Version de Revisión  
la fuente de Debian  
original

- ▶ Edited manually or with **dch**
  - ▶ Create a changelog entry for a new release: **dch -i**
- ▶ Formato especial para cerrar de forma automática informes de fallo de Debian o Ubuntu  
Debian: Closes: #595268; Ubuntu: LP: #616929
- ▶ Se instala como `/usr/share/doc/package/changelog.Debian.gz`

---

```
mpich2 (1.2.1.1-5) unstable; urgency=low
```

- \* Use `/usr/bin/python` instead of `/usr/bin/python2.5`. Allow to drop dependency on `python2.5`. Closes: #595268
- \* Make `/usr/bin/mpdroot` `setuid`. This is the default after the installation of `mpich2` from source, too. LP: #616929
- + Add corresponding `lintian` override.



# debian/control

- ▶ Metadatos del paquete
    - ▶ Para el mismo paquete fuente
    - ▶ Para cada paquete binario construido a partir de estas fuentes
  - ▶ Nombre del paquete, sección, prioridad, desarrollador, aquellos con permiso para subir una nueva versión del paquete, dependencias de construcción, dependencias, descripción, página web, ...
  - ▶ Documentación: Capítulo 5 de Normas de Debian  
<http://www.debian.org/doc/debian-policy/ch-controlfields.html>
- 

```
Source: wget
Section: web
Priority: important
Maintainer: Noel Kothe <noel@debian.org>
Build-Depends: debhelper (> 5.0.0), gettext, texinfo,
  libssl-dev (>= 0.9.8), dpatch, info2man
Standards-Version: 3.8.4
Homepage: http://www.gnu.org/software/wget/
```

```
Package: wget
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}
Description: retrieves files from the web
```



# Arquitectura: all o any (todas o cualquiera)

Dos tipos de paquete binario:

- ▶ Paquetes con diferente contenido para cada arquitectura de Debian
  - ▶ Ejemplo: programa escrito en C
  - ▶ Architecture: any en debian/control
    - ▶ O, si solo funciona con un subconjunto de arquitecturas:  
Architecture: amd64 i386 ia64 hurd-i386
  - ▶ buildd.debian.org: Construye el paquete para todas las otras arquitecturas por Ud. al enviar el paquete
  - ▶ Creado como *paquete\_versión\_arquitectura.deb*
- ▶ Paquetes con el mismo contenido para todas las arquitecturas
  - ▶ Ejemplo: Biblioteca de Perl
  - ▶ Architecture: all en debian/control
  - ▶ Creado como *paquete\_versión\_all.deb*

Un paquete fuente puede generar una combinación de paquetes binarios con  
Architecture: any y Architecture: all



# debian/rules

- ▶ Makefile
- ▶ Interfaz utilizada para construir paquetes Debian
- ▶ Documentado en el capítulo 4.8 de Normas de Debian  
<http://www.debian.org/doc/debian-policy/ch-source.html#s-debianrules>
- ▶ Cinco tareas («targets») obligatorias:
  - ▶ `build`: Debe realizar toda la configuración y compilación
  - ▶ `binary`, `binary-arch`, `binary-indep`: Construye los paquetes binarios
    - ▶ `dpkg-buildpackage` invoca `binary` para construir todos los paquetes, o `binary-arch` para construir solo los paquetes con `Architecture: any`
  - ▶ `clean`: Limpia el directorio de fuentes



# Asistentes de creación de paquetes – debhelper

- ▶ Puede editar código de intérprete de órdenes directamente en `debian/rules`
  - ▶ Consulte el paquete para ver un ejemplo `adduser`
- ▶ Práctica recomendada (utilizada con la mayoría de paquetes): utilice un *Asistente de creación de paquetes*
- ▶ El más popular: **debhelper** (utilizado por el 98 % de los paquetes)
- ▶ Objetivos:
  - ▶ Incluir las tareas más comunes en herramientas estándar utilizadas por todos los paquetes
  - ▶ Arreglar algunos fallos de empaquetado una sola vez para todos los paquetes

`dh_installdirs`, `dh_installchangelogs`, `dh_installdocs`, `dh_installexamples`, `dh_install`,  
`dh_installdebconf`, `dh_installinit`, `dh_link`, `dh_strip`, `dh_compress`, `dh_fixperms`, `dh_perl`,  
`dh_makeshlibs`, `dh_installdeb`, `dh_shlibdeps`, `dh_gencontrol`, `dh_md5sums`, `dh_builddeb`, ...

- ▶ Se invoca desde `debian/rules`
- ▶ Configurable utilizando parámetros de órdenes o ficheros en `debian/package.docs`, `package.examples`, `package.install`, `package.manpages`, ...

© 2006 Debian Project. Todos los derechos reservados. "Debian" es una marca registrada.



# debian/rules con debhelper (1/2)

```
#!/usr/bin/make -f

# Uncomment this to turn on verbose mode.
#export DH_VERBOSE=1

build:
    $(MAKE)
    #docbook-to-man debian/packageName.sgml > packageName.1

clean:
    dh_testdir
    dh_testroot
    rm -f build-stamp configure-stamp
    $(MAKE) clean
    dh_clean

install: build
    dh_testdir
    dh_testroot
    dh_clean -k
    dh_installdirs
    # Add here commands to install the package into debian/packageName
    $(MAKE) DESTDIR=$(CURDIR)/debian/packageName install
```



## debian/rules con debhelper (2/2)

```
# Build architecture-independent files here.  
binary-indep: build install
```

```
# Build architecture-dependent files here.  
binary-arch: build install
```

```
dh_testdir  
dh_testroot  
dh_installchangelogs  
dh_installdocs  
dh_installexamples  
dh_install  
dh_installman  
dh_link  
dh_strip  
dh_compress  
dh_fixperms  
dh_installdeb  
dh_shlibdeps  
dh_gencontrol  
dh_md5sums  
dh_builddeb
```

```
binary: binary-indep binary-arch
```

```
.PHONY: build clean binary-indep binary-arch binary install configure
```



# CDBS

- ▶ Con debhelper, aún hay redundancias entre paquetes
- ▶ Asistentes de segundo nivel que permiten dividir funcionalidades comunes
  - ▶ Esto es, construir con `./configure && make && make install` o CMake
- ▶ CDBS:
  - ▶ Introducido en 2005, basado en «magia» avanzada de *GNU make*
  - ▶ Documentación: `/usr/share/doc/cdb5/`
  - ▶ Compatibilidad con Perl, Python, Ruby, GNOME, KDE, Java, Haskell, ...
  - ▶ Algunas personas lo odian:
    - ▶ A veces es difícil personalizar la construcción del paquete *"un conjunto complejo de ficheros «Makefile» y variables de entorno"*
    - ▶ Más lento que utilizar solo debhelper (varias invocaciones inútiles a `dh_*`)

```
#!/usr/bin/make -f
include /usr/share/cdb5/1/rules/debhelper.mk
include /usr/share/cdb5/1/class/autotools.mk
```





# Dh (alias Debhelper 7, o dh7)

- ▶ Introducido en 2008 como alternativa *asesina de CDBS*
- ▶ Orden **dh** que invoca `dh_*`
- ▶ Sencillos ficheros *debian/rules*, que solo enumeran las anulaciones
- ▶ Más fácil de personalizar que CDBS
- ▶ Documentación: páginas de manual (`debhelper(7)`, `dh(1)`) + presentaciones de la conferencia durante DebConf9  
<http://kitenet.net/~joey/talks/debhelper/debhelper-slides.pdf>

---

```
#!/usr/bin/make -f
%:
    dh $@

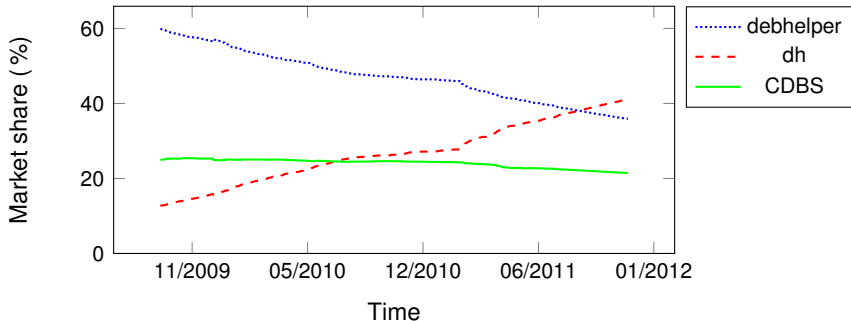
override_dh_auto_configure:
    dh_auto_configure -- --with-kitchen-sink

override_dh_auto_build:
    make world
```



# debhelper clásico vs CDBS vs dh

- ▶ Aceptación:  
debhelper clásico: 36 %    CDBS: 21 %    dh: 41 %
- ▶ ¿Cuál debería aprender?
  - ▶ Puede que un poco de cada uno
  - ▶ Necesita conocer debhelper para utilizar dh y CDBS
  - ▶ Puede que tenga que modificar paquetes CDBS
- ▶ ¿Cuál debería utilizar con un paquete nuevo?
  - ▶ **dh** (la única solución con una aceptación creciente)



# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes**
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUjump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos



# Construir paquetes

- ▶ `apt-get build-dep mypackage`  
Installs the *build-dependencies* (for a package already in Debian)  
Or `mk-build-deps -ir` (inside the package source tree)
- ▶ `debuild`: construcción, comprobación con `lintian`, firma con GPG
- ▶ También se puede invocar `dpkg-buildpackage` directamente
  - ▶ Habitualmente con `dpkg-buildpackage -us -uc`
- ▶ Se recomienda construir paquetes en un entorno mínimo y limpio
  - ▶ `pbuilder` – Asistente de construcción de paquetes en una «*jaula*» *chroot*  
Buena documentación: <https://wiki.ubuntu.com/PbuilderHowto>  
(optimización: `cowbuilder ccache distcc`)
  - ▶ `schroot` y `sbuild`: Utilizados por los servicios de construcción de Debian  
(no es tan sencillo como `pbuilder`, pero es compatible con datos LVM  
Consulte: <https://help.ubuntu.com/community/SbuildLVMHowto>)

# Instalar y comprobar paquetes

- ▶ Instalación local de paquete: **debi** (emplea `.changes` para saber qué instalar)
- ▶ Muestra el contenido del paquete: **debc** `../mi-paquete<TAB>.changes`
- ▶ Compare el paquete con una versión anterior:  
**debdiff** `../mi-paquete_1_*.changes ../mi-paquete_2_*.changes`  
o para comparar las fuentes:  
**debdiff** `../mi-paquete_1_*.dsc ../mi-paquete_2_*.dsc`
- ▶ Compruebe el paquete con **lintian** (análisis estático):  
**lintian** `../mi-paquete<TAB>.changes`  
**lintian -i**: Ofrece más información de fallos
- ▶ Envíe el paquete a (**dput**) (requiere configuración)
- ▶ Administre un archivo Debian privado con **reprepro**  
Documentación: <http://mirrorer.alioth.debian.org/>



# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 **Ejercicio práctico 1: modificar el paquete grep**
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUjump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos



# Ejercicio práctico 1: modificar el paquete grep

- ➊ Go to <http://ftp.debian.org/debian/pool/main/g/grep/> and download version 2.6.3-3 of the package (if you use Ubuntu 11.10 or later, or Debian testing or unstable, use version 2.9-1 or 2.9-2 instead)
  - ▶ If the source package is not unpacked automatically, unpack it with `dpkg-source-x grep_*.dsc`
- ➋ Consulte los ficheros en `debian/`.
  - ▶ ¿Cuántos paquetes binarios genera este paquete fuente?
  - ▶ ¿Qué asistente de creación de paquetes utiliza este paquete?
- ➌ Construya el paquete
- ➍ A continuación, modificaremos el paquete. Añada una entrada al registro de cambios (fichero «changelog») e incremente el número de versión.
- ➎ Desactive la compatibilidad con las expresiones regulares de Perl (`perl-regexp` es una opción de configuración de `./configure`)
- ➏ Reconstruya el paquete
- ➐ Compare el paquete original y el nuevo con `debdiff`
- ➑ Instale el paquete recién construido
- ➒ Llore si provoca problemas ;)



# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes**
- 6 Desarrollar paquetes en Debian
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUjump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos





# debian/copyright

- ▶ Información de derechos de autor y licencia de las fuentes y la tarea de creación del paquete
- ▶ Habitualmente, se escribe como fichero de texto
- ▶ Nuevo formato legible por máquina:

<http://www.debian.org/doc/packaging-manuals/copyright-format/1.0/>

---

```
Format: http://www.debian.org/doc/packaging-manuals/copyright-format/1
Upstream-Name: X Solitaire
Source: ftp://ftp.example.com/pub/games
```

```
Files: *
Copyright: Copyright 1998 John Doe <jdoe@example.com>
License: GPL-2+
This program is free software; you can redistribute it
[...]
```

```
.
On Debian systems, the full text of the GNU General Public
License version 2 can be found in the file
'usr/share/common-licenses/GPL-2'.
```

```
Files: debian/*
Copyright: Copyright 1998 Jane Smith <jsmith@example.net>
License:
[LICENSE TEXT]
```



# Modificar las fuentes del desarrollador original

Habitualmente es necesario:

- ▶ Arreglar informes de fallo o añadir modificaciones específicas para Debian
- ▶ Adaptar a una versión anterior los arreglos de una publicación del software más reciente

Existen varios métodos:

- ▶ Modificación directa de ficheros
  - ▶ Sencillo
  - ▶ No existe forma de registrar y documentar los cambios
- ▶ Utilizar sistemas de parches
  - ▶ Facilita contribuir sus cambios al desarrollador original
  - ▶ Ayuda a compartir los arreglos con distribuciones derivadas
  - ▶ Ofrece una mayor visibilidad de los cambios

<http://patch-tracker.debian.org/>



# Sistemas de parches

- ▶ Principio: los cambios se guardan en parches en `debian/patches/`
- ▶ Se integran y eliminan de las fuentes durante la construcción
- ▶ Pasado: varias implementaciones – *simple-patchsys* (*cdb*s), *dpatch*, ***quilt***
  - ▶ Cada uno permite dos tareas de `debian/rules`:
    - ▶ `debian/rules patch`: Integra todos los parches
    - ▶ `debian/rules unpatch`: Elimina todos los parches de las fuentes
  - ▶ Más documentación: <http://wiki.debian.org/debian/patches>
- ▶ **Nuevo formato de paquete fuente con sistema de parches integrado: 3.0 (quilt)**
  - ▶ Solución recomendada
  - ▶ Debe leer *quilt*  
<http://pkg-perl.alioth.debian.org/howto/quilt.html>
  - ▶ Patch-system-agnostic tool in `devscripts`: `edit-patch`



# Documentación de parches

- ▶ Cabeceras estándar al principio del parche
- ▶ Documentado con las normas de etiquetado de parches; DEP-3 - Patch Tagging Guidelines  
<http://dep.debian.net/deps/dep3/>

---

```
Description: Fix widget frobnication speeds
Frobnicating widgets too quickly tended to cause explosions.
Forwarded: http://lists.example.com/2010/03/1234.html
Author: John Doe <johndoe-guest@users.alioth.debian.org>
Applied-Upstream: 1.2, http://bazaar.foo.com/frobnicator/revision/123
Last-Update: 2010-03-29
```

```
--- a/src/widgets.c
+++ b/src/widgets.c
@@ -101,9 +101,6 @@ struct {
```



# Realizar acciones durante la instalación y eliminación

- ▶ A veces no basta con descomprimir el paquete
- ▶ Crear/eliminar usuarios del sistema, iniciar/detener servicios, gestionar el sistema de *alternativas*
- ▶ Se realiza mediante *scripts de desarrollador*  
preinst, postinst, prerm, postrm
  - ▶ debhelper puede generar secciones de código para acciones comunes
- ▶ Documentación:
  - ▶ Capítulo 6 de Normas de Debian  
<http://www.debian.org/doc/debian-policy/ch-maintainerscripts.html>
  - ▶ Capítulo 6.4 de Referencia del Desarrollador de Debian (Debian Developer's Reference)  
<http://www.debian.org/doc/developers-reference/best-pkging-practices.html>
  - ▶ <http://people.debian.org/~srivasta/MaintainerScripts.html>
- ▶ Consultar al usuario
  - ▶ Se debe realizar mediante **debconf**



# Supervisar las versiones del desarrollador original

- Especifique dónde mirar en `debian/watch` (consulte `uscan(1)`)

```
version=3
```

```
http://tmrc.mit.edu/mirror/twisted/Twisted/(\d\.\d)/ \
Twisted-([\d\.]*)\.tar\.bz2
```

- La infraestructura de Debian utiliza `debian/watch`:

## **Debian External Health Status**

<http://dehs.alioth.debian.org/>

- Se notifica al responsable del paquete mediante correos electrónicos al sistema de seguimiento de paquetes («Package Tracking System»)

<http://packages.qa.debian.org/>

- `uscan`: Ejecuta una comprobación manual
- `uupdate`: Intenta actualizar el paquete a la última versión de la fuente original



# Packaging with a Version Control System

- ▶ Existen varias herramientas que facilitan gestionar las ramas y etiquetas para las tareas de creación de paquete:  
`svn-buildpackage`, `git-buildpackage`
- ▶ Ejemplo: `git-buildpackage`
  - ▶ La rama `upstream` contiene los cambios de la fuente original de software mediante etiquetas `upstream/versión`
  - ▶ La rama `master` contiene los cambios hechos al paquete Debian
  - ▶ Etiquetas `debian/versión` para cada envío de datos
  - ▶ La rama `pristine-tar` para poder reconstruir el archivo tar de la fuente de software original
- ▶ Campos `Vcs-*` en `debian/control` para ubicar el repositorio
  - ▶ `http://wiki.debian.org/Alioth/Git`
  - ▶ `http://wiki.debian.org/Alioth/Svn`

`Vcs-Browser: http://git.debian.org/?p=devscripts/devscripts.git`

`Vcs-Git: git://git.debian.org/devscripts/devscripts.git`

`Vcs-Browser: http://svn.debian.org/viewsvn/pkg-perl/trunk/libwww-perl/`

`Vcs-Svn: svn://svn.debian.org/pkg-perl/trunk/libwww-perl`

- ▶ Interfaz independiente del sistema de control de versiones: `debcheckout`,



# Backporting packages

- ▶ Goal: use a newer version of a package on an older system  
e.g use *mutt* from Debian *unstable* on Debian *stable*
- ▶ General idea:
  - ▶ Take the source package from Debian unstable
  - ▶ Modify it so that it builds and works fine on Debian stable
    - ▶ Sometimes trivial (no changes needed)
    - ▶ Sometimes difficult
    - ▶ Sometimes impossible (many unavailable dependencies)
- ▶ Some backports are provided and supported by the Debian project  
<http://backports.debian.org/>





# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian**
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUjump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos



# Hay varias formas de contribuir a Debian

---

► **La peor** forma de contribuir:

- ❶ Empaquetar su propio programa
- ❷ Introducirlo en Debian
- ❸ Desaparecer

► **Las mejores** formas de contribuir:

- Únase a equipos de creación de paquetes
  - Hay varios equipos que se centran en un conjunto de paquetes, y necesitan ayuda
  - Puede consultar la lista en <http://wiki.debian.org/Teams>
  - Una excelente forma de aprender de otros contribuyentes experimentados
- Adopte paquetes existentes sin responsable, (*paquetes huérfanos*)
- Traiga software nuevo a Debian
  - Por favor, solo si es suficientemente interesante y útil
  - ¿Hay alternativas ya empaquetadas para Debian?



# Adopción de paquetes huérfanos

- ▶ Existen varios paquetes sin responsable en Debian
- ▶ Lista completa y proceso: <http://www.debian.org/devel/wnpp/>
- ▶ Instalados en su sistema: `wnpp-alert`
- ▶ Diferentes estados:
  - ▶ **Orphaned** (huérfano): el paquete no tiene responsable  
Adóptelo sin problemas
  - ▶ **RFA: Request For Adopter**  
El responsable busca alguien que lo adopte, pero continua trabajando en él  
Adóptelo sin problemas. Se recomienda enviar un correo electrónico al responsable actual.
  - ▶ **¡ITA: Intent To Adopt**  
Alguien intenta adoptar el paquete  
¡Puede ofrecer su ayuda!
  - ▶ **RFH: Request For Help**  
El responsable busca ayuda
- ▶ No se detectan algunos paquetes sin desarrollador → aún no están huérfanos



# Adopting a package: example

```
From: You <you@yourdomain>  
To: 640454@bugs.debian.org, control@bugs.debian.org  
Cc: Francois Marier <francois@debian.org>  
Subject: ITA: verbiste -- French conjugator
```

```
retitle 640454 ITA: verbiste -- French conjugator  
owner 640454 !  
thanks
```

Hi,

I am using verbiste and I am willing to take care of the package.

Cheers,

You

- ▶ Polite to contact the previous maintainer (especially if the package was RFAed, not orphaned)
- ▶ Very good idea to contact the upstream project



# Introducir su paquete en Debian

- ▶ No precisa ningún rol oficial para introducir su paquete en Debian
  - 1 Prepare un paquete fuente
  - 2 Encuentre un desarrollador oficial de Debian que patrocine su paquete
- ▶ Rol oficial (cuando ya tiene experiencia):
  - ▶ **Debian Maintainer (DM):**  
Permiso para enviar sus propios paquetes  
Consulte <http://wiki.debian.org/DebianMaintainer>
  - ▶ **Debian Developer (DD):**  
Miembros del proyecto Debian; pueden votar y enviar cualquier paquete



# ¿Dónde encontrar ayuda?

Ayuda necesaria:

- ▶ Consejos y respuestas a sus preguntas, revisiones de código
- ▶ Apoyo y supervisión para sus envíos de paquete, una vez que está listo

Puede conseguir ayuda de:

- ▶ **Otros miembros del equipo de creación de paquetes**
  - ▶ Conocen los detalles específicos de su paquete
  - ▶ Puede convertirse en un miembro del equipo
- ▶ El grupo Debian Mentors (mentores de Debian, si su paquete no encuentra acomodo en ningún equipo)
  - ▶ <http://wiki.debian.org/DebianMentorsFaq>
  - ▶ Lista de correo: [debian-mentors@lists.debian.org](mailto:debian-mentors@lists.debian.org)  
(otra buena forma de aprender es a través de los problemas)
  - ▶ IRC: `#debian-mentors` en [irc.debian.org](http://irc.debian.org)
  - ▶ <http://mentors.debian.net/>



# Documentación oficial

- ▶ Rincón de los desarrolladores de Debian  
<http://www.debian.org/devel/index.es.html>  
Enlaces a varios recursos para el desarrollo en Debian
- ▶ Guía del Nuevo Mantenedor de Debian  
<http://www.debian.org/doc/manuals/maint-guide/index.es.html>  
Una pequeña introducción al empaquetado en Debian, un poco obsoleto
- ▶ Referencia del desarrollador de Debian  
<http://www.debian.org/doc/developers-reference/>  
Básicamente trata procedimientos de Debian, pero también las mejores prácticas de empaquetado (sección 6)
- ▶ Manual de Normas de Debian  
<http://www.debian.org/doc/debian-policy/>
  - ▶ Todos los requisitos que cada paquete debe cumplir
  - ▶ Normas especiales para Perl, Java, Python, ...
- ▶ Guía de empaquetado de Ubuntu  
<https://wiki.ubuntu.com/PackagingGuide>



# Interfaces para desarrolladores de Debian

---

- ▶ **Centrado en el paquete fuente:** Sistema de seguimiento de paquetes (PTS)  
`http://packages.qa.debian.org/dpkg`
- ▶ **Enfocado a desarrolladores y equipos:** Vista general de paquetes de desarrollador (DDPO)  
`http://qa.debian.org/developer.php?login=pkg-ruby-extras-maintainers@lists.alioth.debian.org`





## ¿Más interesado en Ubuntu?

- ▶ En general, Ubuntu gestiona las diferencias con respecto a Debian
- ▶ No hay un enfoque en paquetes específicos  
En su lugar, se colabora con equipos de Debian
- ▶ Habitualmente, recomienda enviar nuevos paquetes primero a Debian  
<https://wiki.ubuntu.com/UbuntuDevelopment/NewPackages>
- ▶ Un mejor plan probablemente sería:
  - ▶ Participar en un equipo de Debian y actuar como enlace con Ubuntu
  - ▶ Ayuda a reducir las diferencias, evalúe los informes de fallo en Launchpad
  - ▶ Muchas herramientas de Debian le pueden ayudar:
    - ▶ La columna de Ubuntu en la vista general de paquetes para desarrolladores de Debian
    - ▶ El recuadro de Ubuntu en el sistema de seguimiento de paquetes (PTS)
    - ▶ Reciba correo electrónico de informes de fallo de Launchpad a través del PTS



# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian
- 7 Conclusión**
- 8 Ejercicio práctico 2: empaquetar GNUjump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos



# Conclusión

- ▶ Ahora tiene una idea general completa de la creación de paquetes Debian
- ▶ Pero tendrá que leer más documentación
- ▶ Las mejores prácticas se desarrollan con los años
  - ▶ Si no está seguro, utilice el asistente de creación de paquetes **dh**, y el formato **3.0 (quilt)**
- ▶ Asuntos que no se han tratado en esta guía:
  - ▶ UCF – Gestionar los cambios de usuario de ficheros de configuración al actualizar
  - ▶ dpkg triggers – Agrupa acciones similares de scripts de desarrollador
  - ▶ Organización del desarrollo Debian:
    - ▶ Sistema de seguimiento de fallos («Bug Tracking System», BTS)
    - ▶ Distribuciones: stable, testing, unstable, experimental, security, \*-updates, backports, ...
    - ▶ Debian Blends – Subconjuntos de Debian enfocados a grupos específicos

Opiniones: [lucas@debian.org](mailto:lucas@debian.org)



# Asuntos legales

Derechos de autor ©2011 Lucas Nussbaum – [lucas@debian.org](mailto:lucas@debian.org)

**Este documento es software libre:** puede redistribuirlo y/o modificarlo bajo ambas (a su elección):

- ▶ Los términos de la GNU General Public License como publica la Free Software Foundation, bien la versión 3 de la licencia, o (a su elección) cualquier versión posterior.  
<http://www.gnu.org/licenses/gpl.html>
- ▶ Los términos de la Creative Commons Attribution-ShareAlike 3.0 Unported License.  
<http://creativecommons.org/licenses/by-sa/3.0/>



# Última versión y código fuente

- ▶ Última versión :

`http://git.debian.org/?p=collab-maint/packaging-tutorial.git;a=blob_plain;f=packaging-tutorial.pdf;hb=refs/heads/pdf`

- ▶ Contribuya:

- ▶ `git clone`

- `git://git.debian.org/collab-maint/packaging-tutorial.git`

- ▶ `apt-get source packaging-tutorial`

- ▶ `http://git.debian.org/?p=collab-maint/packaging-tutorial.git`

- ▶ Opiniones: `lucas@debian.org`



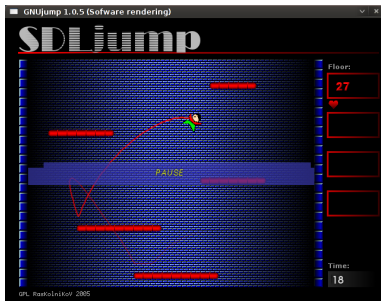
# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUJump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos



## Ejercicio práctico 2: empaquetar GNUjump

- 1 Descargue la versión 1.0.6 de GNUjump desde  
<http://ftp.gnu.org/gnu/gnjump/1.0.6/gnjump-1.0.6.tar.gz>
- 2 Cree un paquete Debian para él
  - ▶ Instale las dependencias de construcción para poder construir el paquete
  - ▶ Obtener un paquete básico funcional
  - ▶ Termine de completar `debian/control` y otros ficheros
- 3 Disfrute



# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUjump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos





# Ejercicio práctico 3: empaquetar una biblioteca de J

- ❶ Consulte brevemente la documentación sobre creación de paquetes de Java:
  - ▶ <http://wiki.debian.org/Java>
  - ▶ <http://wiki.debian.org/Java/Packaging>
  - ▶ <http://www.debian.org/doc/packaging-manuals/java-policy/>
  - ▶ <http://pkg-java.alioth.debian.org/docs/tutorial.html>
  - ▶ Discurso y diapositivas de una conferencia sobre javahelper en Debconf10 :  
<http://pkg-java.alioth.debian.org/docs/debconf10-javahelper-paper.pdf>  
<http://pkg-java.alioth.debian.org/docs/debconf10-javahelper-slides.pdf>
- ❷ Descargue IRCLib desde <http://moepii.sourceforge.net/>
- ❸ Empaquételo



# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUjump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem**
- 11 Respuestas a ejercicios prácticos



# Practical session 4: packaging a Ruby gem

- ❶ Take a quick look at some documentation about Ruby packaging:
  - ▶ <http://wiki.debian.org/Ruby>
  - ▶ <http://wiki.debian.org/Teams/Ruby>
  - ▶ <http://wiki.debian.org/Teams/Ruby/Packaging>
  - ▶ `gem2deb(1)`, `dh_ruby(1)` (in the `gem2deb` package)
- ❷ Create a basic Debian source package from the `net-ssh` gem:  
`gem2deb net-ssh`
- ❸ Improve it so that it becomes a proper Debian package



# Esquema

- 1 Introducción
- 2 Creación de paquetes de fuentes
- 3 Construir y comprobar paquetes
- 4 Ejercicio práctico 1: modificar el paquete grep
- 5 Aspectos avanzados de la creación de paquetes
- 6 Desarrollar paquetes en Debian
- 7 Conclusión
- 8 Ejercicio práctico 2: empaquetar GNUjump
- 9 Ejercicio práctico 3: empaquetar una biblioteca de Java
- 10 Practical session 4: packaging a Ruby gem
- 11 Respuestas a ejercicios prácticos**



# Respuestas a ejercicios prácticos



# Ejercicio práctico 1: modificar el paquete grep

- 1 Go to <http://ftp.debian.org/debian/pool/main/g/grep/> and download version 2.6.3-3 of the package (if you use Ubuntu 11.10 or later, or Debian testing or unstable, use version 2.9-1 or 2.9-2 instead)
- 2 Consulte los ficheros en `debian/`.
  - ▶ ¿Cuántos paquetes binarios genera este paquete fuente?
  - ▶ ¿Qué asistente de creación de paquetes utiliza este paquete?
- 3 Construya el paquete
- 4 A continuación, modificaremos el paquete. Añada una entrada al registro de cambios (fichero «changelog») e incremente el número de versión.
- 5 Desactive la compatibilidad con las expresiones regulares de Perl (`perl-regexp` es una opción de configuración de `./configure`)
- 6 Reconstruya el paquete
- 7 Compare el paquete original y el nuevo con `debdiff`
- 8 Instale el paquete recién construido
- 9 Llore si provoca problemas ;)



# Obtener las fuentes

- ➊ Visite <http://ftp.debian.org/debian/pool/main/g/grep/> y descargue la versión 2.6.3-3 del paquete
- ▶ Utilice dget para descargar el fichero .dsc:  
`dget http://cdn.debian.net/debian/pool/main/g/grep/grep_2.6.3-3.dsc`
- ▶ According to <http://packages.qa.debian.org/grep>, grep version 2.6.3-3 is currently in *stable (squeeze)*. If you have deb-src lines for *squeeze* in your /etc/apt/sources.list, you can use:  
`apt-get source grep=2.6.3-3`  
or `apt-get source grep/stable`  
or, if you feel lucky: `apt-get source grep`
- ▶ El paquete fuente de grep se compone de 3 ficheros:
  - ▶ `grep_2.6.3-3.dsc`
  - ▶ `grep_2.6.3-3.debian.tar.bz2`
  - ▶ `grep_2.6.3.orig.tar.bz2`Esto es típico con el formato «3.0 (quilt)».
- ▶ Si es necesario, descomprima las fuentes con  
`dpkg-source -x grep_2.6.3-3.dsc`



# Explorar y construir el paquete

## 2 Consulte los ficheros en `debian/`.

- ▶ ¿Cuántos paquetes binarios genera este paquete fuente?
- ▶ ¿Qué asistente de creación de paquetes utiliza este paquete?
- ▶ De acuerdo a `debian/control`, este paquete genera un solo paquete binario, llamado `grep`.
- ▶ De acuerdo a `debian/rules`, este paquete es típico del asistente *clásico* `debhelper`, sin utilizar *CDBS* o *dh*. Se pueden ver las múltiples invocaciones a órdenes `dh_*` en `debian/rules`.

## 3 Construya el paquete

- ▶ Utilice `apt-get build-dep grep` para obtener las dependencias de construcción
- ▶ A continuación, ejecute `debuild` o `dpkg-buildpackage -us -uc` (Llevará en torno a 1 minuto)





# Editar el registro de cambios

- ④ A continuación, modificaremos el paquete. Añada una entrada al registro de cambios (fichero «changelog») e incremente el número de versión.
  - ▶ `debian/changelog` es un fichero de texto. Puede editarlo y añadir una entrada nueva manualmente.
  - ▶ O puede utilizar `dch -i`, que añadirá una entrada y ejecutará el editor
  - ▶ El nombre y correo electrónico se puede definir con las variables de entorno `DEBFULLNAME` y `DEBEMAIL`
  - ▶ A continuación, reconstruya el paquete: una nueva versión del paquete es generada
  - ▶ El versionado de paquetes se detalla en la sección 5.6.12 de las Normas de Debian  
<http://www.debian.org/doc/debian-policy/ch-controlfields.html>



# Desactivar la compatibilidad con expresiones regulares

- 5 Desactive la compatibilidad con las expresiones regulares de Perl (perl-regexp es una opción de configuración de ./configure)
- 6 Reconstruya el paquete
  - ▶ Para comprobar, utilice ./configure --help: la opción para desactivar la compatibilidad con expresiones regulares de Perl es --disable-perl-regexp
  - ▶ Edite debian/rules y busque la línea con ./configure
  - ▶ Añada --disable-perl-regexp
  - ▶ Reconstruya el paquete con debuild o dpkg-buildpackage -us -uc



# Comparar y comprobar los paquetes

- 7 Compare el paquete original y el nuevo con debdiff
- 8 Instale el paquete recién construido
  - ▶ Compare los paquetes binarios: `debdiff ../changes`
  - ▶ Compare los paquetes fuente: `debdiff ../dsc`
  - ▶ Instale el paquete recién creado: `debinstall dpkg -i ../grep_  
O dpkg -i ../grep_  
▶ dpkg -P foo ya no funciona!`
- 9 Llore si provoca problemas ;)

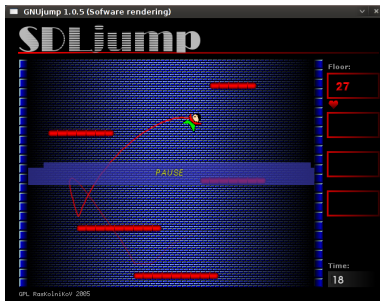
O no: reinstale la versión anterior del paquete

- ▶ `apt-get install --reinstall grep=2.6.3-3` (= *versión anterior*)



## Ejercicio práctico 2: empaquetar GNUjump

- 1 Descargue la versión 1.0.6 de GNUjump desde  
<http://ftp.gnu.org/gnu/gnujump/1.0.6/gnujump-1.0.6.tar.gz>
- 2 Cree un paquete Debian para él
  - ▶ Instale las dependencias de construcción para poder construir el paquete
  - ▶ Obtener un paquete básico funcional
  - ▶ Termine de completar `debian/control` y otros ficheros
- 3 Disfrute



## Paso a paso...

- ▶ `wget http://ftp.gnu.org/gnu/gnujump/1.0.6/gnujump-1.0.6.tar.gz`
- ▶ `mv gnujump-1.0.6.tar.gz gnujump_1.0.6.orig.tar.gz`
- ▶ `tar xf gnujump_1.0.6.orig.tar.gz`
- ▶ `cd gnujump-1.0.6/`
- ▶ `dh_make`
  - ▶ Tipo de paquete: binario único (por ahora)

```
gnujump-1.0.6$ ls debian/
changelog          gnujump.default.ex  preinst.ex
compat            gnujump.doc-base.EX prerm.ex
control           init.d.ex           README.Debian
copyright         manpage.1.ex       README.source
docs              manpage.sgml.ex    rules
emacsen-install.ex manpage.xml.ex      source
emacsen-remove.ex  menu.ex            watch.ex
emacsen-startup.ex postinst.ex
gnujump.cron.d.ex  postrm.ex
```



## Step by step... (2)

- ▶ Revise `debian/changelog`, `debian/rules`, `debian/control` (completado automáticamente por **dh\_make**)
- ▶ En `debian/control`:  
Build-Depends: `debhelper (>= 7.0.50 )`, `autotools-dev`  
Enumera las *dependencias de construcción* = paquetes necesarios para construir el paquete
- ▶ Intente construir el paquete en su estado actual (gracias a la «magia» de **dh**)
  - ▶ Añada dependencias de construcción hasta que se puede construir
  - ▶ Pista: utilice `apt-cache search` y `apt-file` para encontrar los paquetes
  - ▶ Ejemplo:

```
checking for sdl-config... no
checking for SDL - version >= 1.2.0... no
[...]
configure: error: *** SDL version 1.2.0 not found!
```

→ Añada **libsdl1.2-dev** al campo «Build-Depends» e instálelo.
  - ▶ Mejor aún: utilice **pbuilder** para realizar la construcción en un entorno limpio



## Step by step... (3)

- ▶ Tras instalar `libSDL1.2-dev`, `libSDL-image1.2-dev`, `libSDL-mixer1.2-dev`, el paquete se construye correctamente.
- ▶ Utilice `debc` para mostrar el contenido del paquete generado.
- ▶ Utilice `debi` para instalar y comprobar el paquete.
- ▶ Complete `debian/control` utilizando <http://www.debian.org/doc/debian-policy/ch-controlfields.html>
- ▶ Pruebe el paquete con `lintian`
- ▶ Elimine los ficheros que no necesita en `debian/`
- ▶ Compare su paquete con el que existe en Debian:
  - ▶ Separa los ficheros de datos en un segundo paquete, que es idéntico en todas las arquitecturas (→ ahorra espacio en el archivo de Debian)
  - ▶ Instala un fichero «.desktop» (para los menús de GNOME/KDE) y también se integra en el menú de Debian
  - ▶ Arregla algunos problemas pequeños utilizando parches



# Ejercicio práctico 3: empaquetar una biblioteca de J

- ❶ Consulte brevemente la documentación sobre creación de paquetes de Java:
  - ▶ <http://wiki.debian.org/Java>
  - ▶ <http://wiki.debian.org/Java/Packaging>
  - ▶ <http://www.debian.org/doc/packaging-manuals/java-policy/>
  - ▶ <http://pkg-java.alioth.debian.org/docs/tutorial.html>
  - ▶ Discurso y diapositivas de una conferencia sobre javahelper en Debconf10 :  
<http://pkg-java.alioth.debian.org/docs/debconf10-javahelper-paper.pdf>  
<http://pkg-java.alioth.debian.org/docs/debconf10-javahelper-slides.pdf>
- ❷ Descargue IRCLib desde <http://moepii.sourceforge.net/>
- ❸ Empaquételo





## Paso a paso...

- ▶ `apt-get install javahelper`
- ▶ Cree un paquete fuente básico: `jh_makepkg`
  - ▶ Biblioteca
  - ▶ Ninguno
  - ▶ Compilador y sistema de tiempo de ejecución libre predefinido
- ▶ Compruébelo y arregle `debian/*`
- ▶ `dpkg-buildpackage -us -uc OO debuild`
- ▶ `lintian`, `debc`, etc.
- ▶ Compare sus resultados con el paquete fuente `libirclib-java`



# Practical session 4: packaging a Ruby gem

- 1 Take a quick look at some documentation about Ruby packaging:
  - ▶ <http://wiki.debian.org/Ruby>
  - ▶ <http://wiki.debian.org/Teams/Ruby>
  - ▶ <http://wiki.debian.org/Teams/Ruby/Packaging>
  - ▶ `gem2deb(1)`, `dh_ruby(1)` (in the `gem2deb` package)
- 2 Create a basic Debian source package from the `net-ssh` gem:  
`gem2deb net-ssh`
- 3 Improve it so that it becomes a proper Debian package



## Paso a paso...

`gem2deb net-ssh:`

- ▶ Downloads the gem from `rubygems.org`
- ▶ Creates a suitable `.orig.tar.gz` archive, and untar it
- ▶ Initializes a Debian source package based on the gem's metadata
  - ▶ Named `ruby-gemname`
- ▶ Tries to build the Debian binary package (this might fail)

`dh_ruby` (included in `gem2deb`) does the Ruby-specific tasks:

- ▶ Build C extensions for each Ruby version
- ▶ Copy files to their destination directory
- ▶ Update shebangs in executable scripts
- ▶ Run tests defined in `debian/ruby-tests.rb` or `debian/ruby-test-files.yaml`, as well as various other checks



## Step by step... (2)

Improve the generated package:

- ▶ Run `debclean` to clean the source tree. Look at `debian/`.
- ▶ `changelog` and `compat` should be correct
- ▶ Edit `debian/control`: uncomment Homepage, improve Description
- ▶ Write a proper `copyright` file based on the upstream files
- ▶ `ruby-net-ssh.docs`: install `README.rdoc`
- ▶ `ruby-tests.rb`: run the tests. In that case, it is enough to do:  

```
$: << 'test' << 'lib' << '.'  
require 'test/test_all.rb'
```



## Step by step... (3)

Build the package. It fails to build. There are two problems:

- ▶ You need to disable the `gem` call in the test suite.  
In `test/common.rb`, remove the `gem "test-unit"` line:
  - ▶ `edit-patch disable-gem.patch`
  - ▶ Edit `test/common.rb`, remove the `gem` line. Exit the sub-shell
  - ▶ Describe the changes in `debian/changelog`
  - ▶ Document the patch in `debian/patches/disable-gem.patch`
- ▶ The package lacks a build-dependency on `ruby-mocha`, which is used by the test suite (you might need to build your package in a clean environment, using `pbuilder`, to reproduce that problem)
  - ▶ Add `ruby-mocha` to the package's Build-Depends
  - ▶ *gem2deb* copies the dependencies documented in the *gem* as comments in `debian/control`, but *mocha* is not listed as a development dependency by the *gem* (that's a bug in the *gem*)

Compare your package with the `ruby-net-ssh` package in the Debian archive



# Traducción

Omar Campagne Polaino

Si encuentra algún error de traducción en la documentación, envíe un correo a `<debian-l10n-spanish@lists.debian.org>`.

