

# Guía de creación de paquetes Debian

Lucas Nussbaum

`packaging-tutorial@packages.debian.org`

version 0.30 – 2024-03-16



# Acerca de esta guía

- ▶ **Objetivo: ofrecer el conocimiento esencial para la creación de paquetes de 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 fuente
- 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 Conclusions
- 8 Additional practical sessions
- 9 Respuestas a ejercicios prácticos



# Esquema

- 1 Introducción
- 2 Creación de paquetes fuente
- 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 Conclusions
- 8 Additional practical sessions
- 9 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 forma 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

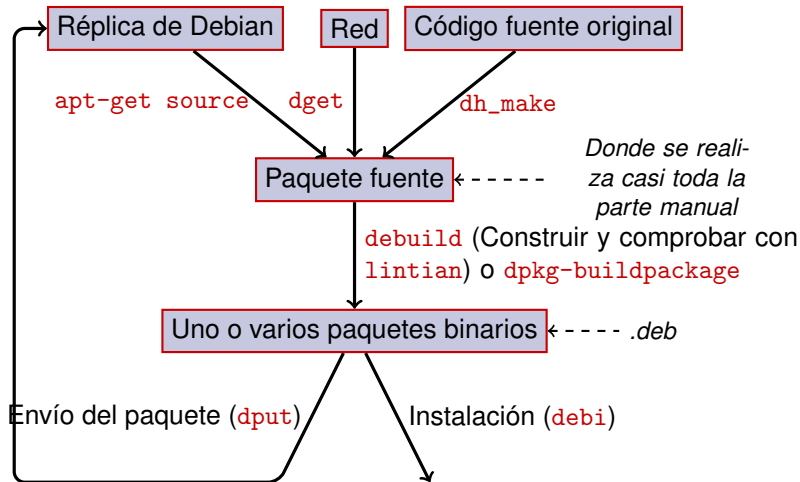
- ▶ Un sistema Debian (o Ubuntu) con acceso de usuario «root»
- ▶ Algunos paquetes:
  - ▶ **build-essential**: has dependencies on the packages that will be assumed to be available on the developer's machine (no need to specify them in the `Build-Depends`: control field of your package)
    - ▶ también 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`, **cdbs**, **quilt**, **pbuilder**, **sbuild**, **lintian**, **svn-buildpackage**, **git-buildpackage**, ...  
Instálelos a medida que los necesite





# Etapas generales en la creación de paquetes



## Ejemplo: reconstruir dash

- 1 **Install packages needed to build dash, and devscripts**  
`sudo apt-get build-dep dash`  
(requires `deb-src` lines in `/etc/apt/sources.list`)  
`sudo apt-get install --no-install-recommends devscripts fakeroot`
- 2 **Cree un directorio de trabajo y entre:**  
`mkdir /tmp/debian-tutorial ; cd /tmp/debian-tutorial`
- 3 **Obtenga el paquete de fuentes de dash**  
`apt-get source dash`  
(Requiere las líneas `deb-src` en `/etc/apt/sources.list`)
- 4 **Construya el paquete**  
`cd dash-*`  
`debuild -us -uc` (-us -uc desactiva el firmado de paquetes con GPG)
- 5 **Compruebe el funcionamiento**
  - ▶ Hay algunos ficheros `.deb` nuevos en el directorio superior
- 6 **Compruebe el directorio `debian/`**
  - ▶ Aquí se realizan las tareas de empaquetado



# Esquema

- 1 Introducción
- 2 Creación de paquetes fuente
- 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 Conclusions
- 8 Additional practical sessions
- 9 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 or 3.0 (native): `package_version.tar.gz`
  - ▶ 1.0 (non-native):
    - ▶ `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 Kothe <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 se 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
- ▶ Rename the directory to `<source_package>-<upstream_version>` (example: `simgrid-3.6`)
- ▶ `cd <source_package>-<upstream_version> && dh_make` (from the **dh-make** package)
- ▶ 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** – Registro histórico del paquete de 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

- ▶ Edición manual o con `dch`
  - ▶ Cree una entrada en el fichero «changelog» para una nueva publicación: `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, ...
- ▶ Documentation: Debian Policy chapter 5  
<https://www.debian.org/doc/debian-policy/ch-controlfields>

---

```
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](http://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
- ▶ Documented in Debian Policy, chapter 4.8  
<https://www.debian.org/doc/debian-policy/ch-source#s-debianrules>
- ▶ Required targets:
  - ▶ `build`, `build-arch`, `build-indep`: 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`
- ▶ 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_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`, ...
- ▶ Otros asistentes para conjuntos específicos de paquetes: **python-support**, **dh\_ocaml**, ...
- ▶ `debian/compat`: Debhelper compatibility version



## 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/package
    $(MAKE) DESTDIR=$(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
  - ▶ E.g. building with `./configure && make && make install` or CMake
- ▶ CDBS:
  - ▶ Introducido en 2005, basado en «magia» avanzada de *GNU make*
  - ▶ Documentación: `/usr/share/doc/cdb/`
  - ▶ 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/cdb/1/rules/debhelper.mk
include /usr/share/cdb/1/class/autotools.mk
```

```
# add an option after the build
```





# 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 sustituciones
- ▶ 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

- ▶ Mind shares:  
Classic debhelper: 15%   CDBS: 15%   dh: 68%
- ▶ ¿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)
  - ▶ See <https://trends.debian.net/#build-systems>



# Esquema

- 1 Introducción
- 2 Creación de paquetes fuente
- 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 Conclusions
- 8 Additional practical sessions
- 9 Respuestas a ejercicios prácticos



# Construir paquetes

- ▶ `apt-get build-dep mypackage`

Instala los *build-dependencies* (for a package already in Debian)

Or `mk-build-deps -ir` (for a package not uploaded yet)

- ▶ `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 del 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`
- ▶ Check the package with `lintian` (static analyzer):  
`lintian` `../mypackage<TAB>.changes`  
`lintian -i`: gives more information about the errors  
`lintian -EviIL +pedantic`: shows more problems
- ▶ Envíe el paquete a (`dput`) (requiere configuración)
- ▶ Manage a private Debian archive with `reprepro` or `aptly`  
Documentation:  
<https://wiki.debian.org/HowToSetupADebianRepository>



# Esquema

- 1 Introducción
- 2 Creación de paquetes fuente
- 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 Conclusions
- 8 Additional practical sessions
- 9 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 2.12-2 of the package
  - ▶ Si el paquete no se desempaqueta de forma automática, utilice `dpkg-source-x grep_*.dsc`
- 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



# Esquema

- 1 Introducción
- 2 Creación de paquetes fuente
- 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 Conclusions
- 8 Additional practical sessions
- 9 Respuestas a ejercicios prácticos





# debian/copyright

- ▶ Información de derechos de autor, licencia de las fuentes y de la tarea de creación del paquete
- ▶ Habitualmente, se escribe como fichero de texto
- ▶ New machine-readable format:  
<https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/>

---

```
Format: https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/  
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
  - ▶ Pero no ofrece una 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
  - ▶ Gives more exposure to the changes

<http://patch-tracker.debian.org/> (down currently)



# 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 (cdfs)*, *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
  - ▶ More documentation: <https://wiki.debian.org/debian/patches>
- ▶ **Nuevo formato de paquete fuente con sistema de parches integrado: 3.0 (quilt)**
  - ▶ Solución recomendada
  - ▶ You need to learn *quilt*  
<https://perl-team.pages.debian.net/howto/quilt.html>
  - ▶ Herramienta de parches independiente del sistema en `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://bZR.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:
  - ▶ Debian Policy Manual, chapter 6  
<https://www.debian.org/doc/debian-policy/ch-maintainerscripts>
  - ▶ Debian Developer's Reference, chapter 6.4  
<https://www.debian.org/doc/developers-reference/best-pkging-practices.html>
  - ▶ <https://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
```

- ▶ There are automated trackers of new upstream versions, that notify the maintainer on various dashboards including <https://tracker.debian.org/> and <https://udd.debian.org/dmd/>
- ▶ `uscan`: Ejecuta una comprobación manual
- ▶ `uupdate`: Intenta actualizar el paquete a la última versión de la fuente original



# Creación de paquetes con un sistema de control de

- ▶ Existen varias herramientas que facilitan la gestión de 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

Doc: <http://honk.sigxcpu.org/projects/git-buildpackage/manual-html/gbp.html>

- ▶ Campos `Vcs-*` en `debian/control` para ubicar el repositorio
  - ▶ <https://wiki.debian.org/Salsa>

Vcs-Browser: <https://salsa.debian.org/debian/devscripts>

Vcs-Git: <https://salsa.debian.org/debian/devscripts.git>

Vcs-Browser: <https://salsa.debian.org/perl-team/modules/packages/libwww-perl>

Vcs-Git: <https://salsa.debian.org/perl-team/modules/packages/libwww-perl.git>

# Adaptación de paquetes a una publicación anterior

- ▶ Objetivo: utilizar una versión más reciente de un paquete en un sistema más antiguo  
Por ejemplo, utilizar *mutt* de la publicación *unstable* («inestable») de Debian en la publicación *stable* («estable»)
- ▶ Idea general:
  - ▶ Obtenga el paquete fuente de Debian «inestable»
  - ▶ Modifique de forma que se construya y funcione de forma adecuada en la publicación estable de Debian
    - ▶ A veces trivial (no se requieren cambios)
    - ▶ A veces difícil
    - ▶ A veces imposible (muchas dependencias no disponibles)
- ▶ El proyecto Debian proporciona y mantiene algunas adaptaciones a publicaciones anteriores  
<http://backports.debian.org/>



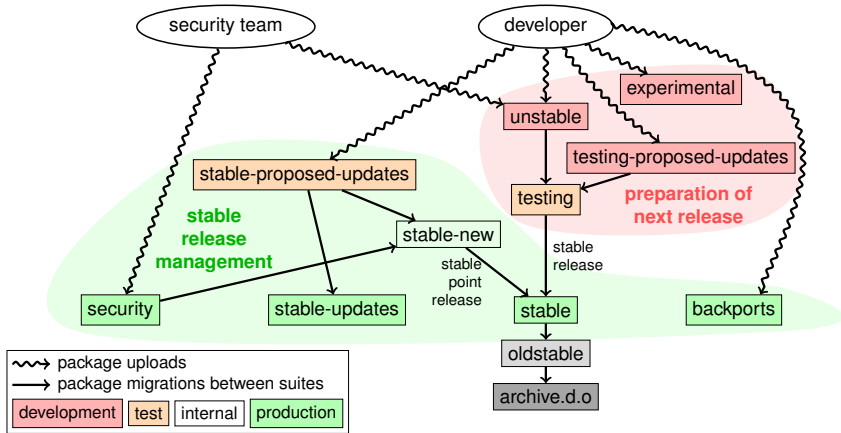


# Esquema

- 1 Introducción
- 2 Creación de paquetes fuente
- 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 Conclusions
- 8 Additional practical sessions
- 9 Respuestas a ejercicios prácticos



# Debian archive and suites



Based on graph by Antoine Beaupré. <https://salsa.debian.org/debian/package-cycle>



# Suites for development

- ▶ New versions of packages are uploaded to **unstable** (**sid**)
- ▶ Packages migrate from **unstable** to **testing** based on several criterias (e.g. has been in unstable for 10 days, and no regressions)
- ▶ New packages can also be uploaded to:
  - ▶ **experimental** (for more *experimental* packages, such as when the new version is not ready to replace the one currently in unstable)
  - ▶ **testing-proposed-updates**, to update the version in **testing** without going through **unstable** (this is rarely used)



# Freezing and releasing

- ▶ At some point during the release cycle, the release team decides to *freeze* testing: automatic migrations from **unstable** to **testing** are stopped, and replaced by manual review
- ▶ When the release team considers **testing** to be ready for release:
  - ▶ The **testing** suite becomes the new **stable** suite
  - ▶ Similarly, the old **stable** becomes **oldstable**
  - ▶ Unsupported releases are moved to `archive.debian.org`
- ▶ See `https://release.debian.org/`



# Stable release suites and management

- ▶ Several suites are used to provide stable release packages:
  - ▶ **stable**: the main suite
  - ▶ **security** updates suite provided on `security.debian.org`, used by the security team. Updates are announced on the `debian-security-announce` mailing list
  - ▶ **stable-updates**: updates that are not security related, but that should urgently be installed (without waiting for the next point release): antivirus databases, timezone-related packages, etc. Announced on the `debian-stable-announce` mailing list
  - ▶ **backports**: new upstream versions, based on the version in **testing**
- ▶ The **stable** suite is updated every few months by *stable point releases* (that include only bug fixes)
  - ▶ Packages targetting the next stable point release are uploaded to **stable-proposed-updates** and reviewed by the release team
- ▶ The **oldstable** release has the same set of suites



# Hay varias formas de contribuir a Debian

---

▶ **La peor** forma de contribuir:

- 1 Empaquetar su propio programa
- 2 Introducirlo en Debian
- 3 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
  - ▶ List available at <https://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
- ▶ Full list + process: <https://www.debian.org/devel/wnpp/>
- ▶ Installed on your machine: `wnpp-alert`  
Or better: `how-can-i-help`
- ▶ 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



# Adopción de un paquete: ejemplo

```
From: Usted <usted@su-dominio>  
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,

Su nombre

- ▶ Se recomienda contactar con el responsable anterior (en particular si el paquete se declaró como RFA, petición de adopción, en lugar de declararse huérfano)
- ▶ Se recomienda contactar con la fuente original del proyecto





# Introducir su paquete en Debian

- ▶ No precisa de ningún rol oficial para introducir su paquete en Debian
  - ➊ Submit an **ITP** bug (**I**ntent **T**o **P**ackage) using `reportbug wnpp`
  - ➋ Prepare un paquete fuente
  - ➌ Encuentre un desarrollador oficial de Debian que patrocine su paquete
  
- ▶ Official status (when you are an experienced package maintainer):
  - ▶ **Debian Maintainer (DM):**  
Permission to upload your own packages  
See <https://wiki.debian.org/DebianMaintainer>
  - ▶ **Debian Developer (DD):**  
Debian project member; can vote and upload any package



# Things to check before asking for sponsorship

- ▶ Debian puts **a lot of focus on quality**
- ▶ Generally, **sponsors are hard to find and busy**
  - ▶ Make sure your package is ready before asking for sponsorship
- ▶ Things to check:
  - ▶ Avoid missing build-dependencies: make sure that your package build fine in a clean *sid chroot*
    - ▶ Using `pbuilder` is recommended
  - ▶ Run `lintian -EviIL +pedantic` on your package
    - ▶ Errors must be fixed, all other problems should be fixed
  - ▶ Do extensive testing of your package, of course
- ▶ In doubt, ask for help



# ¿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**
  - ▶ List of teams: <https://wiki.debian.org/Teams>
- ▶ The **Debian Mentors group** (if your package does not fit in a team)
  - ▶ <https://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](irc://irc.debian.org)
  - ▶ <http://mentors.debian.net/>
  - ▶ Documentation: <http://mentors.debian.net/intro-maintainers>
- ▶ **Localized mailing lists** (get help in your language)
  - ▶ [debian-devel-{french,italian,portuguese,spanish}@lists.d.o](mailto:debian-devel-{french,italian,portuguese,spanish}@lists.d.o)
  - ▶ Full list: <https://lists.debian.org/devel.html>
  - ▶ Or users lists: <https://lists.debian.org/users.html>



## More documentation

- ▶ Debian Developers' Corner  
<https://www.debian.org/devel/>  
Links to many resources about Debian development
- ▶ Guide for Debian Maintainers  
<https://www.debian.org/doc/manuals/debmake-doc/>
- ▶ Debian Developer's Reference  
<https://www.debian.org/doc/developers-reference/>  
Mostly about Debian procedures, but also some best packaging practices (part 6)
- ▶ Debian Policy  
<https://www.debian.org/doc/debian-policy/>
  - ▶ Todos los requisitos que cada paquete debe cumplir
  - ▶ Normas especiales para Perl, Java, Python, ...
- ▶ Ubuntu Packaging Guide  
<https://packaging.ubuntu.com/html/>



# Interfaces para desarrolladores de Debian

- ▶ **Source package centric:**  
<https://tracker.debian.org/dpkg>
- ▶ **Maintainer/team centric:** Developer's Packages Overview (DDPO)  
<https://qa.debian.org/developer.php?login=pkg-ruby-extras-maintainers@lists.alioth.debian.org>
- ▶ **TODO-list oriented:** Debian Maintainer Dashboard (DMD)  
<https://udd.debian.org/dmd/>



# Using the Debian Bug Tracking System (BTS)

- ▶ A quite unique way to manage bugs
  - ▶ Web interface to view bugs
  - ▶ Email interface to make changes to bugs
- ▶ Adding information to bugs:
  - ▶ Write to `123456@bugs.debian.org` (does not include the submitter, you need to add `123456-submit@bugs.debian.org`)
- ▶ Changing bug status:
  - ▶ Send commands to `control@bugs.debian.org`
  - ▶ Command-line interface: `bts` command in `devscripts`
  - ▶ Documentation: <https://www.debian.org/Bugs/server-control>
- ▶ Reporting bugs: use `reportbug`
  - ▶ Normally used with a local mail server: install `ssmtp` or `nullmailer`
  - ▶ Or use `reportbug --template`, then send (manually) to `submit@bugs.debian.org`



# Using the BTS: examples

- ▶ Sending an email to the bug and the submitter:  
`https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#10`
- ▶ Tagging and changing the severity:  
`https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680227#10`
- ▶ Reassigning, changing the severity, retitling ...:  
`https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=680822#93`
  - ▶ `notfound`, `found`, `notfixed`, `fixed` are for **version-tracking**  
See `https://wiki.debian.org/HowtoUseBTS#Version_tracking`
- ▶ Using usertags: `https://bugs.debian.org/cgi-bin/bugreport.cgi?msg=42;bug=642267`  
See `https://wiki.debian.org/bugs.debian.org/usertags`
- ▶ BTS Documentation:
  - ▶ `https://www.debian.org/Bugs/`
  - ▶ `https://wiki.debian.org/HowtoUseBTS`



## ¿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, recomiende 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
  - ▶ Ayude 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 fuente
- 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 **Conclusions**
- 8 Additional practical sessions
- 9 Respuestas a ejercicios prácticos



# Conclusions

- ▶ Ahora tiene una idea general completa de la creación de paquetes Debian
- ▶ Pero tendrá que leer más documentación
- ▶ Las mejores prácticas se desarrollan con el tiempo
  - ▶ Si no está seguro, utilice el asistente de creación de paquetes **dh**, y el formato **3.0 (quilt)**

Feedback: **[packaging-tutorial@packages.debian.org](mailto:packaging-tutorial@packages.debian.org)**



# Asuntos legales

Copyright ©2011–2019 Lucas Nussbaum – 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/>



# Contribute to this tutorial

- ▶ Contribuya:
  - ▶ `apt-get source packaging-tutorial`
  - ▶ `debcheckout packaging-tutorial`
  - ▶ `git clone`  
`https://salsa.debian.org/debian/packaging-tutorial.git`
  - ▶ `https://salsa.debian.org/debian/packaging-tutorial`
  - ▶ Open bugs: `bugs.debian.org/src:packaging-tutorial`
- ▶ Provide feedback:
  - ▶ `mailto:packaging-tutorial@packages.debian.org`
    - ▶ What should be added to this tutorial?
    - ▶ What should be improved?
  - ▶ `reportbug packaging-tutorial`



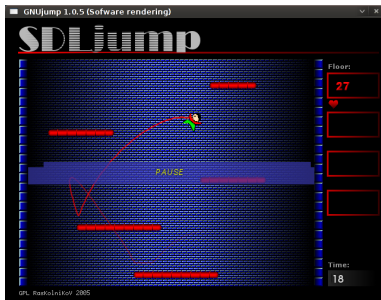
# Esquema

- 1 Introducción
- 2 Creación de paquetes fuente
- 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 Conclusiones
- 8 Additional practical sessions**
- 9 Respuestas a ejercicios prácticos



## Ejercicio práctico 2: empaquetar GNUjump

- 1 Download GNUjump 1.0.8 from <http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.tar.gz>
- 2 Cree un paquete Debian para él
  - ▶ Instale las dependencias de construcción para poder construir el paquete
  - ▶ Fix bugs
  - ▶ Obtener un paquete básico funcional
  - ▶ Termine de completar `debian/control` y otros ficheros
- 3 Disfrute



## Practical session 2: packaging GNUjump (tips)

- ▶ To get a basic working package, use `dh_make`
- ▶ To start with, creating a *1.0* source package is easier than *3.0* (*quilt*) (change that in `debian/source/format`)
- ▶ To search for missing build-dependencies, find a missing file, and use `apt-file` to find the missing package
- ▶ If you encounter that error:

```
/usr/bin/ld: SDL_rotozoom.o: undefined reference to symbol 'ceil@@GLIBC_2.2.5'  
//lib/x86_64-linux-gnu/libm.so.6: error adding symbols: DSO missing from command line  
collect2: error: ld returned 1 exit status  
Makefile:376: recipe for target 'gnujump' failed
```

You need to add `-lm` to the linker command line:

Edit `src/Makefile.am` and replace

```
gnujump_LDFLAGS = $(all_libraries)
```

by

```
gnujump_LDFLAGS = -Wl,--as-needed  
gnujump_LDADD = $(all_libraries) -lm
```

Then run `autoreconf -i`



# Ejercicio práctico 3: empaquetar una biblioteca de J

---

- 1 Consulte brevemente la documentación sobre creación de paquetes de Java:
  - ▶ <https://wiki.debian.org/Java>
  - ▶ <https://wiki.debian.org/Java/Packaging>
  - ▶ <https://www.debian.org/doc/packaging-manuals/java-policy/>
  - ▶ [/usr/share/doc/javahelper/tutorial.txt.gz](#)
- 2 Descargue IRCLib desde <http://moepii.sourceforge.net/>
- 3 Empaquételo





## Ejercicio práctico 4: empaquetar un «gem» de Ruby

---

- 1 Consulte brevemente la documentación sobre creación de paquetes de Ruby:
  - ▶ <https://wiki.debian.org/Ruby>
  - ▶ <https://wiki.debian.org/Teams/Ruby>
  - ▶ <https://wiki.debian.org/Teams/Ruby/Packaging>
  - ▶ `gem2deb(1)`, `dh_ruby(1)` (en el paquete `gem2deb`)
- 2 Create a basic Debian source package from the peach gem:  
`gem2deb peach`
- 3 Modifíquelo para que sea un paquete de Debian adecuado



# Practical session 5: packaging a Perl module

- 1 Take a quick look at some documentation about Perl packaging:
  - ▶ <https://perl-team.pages.debian.net>
  - ▶ <https://wiki.debian.org/Teams/DebianPerlGroup>
  - ▶ `dh-make-perl(1)`, `dpt(1)` (in the `pkg-perl-tools` package)
- 2 Create a basic Debian source package from the `Acme` CPAN distribution:  
`dh-make-perl --cpan Acme`
- 3 Modifíquelo para que sea un paquete de Debian adecuado



# Esquema

- 1 Introducción
- 2 Creación de paquetes fuente
- 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 Conclusions
- 8 Additional practical sessions
- 9 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 2.12-2 of the package
- 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



# Obtener las fuentes

- ➊ Go to <http://ftp.debian.org/debian/pool/main/g/grep/> and download version 2.12-2 of the package
- ▶ Use `dget` to download the `.dsc` file:  
`dget http://cdn.debian.net/debian/pool/main/g/grep/grep_2.12-2.dsc`
- ▶ If you have `deb-src` for a Debian release that has `grep` version 2.12-2 (find out on <https://tracker.debian.org/grep>), you can use: `apt-get source grep=2.12-2`  
Or `apt-get source grep/release` (e.g. `grep/stable`)  
or, if you feel lucky: `apt-get source grep`
- ▶ El paquete fuente de `grep` se compone de 3 ficheros:
  - ▶ `grep_2.12-2.dsc`
  - ▶ `grep_2.12-2.debian.tar.bz2`
  - ▶ `grep_2.12.orig.tar.bz2`

Esto es típico con el formato «3.0 (quilt)».

- ▶ If needed, uncompress the source with  
`dpkg-source -x grep_2.12-2.dsc`



# Explorar y construir el paquete

## 2 Look at the files in `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
  - ▶ Package versioning is detailed in section 5.6.12 of the Debian policy <https://www.debian.org/doc/debian-policy/ch-controlfields>





# 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: `debi`  
`o dpkg -i ../grep_  
<TAB>`
  - ▶ `!grep -P foo` ya no funciona!

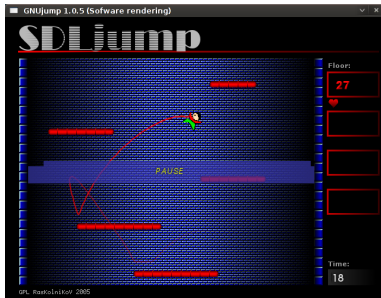
Reinstall the previous version of the package:

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



## Ejercicio práctico 2: empaquetar GNUjump

- 1 Download GNUjump 1.0.8 from  
<http://ftp.gnu.org/gnu/gnujump/gnujump-1.0.8.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/gnujump-1.0.8.tar.gz`
- ▶ `mv gnujump-1.0.8.tar.gz gnujump_1.0.8.orig.tar.gz`
- ▶ `tar xf gnujump_1.0.8.orig.tar.gz`
- ▶ `cd gnujump-1.0.8/`
- ▶ `dh_make -f ../gnujump-1.0.8.tar.gz`
  - ▶ Tipo de paquete: binario único (por ahora)

```
gnujump-1.0.8$ 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
```



## Paso a paso... (2)

- ▶ Examine `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
- ▶ Try to build the package as-is with `debuild` (thanks to **dh** magic)
  - ▶ 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



## Paso a paso... (3)

- ▶ Required build-dependencies are `libsdl1.2-dev`, `libsdl-image1.2-dev`, `libsdl-mixer1.2-dev`
- ▶ Then, you will probably run into another error:

```
/usr/bin/ld: SDL_rotozoom.o: undefined reference to symbol 'ceil@@GLIBC_2.2.5'  
//lib/x86_64-linux-gnu/libm.so.6: error adding symbols: DSO missing from command line  
collect2: error: ld returned 1 exit status  
Makefile:376: recipe for target 'gnujump' failed
```

- ▶ This problem is caused by bitrot: `gnujump` has not been adjusted following linker changes.
- ▶ If you are using source format version **1.0**, you can directly change upstream sources.

- ▶ Edit `src/Makefile.am` and replace

```
gnujump_LDFLAGS = $(all_libraries)
```

by

```
gnujump_LDFLAGS = -Wl,--as-needed  
gnujump_LDADD = $(all_libraries) -lm
```

- ▶ Then run `autoreconf -i`



## Step by step... (4)

- ▶ If you are using source format version **3.0 (quilt)**, use quilt to prepare a patch. (see <https://wiki.debian.org/UsingQuilt>)

- ▶ `export QUILT_PATCHES=debian/patches`

- ▶ `mkdir debian/patches`

- `quilt new linker-fixes.patch`

- `quilt add src/Makefile.am`

- ▶ Edit `src/Makefile.am` and replace

- `gnujump_LDFLAGS = $(all_libraries)`

- by

- `gnujump_LDFLAGS = -Wl,--as-needed`

- `gnujump_LDADD = $(all_libraries) -lm`

- ▶ `quilt refresh`

- ▶ Since `src/Makefile.am` was changed, `autoreconf` must be called during the build. To do that automatically with `dh`, change the `dh` call in `debian/rules` from:

- `dh $ --with autotools-dev`

- `to: dh $ --with autotools-dev --with autoreconf`



## Step by step... (5)

- ▶ The package should now build fine.
- ▶ Use `debnc` to list the content of the generated package, and `debi` to install it and test it.
- ▶ Pruebe el paquete con `lintian`
  - ▶ While not a strict requirement, it is recommended that packages uploaded to Debian are *lintian-clean*
  - ▶ More problems can be listed using `lintian -EviIL +pedantic`
  - ▶ Some hints:
    - ▶ Elimine los ficheros que no necesita en `debian/`
    - ▶ Fill in `debian/control`
    - ▶ Install the executable to `/usr/games` by overriding `dh_auto_configure`
    - ▶ Use *hardening* compiler flags to increase security. See <https://wiki.debian.org/Hardening>





## Step by step... (6)

- ▶ 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

---

- 1 Consulte brevemente la documentación sobre creación de paquetes de Java:
  - ▶ <https://wiki.debian.org/Java>
  - ▶ <https://wiki.debian.org/Java/Packaging>
  - ▶ <https://www.debian.org/doc/packaging-manuals/java-policy/>
  - ▶ [/usr/share/doc/javahelper/tutorial.txt.gz](#)
- 2 Descargue IRCLib desde <http://moepii.sourceforge.net/>
- 3 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
- ▶ Compruebe y modifique `debian/*`
- ▶ `dpkg-buildpackage -us -uc 0 debuild`
- ▶ `lintian`, `debc`, etc.
- ▶ Compare sus resultados con el paquete fuente `libirclib-java`



## Ejercicio práctico 4: empaquetar un «gem» de Ruby

---

- 1 Consulte brevemente la documentación sobre creación de paquetes de Ruby:
  - ▶ <https://wiki.debian.org/Ruby>
  - ▶ <https://wiki.debian.org/Teams/Ruby>
  - ▶ <https://wiki.debian.org/Teams/Ruby/Packaging>
  - ▶ `gem2deb(1)`, `dh_ruby(1)` (en el paquete `gem2deb`)
- 2 Create a basic Debian source package from the peach gem:  
`gem2deb peach`
- 3 Modifíquelo para que sea un paquete de Debian adecuado



## Paso a paso...

gem2deb peach:

- ▶ Descarga el fichero «gem» de [rubygems.org](http://rubygems.org)
- ▶ Genera un archivo «.orig.tar.gz» adecuado, y descomprime el archivo tar
- ▶ Inicia una paquete fuente de Debian en base o los metadatos del «gem»
  - ▶ Se denomina `ruby-gemname`
- ▶ Intenta construir un paquete binario de Debian (puede fallar)

dh\_ruby (incluido en *gem2deb*) realiza las tareas específicas de Ruby:

- ▶ Genera extensiones C para cada versión de Ruby
- ▶ Copia ficheros a su directorio de destino
- ▶ Actualiza los «shebang» de los scripts ejecutables
- ▶ Run tests defined in `debian/ruby-tests.rb`, `debian/ruby-tests.rake`, or `debian/ruby-test-files.yaml`, as well as various other checks



## Paso a paso... (2)

Mejore el paquete generado:

- ▶ Ejecute `debclean` para limpiar el árbol de fuentes. Compruebe `debian/`.
- ▶ `changelog` y `compat` deben ser correctas
- ▶ Edit `debian/control`: `improve Description`
- ▶ Cree un fichero `copyright` adecuado basado en los ficheros del desarrollador original
- ▶ Construya el paquete
- ▶ Compare your package with the `ruby-peach` package in the Debian archive



# Practical session 5: packaging a Perl module

- 1 Take a quick look at some documentation about Perl packaging:
  - ▶ <https://perl-team.pages.debian.net>
  - ▶ <https://wiki.debian.org/Teams/DebianPerlGroup>
  - ▶ `dh-make-perl(1)`, `dpt(1)` (in the `pkg-perl-tools` package)
- 2 Create a basic Debian source package from the `Acme` CPAN distribution:  
`dh-make-perl --cpan Acme`
- 3 Modifíquelo para que sea un paquete de Debian adecuado



## Paso a paso...

`dh-make-perl --cpan Acme:`

- ▶ Downloads the tarball from the CPAN
- ▶ Creates a suitable `.orig.tar.gz` archive, and untars it
- ▶ Initializes a Debian source package based on the distribution's metadata
  - ▶ Named `libdistname-perl`





## Paso a paso... (2)

Mejore el paquete generado:

- ▶ `debian/changelog`, `debian/compat`, `debian/libacme-perl.docs`, and `debian/watch` should be correct
- ▶ Edit `debian/control`: improve `Description`, and remove boilerplate at the bottom
- ▶ Edit `debian/copyright`: remove boilerplate paragraph at the top, add years of copyright to the `Files: *` stanza



# Traducción

Omar Campagne Polaino

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

