# Guía del Nuevo Desarrollador de Debian

Josip Rodin <jrodin@jagor.srce.hr>
Traducido por: Javier Fernández–Sanguino Peña <jfs@computer.org>

versión 0.9a.19, 25 de enero de 2000

## Copyright Notice

Copyright © 1998, 1999 Josip Rodin. Translation Copyright © 1999, 2000 Javier Fernández–Sanguino Peña.

Este documento puede ser usado en los términos descritos en la Licencia Pública GNU versión 2 o posterior.

Con permiso de los autores, se han usado u modificado porciones de los siguientes documentos:

Making a Debian Package (AKA the Debmake Manual), copyright © 1997 Jaldhar Vyas `<jaldhar@debian.org>`.

The New–Maintainer's Debian Packaging Howto, copyright © 1997 Will Lowe `<lowe@debian.org>`.

# Contents


**1 Empezando “de la Forma Correcta” 1**

1.1 Programas que necesitas para el desarrollo . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.2 Más información . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

**2 Primeros pasos 5**

2.1 Elija su programa. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.2 Obtenga el programa, y pruebelo. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.3 Cosas antes de ‘dh_make’ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2.4 Ejecutando ‘dh_make’ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

**3 Modificando las fuentes. 9**

3.1 Instalación en un subdirectorio . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

3.2 Librerías diferentes. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

**4 Las cosas necesarias bajo debian/ 13**

4.1 El fichero ‘control’. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

4.2 el fichero ‘copyright’ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

4.3 el fichero ‘changelog’ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

4.4 el fichero ‘rules’. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19

**5 Otros ficheros en el directorio debian/. 27**

5.1 README.debian (LEEME.debian, n. del t.) . . . . . . . . . . . . . . . . . . . . . . . . . 27

5.2 conffiles . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27

5.3 dirs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

5.4 manpage.1.ex . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

5.5 menu.ex . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
5.6 watch.ex . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
5.7 ex.doc–base . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
5.8 postinst.ex, preinst.ex, postrm.ex, prerm.ex . . . . . . . . . . . . . . . . . . . . . . . . 30

**6 Pasos Finales 31**

6.1 Construir el paquete. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
6.2 Comprobar su paquete para encontrar errores. . . . . . . . . . . . . . . . . . . . . . . . 32
6.3 Enviando su paquete. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
6.4 Actualizar su paquete . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
6.5 Dónde pedir ayuda. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

# Chapter 1

# Empezando “de la Forma Correcta”

Este documento tratará de describir cómo se construye un paquete Debian GNU/Linux para el usuario normal de Debian (y futuros desarrolladores) en un lenguaje informal, y con multitud de ejemplos. Hay un antiguo dicho romano que dice, *¡Longum iter est per preaecepta, breve et efficax per exempla!* (¡Es un largo camino con las reglas, pero corto y eficiente con ejemplos!)

Una de las cosas que hace a Debian una de las distribuciones más importantes del mercado es su sistema de paquetes. Aunque hay una gran cantidad de programas disponibles en formato de paquetes de Debian, algunas veces necesitará instalar programas que no estén así. Puede preguntarse cómo hará usted sus propios paquetes y que quizás esta sea una tarea demasiada difícil. Bueno, si es usted un novato en Linux, sí es duro, pero si eres todo un experto, no deberías estar leyendo esto ahora mismo. :–) Necesitas saber algo sobre programación en Unix, pero desde luego no tienes que ser un maestro.

Se pueden obtener versiones nuevas de este documento en línea en `http://www.debian.org/doc/maint-guide` y en el paquete ’maint–guide–es’.

## 1.1 Programas que necesitas para el desarrollo

Antes de empezar nada, deberías asegurarte que tienes instalados algunos paquetes adicionales necesarios para el desarrollo. Observe que en la lista no están incluidos paquetes ’esenciales’ o ’necesarios’ que se suponen ya instalados.

Este documento ha sido escrito cuando la distribución ’slink’ 2.1 era las distribución estable oficial de Debian, y ’potato’ 2.2 estaba en fase de desarrollo, por ello los paquetes que se indicarán aquí son principalmente aquellos de ’potato’.

Los siguientes paquetes vienen en una instalación estándar de Debian 2.1, así que probablemente ya los tenga (junto con los paquetes de los que depende). Aun así, debería comprobarlo con ‘dpkg –s <paquete>e‘.

- *binutils* – estos programas se usan para ensamblar y enlazar ficheros objetos – aquellos que componen los programas (vea ‘info binutils‘)

- *cpp* – el preprocesador de C. (vea `cpp(1)`)
- *cpio* – este es un programa archivador como tar o zip. (vea `cpio(1)`)
- *dpkg–dev* – este paquete contiene las herramientas necesarias para desempaquetar, construir y enviar paquetes fuente de Debian. (vea `dpkg-source(1)`)
- *file* – este útil programa puede determinar de qué tipo es un fichero. (vea `file(1)`)
- *gcc* – el compilador de C de GNU. La mayor parte de los programas de Linux están escritos en C. Si su programa está escrito en algún otro lenguaje de programación como C++, Fortran o Pascal, debería instalar g++, g77, o gpc, respectivamente. (lea `gcc(1)`, `g++(1)`, `g77(1)`, `gpc(1)`)
- *libc6–dev* – las librerías y cabeceras de fichero de C que gcc necesita para enlazar y crear ficheros objeto. Aunque algunos programas recomienda y/o usan libc5, le sugiero que utilice la nueva versión (libc6). (vea ‘info libc‘)
- *make* – habitualmente la creación de un programa tiene varios pasos. En lugar de ejecutar los mismos comandos una y otra vez, puede utilizar este programa para automatizar el proceso, creando ‘Makefiles’. Algunos programas también usan imake y xmkmf, programas para generar Makefiles de un conjunto de funciones macro. Algunos programas más nuevos usan guiones de configuración y Makefiles con la ayuda de programas como autoconf y automake, así que puede necesitarlos también. (vea ‘info make‘, `imake(1)`, `xmkmf(1)`, `autoconf(1)`, `automake(1)`)
- *patch* – esta utilidad es muy útil ya que permite coger el fichero que contiene un listado de diferencias (producido por el programa diff) y aplicárselas al fichero original, produciendo una versión “parcheada”. (vea `patch(1)`)
- *perl5* – Perl es uno de los lenguajes interpretados para hacer guiones más usados en los sistemas un*x de hoy, comúnmente referido como la “Sierra Mecánica Suiza de Unix”. (vea `perl(1)`)

De la sección ’devel’ de la distribución posiblemente necesite instalar esto usted mismo:

- *dh–make* y *debhelper* – dh–make es necesario para crear el esqueleto de nuestro paquete ejemplo, y se usarán algunas de las herramientas de debhelper para crear paquetes. No son imprescindibles para la creación de paquetes, pero se recomienda **encarecidamente** para nuevos desarrolladores. Hace el proceso mucho más fácil al principio, y más fácil de controlar también más adelante. (vea `dh_make(1)`, `debhelper(1)`, /usr/share/doc/debhelper/README)
- *devscripts* – este paquete contiene algunos guiones útiles para los desarrolladores, pero no son necesarios para crear paquetes. (vea /usr/share/doc/devscripts/README.gz)
- *fakeroot* o *libtricks* – estos le permiten emular ser root (superusuario, n. del t.) lo cual es necesario para ciertas partes del proceso de construcción. (vea `fakeroot(1)`)
- *lintian* – este paquete le indica muchos de los errores comunes después de construir un paquete, y explica los errores. Hace falta tener instalado diffstat también, una pequeña utilidad que crea histogramas a partir de la salida de diff. (vea `lintian(1)`, `diffstat(1)`, /usr/share/doc/lintian/lintian.html/index.htm

Por último, estos paquetes de *gran importancia* están en la sección ’doc’ de la distribución:

- *debian–policy* – incluye la estructura y contenidos del archivo, ciertas notas sobre diseño del SO, el Estándar de la Jerarquía de Ficheros de Linux (Filesystem Hierarchy Standard, n. del t), y, lo más importante (para usted) es que describe los requisitos que debe satisfacer cada paquete para ser incluido en la distribución. (vea /usr/share/doc/debian–policy/policy.html/index.html)
- *developers–reference* – para todos los temas no específicamente relacionados con los detalles técnicos de cómo empaquetar, como la estructura del archivo, como renombrar, abandonar, coger paquetes, cómo hacer NMUs (Non–Maintainer Uploads, o envíos por personas distintos del desarrollador, n. del t.), como gestionar los bugs (errores, n. del t.) y cómo y cuando enviar etc. (lea /usr/share/doc/developers–reference/developers–reference.html/index.html)
- *packaging–manual* – describe los aspectos técnicos de cómo crear paquetes binarios y fuentes en Debian. (lea /usr/share/doc/packaging–manual/packaging.html/index.html)

También necesitará un paquete criptográfico, o bien PGP (los paquetes pgp–*) o GPG (el paquete gpg), para *firmar* digitalmente su paquete. Esto es especialmente importante si quiere distribuir su paquete a otras personas (y hará precisamente esto cuando si su paquete se incluye en la distribución de Debian). Sin embargo, debido a las leyes de exportación de los EEUU, no puede simplemente obtenerlo de su servidor de FTP de Debian más cercano. Pero Debian ofrece estos paquetes a través de un servidor que no está físicamente en EEUU, llamado non–US.debian.org (ftp://non–us.debian.org/debian–non–US/). Su servidor de FTP tendrá un fichero llamado README.non–us, que le indicará dónde está la réplica más cercana de este servidor.

Las breves descripciones dadas anteriormente sólo sirven para introducirle a lo que hace cada paquete. Antes de continuar, por favor lea la documentación de cada programa, al menos para su uso normal. Puede parecerle algo duro ahora, pero más adelante estará *muy* contento de haberla leído. Nota: *debmake* es una paquete que incluye otros programas con funciones similares a dh–make, pero su uso específico **no** está cubierto en este documento. Lea el manual de Debmake (`http://www.debian.org/~jaldhar/`) para más información.

## 1.2 Más información

Usted puede construir dos tipos de paquetes, fuentes y binarios. Un paquete fuente contiene el código que puede compilar en un programa. Un paquete binario contiene sólo el programa terminado. ¡No mezcle los términos como fuentes de un programa y el paquete fuente de un programa! Por favor lea los otros manuales si necesita más detalles sobre terminología.

Debian usa el término ’desarrollador’ para la persona que hace paquetes, ’autor fuente’ (’upstream author, n. del t.) para la persona que hizo el programa, y ’desarrollador fuente’ (’upstream maintainer’, n. del t.) para la persona que actualmente mantiene el programa fuera de Debian. Generalmente el autor y el desarrollador fuente son la misma persona – y algunas veces incluso el desarrollador es el mismo. Si hace usted un programa, y quiere incluirlo en Debian, tiene total libertad para solicitar convertirse en desarrollador.

Después de construir su paquete (o mientras lo hace), deberá convertirse en un desarrollador oficial de Debian si desea que su programa entre en la próxima distribución (si el programa es útil, ¿por qué no?). Este proceso se explica en la Referencia del Desarrollador, por favor, leala.

# Chapter 2

# Primeros pasos

Mientras que la documentación en el Rincón del Desarrollador (`http://www.debian.org/devel/`) no está tan clara sobre dónde y cómo debería empezar un nuevo desarrollador empezar su trabajo, este documento explicara cada pequeño (y algunas veces incluso irrelevante) detalle, para ayudarle crear ese primer paquete, y ganar alguna experiencia en la construcción de nuevas versiones de éste y quizás otros paquetes más adelante.

## 2.1 Elija su programa.

Probablemente haya escogido ya el paquete que desea construir, pero aquí hay algunos punteros para los novatos:

- compruebe si el paquete ya está en la distribución. Si usa la distribución 'estable', quizás sea mejor que vaya a la página de búsqueda de paquetes (`http://www.debian.org/distrib/packages.html`). Si usa la distribución 'inestable' **actual**, compruebe esto con los comandos:

```
dpkg -s programa
dpkg -l '*programa*'
```

- consulte la página WNPP (`http://www.debian.org/doc/prospective-packages.html`) y la lista de correo de debian–devel para ver si alguna otra persona está construyendo ese mismo programa. Si es así, contacte el desarrollador actual si creo que lo necesita mantener. Sino, intente buscar otro programa interesante que nadie mantenga.
- el programa **debe** tener una licencia, si es posible, libre en el sentido marcado por las Guías del Software Libre de Debian (`http://www.debian.org/social_contract.html#guidelines`). Si no sigue una de estas reglas, aún puede incluirse en las secciones 'contrib' o 'non–free' de Debian. Si no está seguro sobre en qué lugar debería ir, pregunte en `<debian-legal@lists.debian.org>`.
- el programa **no** debería ejecutarse con setuid root, o aún mejor: no debería estar 'setuid' ni 'setgid' a nada.

- el programa no debería ser un demonio, o algo que va en los directorios */sbin.
- el programa debería ser binarios ejecutables, no intente librerías aún.
- debería estar bien documentado, o al menos legible (para cualquiera)
- debería contactar el autor/es del programa para comprobar si están de acuerdo con que se empaquete. Esto es importante para consultar al autor/es sobre el programa en caso de que haya problemas específicos, así que no intente empaquetar programas que no estén mantenidos.
- y por último pero no menos importante, debería saber cómo funciona, y haberlo utilizado por algún tiempo.

Por supuesto, esta lista es para tomar medidas de seguridad, y con la intención de salvarle de usuarios enfurecidos si hace algo mal con algún demonio de ftp setuid... Pero cuando tenga más experiencia en empaquetar cosas, podrá hacer este tipo de paquetes, pero incluso los desarrolladores más experimentados preguntan en la lista de distribución de debian–devel cuando tienen dudas. Y la gente allí le ayudara gustosamente.

Para más ayuda sobre esto, lea la Referencia del Desarrollador

## 2.2 Obtenga el programa, y pruebelo.

La primera cosa a hacer es encontrar y descargar el paquete original. Supongo que ya tiene el código fuente que obtuvo de la página del autor. Las fuentes de Linux generalmente vienen en el formato tar/gzip, con extensión `.tar.gz`, y generalmente contienen un subdirectorio llamado programa–versión con todas las fuentes en él. Si su programa viene en otro tipo de archivo (como por ejemplo, el fichero termina con `.Z` o `.zip`), descomprima con las herramientas adecuadas, o pregunte en debian–mentors si no está seguro de cómo se puede desempaquetar correctamente (pista: pruebe 'file archivo.extensión').

Como ejemplo, usaré el programa conocido como 'gentoo', un gestor de ficheros de X11 con GTK+. Observe que el programa ya ha sido empaquetado previamente pero ha cambiado sustancialmente de versión desde que este texto se escribió.

Cree un subdirectorio bajo su directorio personal llamado 'debian' o 'deb' o lo que crea apropiado (por ejemplo ~/gentoo/ estaría bien en este caso) Y mueva a él el archivo que ha descargado, y descomprimalo de la siguiente forma: 'tar –xzf gentoo–0.9.12.tar.gz'. Asegúrese de que no hay errores, incluso errores "irrelevantes", porque es muy probable que haya problemas desempaquetando en sistemas de otras personas, cuyas herramientas de desempaquetamiento puede que ignoren o no estas anomalías.

Ahora tiene otro subdirectorio, llamado 'gentoo–0.9.12'. Muévase a ese directorio y lea **en profundidad** la documentación que encuentre. Está generalmente en ficheros que se llaman README*, INSTALL*, *.lsm o *.html. Allí encontrará instrucciones de cómo compilar e instalar el programa (muy probablemente asumirán que quiere instalar en el directorio /usr/local/bin: no hará esto, pero lo veremos más adelante en 'Instalación en un subdirectorio' on page 9).

El proceso varía de programas a programas, pero gran parte de los programas modernos vienen con un guión 'configure' que configura las fuentes para su sistema y se asegura de que su sistema está en condiciones de compilarlo. Después de configurarlo (con './configure'), los programas generalmente se compilan con 'make'. Algunos de ellos soportan 'make check' para ejecutarse incluyendo auto–comprobaciones. Generalmente se instalarán en sus directorios de destino ejecutando 'make install'.

Ahora intente compilar, y ejecutar el programa, para asegurarse de que funciona bien y que no rompe nada más mientras está instalándose o ejecutándose.

También, generalmente puede escribir 'make uninstall' para eliminar todos los programas instalados, y 'make clean' (o mejor 'make distclean') para limpiar el directorio donde se genera e programa.

## 2.3 Cosas antes de 'dh_make'

Debería empezar a construir su paquete en un directorio de fuentes completamente limpio, o simplemente con las fuentes recién desempaquetadas.

Para construir correctamente el paquete, debe poner el nombre original del programa en letras minúsculas (si no lo está ya), y debería mover el directorio de fuentes a <nombre_de_paquete>–<versión>.

Si el nombre del programa consiste de una o más palabras, contraigalas a una palabra o haga una abreviatura. Por ejemplo, el paquete del programa "el editor para X de Javi" se podría llamar javiedx o jle4x, o lo que decida, siempre y cuando no se exceda de unos límites razonables, como 20 caracteres.

También compruebe la versión exacta del programa (la que se incluye en la versión del paquete) Si el programa no está numerado con versiones del estilo de X.Y.Z, pero con fecha de lanzamiento, es usted libre de utilizar la fecha como número de versión, precedida por "0.0" (sólo por si los desarrolladores originales deciden sacar una versión nueva como 1.0). Así, si la fecha es el 19 de diciembre de 1998, utilizó la abreviatura norteamericana 0.0.19981219 como número de versión. Aún así habrá algunos que ni siquiera estén numerados, en cualquier caso debe contactar con el 'desarrollador fuente' para ver si tienen algún otro sistema de seguimiento de revisiones.

## 2.4 Ejecutando 'dh_make'

Asegúrese que está en el directorio donde están las fuentes del programa, y ejecute lo siguiente:

```
        dh_make -e tu.dirección.de.desarrollador@dirección -f ../gentoo-
0.9.12.tar.gz
```

Por supuesto, cambie la cadena "tu.dirección.de.desarrollador@dirección" por tu dirección de correo electrónico para que se incluya en la entrada del fichero de cambios así como en otros ficheros, y el nombre de fichero de tu archivo fuente original. Lea `dn_make(1)` para más detalles.

Saldrá alguna información. Le preguntará qué tipo de paquete desea crear. Gentoo es un solo paquete de binarios – crea sólo un binario, y, por tanto, sólo un fichero .deb – así que seleccionaremos la primera opción, con la tecla 's'. Compruebe la información que aparece en la pantalla y confirme pulsando la tecla <intro>. Como nuevo desarrollador, está desaconsejado crear paquetes multibinarios, o librerías, como se explicó antes. No es extremadamente difícil, pero sí necesita algunos conocimientos más, así que aquí no describiremos el proceso.

Tenga en cuenta que debería ejecutar dh_make **sólo una vez**, y que no se comportará correctamente si lo hace otra vez en el mismo directorio, ya "debianizado". Esto también significa que usará un método distinto para crear una nueva revisión o una nueva versión de su paquete en el futuro. Lea más sobre esto más adelante en 'Actualizar su paquete' on page 34.

# Chapter 3

# Modificando las fuentes.

Normalmente, los programas se instalan a sí mismos en el subdirectorio /usr/local. Pero los paquetes Debian no pueden utilizar este directorio ya que está reservado para el uso privado del administrador (o de los usuario). Esto significa que tiene que mirar el sistema de construcción de su programa, generalmente empezando por el Makefile. Éste es el guión `make(1)` que usará para automatizar la creación de éste programa. Para más detalles sobre Makefiles, mire en 'el fichero 'rules'.' on page 19.

Tenga en cuenta que no hay espacio aquí para entrar en *todos* los detalles respecto a los arreglos, pero aquí hay algunos de los problemas frecuentes a los que se enfrenta uno.

## 3.1 Instalación en un subdirectorio

La mayor parte de los programas tienen alguna manera de instalarse en la estructura de directorios existente en su sistema, para que los binarios se incluyan en su $PATH, y para que encuentre la documentación y páginas de manual en los lugares habituales. Tiene que asegurarse que ésto lo hace de forma correctamente, pero tiene que hacer que se instale en un directorio temporal que se creará bajo su directorio debian/, generalmente con el nombre `debian/tmp`, desde el que las herramientas de desarrollo construirán un paquete .deb que se pueda instalar. Todo lo que se incluye en este directorio será instalado en el sistema del usuario cuando instale su paquete, la única diferencia es que dpkg instalará los ficheros en el directorio raíz.

Básicamente, debe hacer que el programa se instale en debian/tmp, pero se comporte correctamente cuando se instale en el directorio raíz, esto es cuando se instale desde el paquete .deb. Con programas que utilizan la herramienta GNU autoconf, ésto será bien sencillo, porque dh_make pondrá las opciones necesarias para hacer esto automáticamente, así que podría dejar de leer esta sección. Pero con otros programas, puede ser necesario que usted examine y edite los Makefiles.

Ésta es la parte importante del Makefile de gentoo:

```
# ¿Dónde poner el binario en  'make install'?
BIN     = /usr/local/bin
# ¿Dónde poner los iconos en 'make install'? Nota: si cambia esto,
```

```
# gentoo no encontrará los iconos cuando arranque. Deberá cambiar
# el path de iconos de gentoo (en la ventana de configuración:
# "Paths") para que funcione.
ICONS   = /usr/local/lib/gentoo/
```

Antes de esto debería insertar dos nuevas líneas que dicen:

```
# Editado para Debian GNU/Linux.
DESTDIR =
```

porque el proceso de construcción lo necesita (explicado más tarde, en ‘el fichero ‘rules’.’ on page 19).

Después el Makefile menciona la localización del binario final. Sólo necesita cambiar esto:

```
# ¿Dónde poner el binario en  'make install'?
BIN     = $(DESTDIR)/usr/X11R6/bin
```

¿Pero por qué en este directorio y no en otro? Porque Debian tiene unas reglas definidas de dónde deberían estar instalados los programas. Están especificas en el Linux Filesystem Structure Standard (/usr/share/doc/debian–policy/fhs) [Estándar de Linux de la Estructura de los Sistemas de Ficheros, n. del t.]. Así, deberíamos instalar el binario en /usr/X11R6/bin en lugar de /usr/local/bin, y la página de manual (no existe aquí, pero casi todos los programas tienen una así que haremos una después) en /usr/share/man/man1 en lugar de /usr/local/man/man1.

Después de esto tenemos un situación un poco más complicada. Si cambia la línea a:

```
ICONS   = $(DESTDIR)/usr/share/gentoo/
```

que estará dentro de la política de Debian, deberá editar algunos ficheros de fuentes reales de C. Pero, ¿dónde buscar? Puede probar a encontrarlos usando:

```
grep -n usr/local/lib *.[ch]
```

(en cada subdirectorio que contiene ficheros .c y .h). Grep dirá el nombre del fichero y la línea, cuando encuentra una ocurrencia. Ahora edite esos ficheros y cambie en esas líneas usr/local/lib con usr/share – y ya está. Simplemente reemplace usr/local/lib por su localización, y sea muy cuidadoso para no mezclar el resto del código, si no sabe mucho sobre cómo programas en C. :–)

Después de esto debería encontrar el objetivo ’install’ (busque una línea que comience por ’install:’) y renombre todas las referencias a directorios distintos de los definidos al comienzo del Makefile. Anteriormente el objetivo ’install’ decía:

```
# ----------------------------------------- Installation
```

```
# ¡Debe ser superusuario para hacer esto!
install:        gentoo-target
                install ./gentoo $(BIN)
                install icons $(ICONS)
                install gentoorc-example $(HOME)/.gentoorc
```

Después del cambio dice:

```
# ----------------------------------------- Installation

# You're going to have to be root to do this!
install:        gentoo-target
                install -d $(BIN) $(ICONS) $(DESTDIR)/etc
                install ./gentoo $(BIN)
                install -m644 icons/* $(ICONS)
                install -m644 gentoorc-example $(DESTDIR)/etc/gentoorc
                install -d $(DESTDIR)/usr/doc/gentoo/html
                cp -a docs/* $(DESTDIR)/usr/doc/gentoo/html
```

Un lector atento se dará cuenta de que he cambiado 'gentoo' a 'gentoo–target' en la línea 'install:'. Esto es para arreglar un fallo en el programa. :–)

Siempre que haga cambios que no estén específicamente relacionados con el paquete Debian, asegúrese de que los envía al desarrollador original para que éste los incluya en la próxima revisión del programa. Observe que no tiene que enviar ninguno de los fichero s debian/* al desarrollador original, pero debería hacerlo con cualquier otro parche. Y sea amable haciendo sus cambios no específicos a Debian o Linux (¡ni siquiera a Unix!) antes de enviarlos.

## 3.2 Librerías diferentes.

Hay un problema común: las librerías son generalmente diferentes de plataforma a plataforma. Por ejemplo, un Makefile puede contener una referencia a una librería que no exista en Debian o ni siquiera en Linux. En este caso, se necesita cambiarla a una librería que sí existe en Debian y sirve para el mismo propósito. La mejor forma es comentar esas líneas porque puede que haber otras personas que intenten compilar en diferentes plataformas y éstas les pueden dar algunas pistas de dónde puede estar el problema.

Así, si hay una línea en el Makefile de su programa que dice algo como lo siguiente (y su programa no compila):

```
LIBS = -lcurses -lsomething -lsomethingelse
```

Entonces cambiela a lo siguiente, y funcionará casi con seguridad:

```
LIBS = -lncurses -lsomething -lsomethingelse
#LIBS = -lcurses -lsomething -lsomethingelse
```

# Chapter 4

# Las cosas necesarias bajo debian/

Ahora hay un nuevo subdirectorio bajo el directorio principal del programa ('gentoo–0.9.12'), que se llama 'debian'. Hay bastantes ficheros en este directorio. Empezaremos editando éstos para adaptar el comportamiento del paquete. La parte más importante es modificar los ficheros 'control', 'rules' (reglas, n. del t.), 'changelog', y 'copyright' que se necesitan en todos los paquetes.

## 4.1 El fichero 'control'.

Este fichero contiene varios valores que dpkg y dselect usarán para gestionar el paquete. Aquí está el fichero de control que dh_make crea para nosotros.

```
1 Source: gentoo
  2 Section: unknown
  3 Priority: optional
  4 Maintainer: Josip Rodin <jrodin@jagor.srce.hr>
  5 Standards-Version: 3.0.1
  6
  7 Package: gentoo
  8 Architecture: any
  9 Depends: ${shlibs:Depends}
  9  Depends: ${shlibs:Depends}
  10 Deguiónion: <insertar unos 60 caracteres de descripción>
  11  <inserta una descripción ligera, indentado con espacios>
```

(He añadido los número de línea.)

La línea 3 describe cómo de importante es para el usuario la instalación de éste paquete. La sección y prioridad son actualmente sólo usadas por dselect cuando ordena paquetes y selecciona valores por defecto, y pueden ser modificadas (y generalmente lo serán) por nuestros administradores del FTP. Lea el manual de Política de Debian para una guía de qué valor deben tener estos campos.

Como es un paquete de prioridad normal, lo dejaremos con prioridad 'optional' (opcional, n. del t.).

La línea 4 es el nombre y correo electrónico del desarrollador.

La línea 5 es la versión de los estándares de la Política de Debian que sigue este paquete (dos versiones importantes del paquete debian–policy instalado).

Si se necesita un compilador no estándar o alguna otra herramienta para construir el paquete, debería añadir aquí una línea 'Build–depends' y listar los paquetes necesarios. Para más información sobre ésto lea el Manual de Empaquetamiento (sección 8.7) y la documentación en el paquete 'build–essential'.

La línea 7 es el nombre del paquete binario.

La línea 8 describe la arquitectura de CPU para la que se compiló el paquete. Podemos dejar ésta como 'any' (cualquiera, n. del. t), ya que `dpkg-gencontrol(1)` lo rellenará con el valor apropiado cuando se compile este paquete (lea la Referencia del Desarrollador para una explicación de qué es portar paquetes). Si su paquete es independiente de la arquitectura (por ejemplo, un documento, un programa de shell o un guión en Perl), cambie esto a "all", y lea más adelante 'el fichero 'rules'.' on page 19 sobre cómo usar la regla 'binary–indep' en lugar de 'binary–arch' para construir el paquete.

La línea 9 muestra una de las más poderosas posibilidades del sistema de paquetes de Debian. Los paquetes se pueden relacionar unos con otros de diversas formas. Aparte de 'Depends:' (depende de, n. del t.) otros campos de relación son 'Recommends:' (recomienda, n. del t.), 'Suggests:' (sugiere, n. del t.), 'Pre–depends:' (predepende de, n. del t.), 'Conflicts:' (entra en conflicto con, n. del t.), 'Provides:' (provee, n. del t.), 'Replaces:' (reemplaza a, n. del t.). Ésto es lo que significan:

Esto es lo que generalmente significan:

- Depends:

  No se instalará el programa a menos que los paquetes de los que depende estén instalados. Use esto si su programa no funcionará de ninguna forma (o causará daños) a menos que esté disponible un paquete determinado.

- Recommends:

  Dselect no le permitirá instalar el paquete a menos que los paquetes que recomienda estén instalados. Dpkg y apt, sin embargo, sí le dejará. Use esto para paquetes que no son estrictamente necesarios pero generalmente son usados con su programa.

- Suggests:

  Cuando un usuario instale el programa, dselect le sugerirá instalar cualquier paquete que éste sugiera. Dpkg y apt no hace caso a esto. Utilice esto para paquetes que funcionarán bien con su programa pero no son necesarios en absoluto.

- Pre–Depends:

  Esto es más fuerte que 'Depends'. Dpkg y dselect no instalarán el paquete a menos que el paquete del que pre–dependa esté instalado *y correctamente configurado*. Utilice esto **muy** poco y sólo después de haberlo discutido en la lista de distribución de debian–devel. En resumidas cuentas: no lo utilice en absoluto. :–>

- Conflicts:

  El paquete no será instalado su programa hasta que todos los paquetes con los que entra en conflicto hayan sido eliminados. Utilice esto si su programa no funcionará en absoluto (o causará daños) si un paquete particular están presentes.

- Provides:

  Para algunos tipos determinados de paquetes dónde hay múltiples alternativas se han definido nombres virtuales. Puede obtener la lista completa en el fichero /usr/share/doc/debian–policy/virtual–package–names–list.text.gz. Use esto si su programa provee las funciones de un paquete virtual existente.

- Replaces:

  Use esto si su programa reemplaza ficheros de otro paquete, o reemplaza totalmente otro paquete (generalmente se usa conjuntamente con 'Conflicts':). Se eliminarán los ficheros de los paquetes indicados antes de instalar el suyo.

Todos estos campos tienen una sintaxis uniforme. Tienen una lista de nombres de paquetes separados por comas. Estos nombres de paquetes también puede ser listas de paquetes alternativos, separados por los símbolos de barra vertical | (símbolos tubería). Los campos pueden restringir su aplicabilidad a versiones determinadas de cada paquete nombrado. Esto se hace entre paréntesis para cada nombre de paquete individual; los paréntesis deberían contener una relación de la siguiente lista seguida por un número de versión. Las relaciones permitidas son <<, <=, =, >= y >> estrictamente para anterior, anterior o igual, exactamente igual, posterior o igual o estrictamente posterior, respectivamente.

La última funcionalidad que quiero necesita conocer es $(shlibs:Depends). Ésta se sustituirá automáticamente por `dh_shlibdeps(1)` y rellenada por `dh_gencontrol(1)` por los nombres de cualquier librería compartida, como libc6 o xlib6g, que use su programa, así que no necesita especificar esto usted mismo. Habiendo dicho todo esto, puede dejar la línea 9 exactamente como está ahora.

La línea 10 es donde va la lista de sugerencias. Aquí sólo es 'file', porque gentoo utiliza algunas funciones de este paquete/programa.

La línea 11 es una descripción corta. La mayor parte de los monitores de la gente son de 80 columnas de ancho, así que no debería tener más de 60 caracteres. Cambiaré esto a "A fully GUI configurable GTK+ file manager" ("Una GUI GTK+ a un gestor de ficheros completamente configurable").

La línea 12 es donde va la descripción larga del paquete. Debería ser al menos un párrafo que da más detalles del paquete. La Columna 1 de cada línea debería estar vacía. No puede haber líneas en blanco, pero puede poner un . (punto) en una columna para simularlo. También no debe haber más de una línea en blanco después de la descripción completa.

Aquí está el fichero de control actualizado:

```
1  Source: gentoo
2  Section: x11
3  Priority: optional
4  Maintainer: Josip Rodin <jrodin@jagor.srce.hr>
```

```
Standards-Version: 3.0.1

Package: gentoo
Architecture: any
Depends: ${shlibs:Depends}
Suggests: file
Deguiónion: A fully GUI configurable GTK+ file manager
gentoo is a file manager for Linux written from scratch in pure C. It
uses the GTK+ toolkit for all of its interface needs. gentoo provides
100% GUI configurability; no need to edit config files by hand and re-
start the program. gentoo supports identifying the type of various
files (using extension, regular expressions, or the 'file' command),
and can display files of different types with different colors and icons.
.
gentoo borrows some of its look and feel from the classic Amiga file
manager "Directory OPUS" (written by Jonathan Potter).
```

## 4.2 el fichero 'copyright'

Este fichero contiene la información sobre la licencia y copyright de las fuentes originales del paquete. El formato no está definido en la Política, pero sí sus contenidos (sección 6.5). Dh_make crea un fichero por defecto que es así:

```
This package was debianized by Josip Rodin <jrodin@jagor.srce.hr> on
Wed, 11 Nov 1998 21:02:14 +0100.

It was downloaded from <fill in ftp site>

Upstream Author(s): <put author(s) name and email here>

Copyright:

<Must follow here>
```

(He añadido los número de línea)

Las cosas importantes que se deben añadir a este fichero son el lugar de donde obtuvo el paquete y la nota de copyright y licencia originales. Debe incluir la licencia completa, a menos que sea una licencia común en el mundo del software libre como GNU GPL o LGPL, BSD o la 'Artistic license', basta referirse al fichero apropiado en el directorio /usr/share/common–licenses/ que existe en todo sistema Debian. Gentoo está licenciado bajo la Licencia Pública General GNU, así que cambiaremos el fichero a esto:

```
This package was debianized by Josip Rodin <jrodin@jagor.srce.hr> on
Wed, 11 Nov 1998 21:02:14 +0100.

It was downloaded from: ftp://ftp.obsession.se/gentoo/

Upstream author: Emil Brink <emil@obsession.se>

This software is copyright (c) 1998-99 by Emil Brink, Obsession
Development.

You are free to distribute this software under the terms of
the GNU General Public License.
On Debian systems, the complete text of the GNU General Public
License can be found in /usr/share/common-licenses/GPL file.
```

(He añadido los números de línea) (Nota del T.: la política de Debian actual indica que los documentos aquí citados estén escritos en inglés, al ser el idioma oficial del, proyecto por ello no se traducen en este documento)

## 4.3 el fichero 'changelog'

Este es un fichero requerido, que tiene un formato especial descrito en el Manual de Empaquetamiento (sección 3.2.3). Este formato es usado por dpkg y otros programas para obtener el número de versión, revisión, distribución y urgencia de su paquete.

Para usted es también importante, ya que es bueno tener documentados todos los cambios que haya hecho. Le ayudará a las personas que se descargen su paquete para ver si hay temas pendientes en el paquete que deberían conocer de forma inmediata. Se salvará como '/usr/share/doc/gentoo/changelog.Debian.gz' en el paquete binario.

Dh_make crea uno por defecto, que es algo así:

```
gentoo (0.9.12-1) unstable; urgency=low

 * Initial Release.

-- Josip Rodin <jrodin@jagor.srce.hr> Wed, 11 Nov 1998 21:02:14 +0100

Local variables:
mode: debian-changelog
End:
```

(He añadido los números de línea)

(N. del T.: se traduce el fichero de copyright, dh_make sólo lo ofrece en inglés)

```
1  Éste paquete fue debianizado por Josip Rodin jrodin@jagor.srce.hr el
2  Mie, 11 Nov 1998 21:02:14 +0100.
3
4  Se descargó de  <rellenar el servidor de ftp>
5
6  Autor/es Original/es: <poner el nombre/s de el/los autor/es y
7  dirección de correo electrónico aquí>
8  Copyright:
9
10 <Debe venir aquí>
```

La línea 1 es el nombre del paquete, versión, distribución y urgencia. El nombre debe coincidir con el nombre del paquete fuente, la distribución debería ser, por ahora, 'unstable' o 'experimental', y la urgencia no debería cambiarse a algo mayor que 'low'. :–)

Las línea 3–5 son una entrada de registro, donde documenta los cambios hechos en esta revisión del paquete (no los cambios de la fuente original – hay un fichero especial para este propósito, creado por los autores originales e instalado como /usr/share/doc/gentoo/changelog.gz). Las nuevas líneas deben ser insertadas justo antes de la línea que hay más arriba que comienza por un asterisco ('*'). Puede hacerlo con `dch(1)`, `emacs(1)` (las líneas 7 a la 9 contienen información de modo para el editor Emacs), o cualquier otro editor de texto. Terminará con algo así:

```
1  gentoo (0.9.12-1) unstable; urgency=low
2
3   * Initial Release.
4   * This is my first Debian package.
5   * Adjusted the Makefile to fix $DESTDIR problems.
6
7  -- Josip Rodin <jrodin@jagor.srce.hr> Wed, 11 Nov 1998 21:02:14 +0100
8
9  Local variables:
10  mode: debian-changelog
11 End:
```

(He añadido los números de línea.)

Cuando distribuya una nueva revisión, sólo debe incrementar el número de versión. Puede hacer esto sólo con 'dch –i' o explícitamente con 'dch –v <versión>–<revisión>' y después insertar los comentarios usando su editor preferido. Pista: ¿cómo obtener rápidamente la fecha en el formato necesario? Use '822–date', o 'date –R'.

La nueva información sobre la versión se añade al principio del fichero de cambios. Esto es como el fichero de cambios quedará después:

```
1  gentoo (0.9.12-2) unstable; urgency=low
2
```

```
  * Fixed a glitch in the menu file.

 -- Josip Rodin <jrodin@jagor.srce.hr> Wed, 11 Nov 1998 22:15:39 +0100

gentoo (0.9.12-1) unstable; urgency=low

  * Initial Release.
  * This is my first Debian package.
  * Adjusted the Makefile to fix $DESTDIR problems.

 -- Josip Rodin <jrodin@jagor.srce.hr> Wed, 11 Nov 1998 21:02:14 +0100

Local variables:
mode: debian-changelog
End:
```

(I've added the line numbers.)

Puede leer más sobre nuevas versiones/revisiones de paquetes más adelante en 'Actualizar su paquete' on page 34.

## 4.4 el fichero 'rules'.

Ahora volvemos al directorio 'debian' para mirar las reglas que `dpkg-buildpackage(1)` utilizará para crear el paquete. Este fichero es en realidad otro Makefile, ya que es ejecutado con 'make –f', pero diferente al que viene en las fuentes originales.

Cada fichero 'rules' (de reglas, n. del t.), como muchos otros Makefiles, consiste de varias reglas sobre cómo compilar las fuentes. Las reglas consisten en objetivos: ficheros o nombres de acciones que se deben llevar a cabo (por ejemplo, 'build:' o 'install:'). Las reglas que quiere ejecutar deberían llamarse como los argumentos de comandos (por ejemplo, './debian/rules build' o 'make –f rules install'). Después del nombre del objetivo, puede nombrar las dependencias, programas o ficheros de los que la regla depende. Después de esto hay un número cualquiera de instrucciones (¡indentado con <tab>!), hasta que se llega a una línea en blanco, ahí empieza otra regla. Líneas múltiples en blanco, o líneas que empiezan por almohadillas ('#') se tratan como comentarios y se ignoran.

Probablemente ya se haya perdido, pero todo quedará más claro después de ver un fichero 'rules' que dh_make pone por defecto. Debería también leer la entrada de 'make' en info para más información.

La parte importante que debe conocer sobre el fichero de reglas creado por dh_make, es que sólo es una sugerencia. Funcionará para paquetes simples pero para más complicados, no se asuste y añada o sustraiga de éste para ajustarse a sus necesidades. Una cosa que no debe cambiar son los nombres de las reglas, porque todas las herramientas utilizan estos nombres, como se indica en el Manual de Empaquetamiento.

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

```
# Made with the aid of dh_make, by Craig Small
# Sample debian/rules that uses debhelper. GNU copyright 1997 by Joey Hes
# Some lines taken from debmake, by Christoph Lameter.

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

# This is the debhelper compatability version to use.
export DH_COMPAT=1
build: build-stamp
build-stamp:
dh_testdir


# Add here commands to compile the package.
$(MAKE)

touch build-stamp

clean:
dh_testdir
dh_testroot
rm -f build-stamp

# Add here commands to clean up after the build process.
-$(MAKE) clean

dh_clean

install-stamp: build-stamp
dh_testdir
dh_testroot
dh_clean -k
dh_installdirs

# Add here commands to install the package into debian/tmp.
$(MAKE) install DESTDIR=`pwd`/debian/tmp

# Build architecture-independent files here.
binary-indep: build install
# We have nothing to do by default.

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

```
dh_testroot
#  dh_installdebconf
dh_installdocs
dh_installexamples
dh_installmenu
# dh_installemacsen
# dh_installpam
# dh_installinit
dh_installcron
dh_installmanpages
dh_installinfo
# dh_undocumented
dh_installchangelogs
dh_link
dh_strip
dh_compress
dh_fixperms
# You may want to make some executables suid here.
dh_suidregister
# dh_makeshlibs
dh_installdeb
# dh_perl
dh_shlibdeps
dh_gencontrol
dh_md5sums
dh_builddeb

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

(He añadido los números de línea)

(N. del T.: se traduce el fichero de reglas, dh_make sólo lo ofrece en inglés)

```
#!/usr/bin/make -f
# Creado con la ayuda de dh_make, por Craig Small
# Fichero de ejemplo debian/rules que usa debhelper. Copyright GNU 1997 p
# Algunas líneas son de debmake, por Christoph Lameter.

# Quitele el comentario para activar el modo verboso
#export DH_VERBOSE=1

# Esta es la versión de compatibilidad con debhelper a usar
export DH_COMPAT=1
build: build-stamp
build-stamp:
```

```
dh_testdir
# Añada aquí los comandos para compilar el paquete.
$(MAKE)
touch build-stamp
clean:
dh_testdir
dh_testroot
rm -f build-stamp
# Añada aquí los comandos para limpiar después del proceso de creación.
-$(MAKE) clean
dh_clean
install-stamp: build-stamp
dh_testdir
dh_testroot
dh_clean -k
dh_installdirs
# Añada aquí los comandos para instalar el paquete en debian/tmp.
$(MAKE) install DESTDIR=`pwd`/debian/tmp
# Construir los ficheros independientes de arquitectura aquí
binary-indep: build install
# Por defecto no se hace nada.
# Construir los ficheros dependientes de arquitectura aquí.
binary-arch: build install
# dh_testversion
dh_testdir
dh_testroot
#  dh_installdebconf
dh_installdocs
dh_installexamples
dh_installmenu
# dh_installemacsen
# dh_installpam
# dh_installinit
dh_installcron
dh_installmanpages
dh_installinfo
```

```
59 # dh_undocumented
60 dh_installchangelogs
61 dh_link
62 dh_strip
63 dh_compress
64 dh_fixperms
65 # Puede hacer algunos ejecutables suid aquí
66 dh_suidregister
67 # dh_makeshlibs
68 dh_installdeb
69 # dh_perl
70 dh_shlibdeps
71 dh_gencontrol
72 dh_md5sums
73 dh_builddeb
74
75 binary: binary-indep binary-arch
76 .PHONY: build clean binary-indep binary-arch binary install
```

Probablemente esté familiarizado con líneas como la 1 de guiones hechos en shell o perl. Esto significa que el fichero debe ejecutarse con make.

Las líneas 12 a la 19 describen la regla de construcción (en inglés: build, n. del t.) que ejecuta el Makefile del propio programa para compilarlo.

Las cosas rara vez funcionan perfectamente la primera vez, así que se especifica la regla de limpieza (del inglés ’clean’, n. del t.) en las líneas 18–26 que limpian cualquier resto innecesario dejado de intentos previos fallidos.

La regla ’clean’ (limpia, n. del t.), como se especifica en las líneas 21–29, limpia cualquier binario innecesario o cosas automáticamente generadas, dejadas después de la construcción del paquete. Esta regla debe funcionar en todo momento (incluso cuando el árbol de fuentes *está* limpio!), así que por favor use las opciones que fuerzan a hacer cosas (por ejemplo para rm, sería ‘–f’), o ignore los valores devueltos (con un ‘–’ al principio de la orden).

El proceso de instalación, la regla ’install’, comienza en la línea 31. Básicamente ejecuta la regla ‘install’ del Makefile del programa, pero lo instala en el directorio ‘pwd‘/debian/tmp – esto es por lo que especificamos $(DESTDIR) como el directorio raíz de instalación del Makefile de gentoo.

Como sugiere el comentario, la regla ’binary–indep’, en la línea 41, se usa para construir paquetes independientes de arquitectura. Como no tenemos ninguno aquí no se hará nada. Si su paquete es del tipo ‘Architecture: all’ necesita incluir todas las órdenes para crear el paquete bajo esta regla, y dejar las siguiente regla (‘binary–arch’) vacía en su lugar.

Lo siguiente es en la regla ’binary–arch’, en las líneas 45 a 73, en la que ejecutamos varias utilidades diversas del paquete debhelper que nos permiten hacer operaciones variadas en nuestro paquete para que cumpla la política de Debian.

Los nombres comienza con dh_ y el resto es la descripción de lo que la utilidad en particular realmente hace. Es todo más o menos auto–explicativo, pero a continuación se indica lo que realmente hace cada pequeña utilidad:

- `dh_testdir(1)` comprueba que estás en el directorio correcto (esto es el directorio raíz de la distribución de las fuentes),
- `dh_testroot(1)` comprueba que tienes permisos de superusuario que son necesarios para las reglas binary* y clean (root, n. del t.),
- `dh_installdirs(1)` crea los directorios que se mencionan en el fichero ’dirs’ [no existe aquí],
- `dh_installdocs(1)` copia la documentación al directorio debian/tmp/usr/doc/gentoo,
- `dh_installmenu(1)` copia el fichero ’menu’ a debian/tmp/usr/lib/menu/gentoo,
- `dh_installmanpages(1)` copia todas las páginas de manual que puede encontrar en el paquete fuente en el paquete (cuidado esto es DWIM),
- `dh_installchangelogs(1)` copia los ficheros ’changelogs’ (registros de cambios, n. del t.) en el directorio debian/tmp/usr/doc/gentoo,
- `dh_installinit(1)` copia los guiones init.d [aquí no tenemos ninguno],
- `dh_installcron(1)` copia los guiones de crontab a debian/tmp/etc/cron.* [aquí no hay ninguno],
- `dh_installexamples(1)` copia los ficheros de ejemplos a debian/tmp/usr/doc/gentoo/examples [aquí no hay ninguno],
- `dh_strip(1)` elimina las cabeceras de depuración de los ficheros ejecutables para hacerlos más pequeños,
- `dh_compress(1)` comprime con gzip las páginas de manual y los ficheros de documentación que sean más grandes de 4 kb,
- `dh_fixperms(1)` comprueba y arregla permisos no válidos en el directorio debian/tmp,
- `dh_suidregister(1)` adapta los ficheros para que se registren los ejecutables con ’setuid’ con `suidregister(8)` [aquí no hay ninguno],
- `dh_installdeb(1)` copia los ficheros relativos al sistema de paquetes en el directorio debian/tmp,
- `dh_makeshlibs(1)` genera el fichero de dependencias con librerías compartidas [aquí no existe],
- `dh_builddeb(1)` construye el paquete.
- `dh_installedeb(1)` copia los ficheros relacionados con los paquetes (por ejemplo los guiones del desarrollador) bajo el directorio debian/tmp/DEBIAN,

- `dh_shlibdeps(1)` calcula las dependencias de los ejecutables y librerías con las librerías compartidas ,
- `dh_gencontrol(1)` genera e instala el fichero de control,
- `dh_md5sums(1)` genera las sumas de chequeo MD5 para todos los ficheros del paquete.

Para información más completa de lo que hacen cada uno de estos guiones dh_* , y qué otras opciones tienen, por favor lea sus páginas de manual respectivas. Hay otros guiones con la misma nomenclatura (dh_*) que no se han mencionado aquí, pero pueden serle útiles. Si los necesita, lea la documentación de debhelper.

Las líneas 70 a la 74 son sólo algunas necesidades sobre las que puede leer en el manual de make. Por ahora, no es importante conocerlas.

La parte importante sobre el fichero de reglas creado por dh_make es que sólo es una sugerencia. Funcionará para paquetes simples pero para los más complicados no se asuste y modifique, borre o añada a éste para ajustarse a sus necesidades. Esto se aplica de forma especial a las secciones binary–arch, donde debería comentar las líneas que llaman a funciones que no necesita, en este caso he comentado las líneas sobre testversion, emacsen, pam, init, cron, manpages, info, undocumented, suidregister, makeshlibs, and perl, simplemente porque gentoo no las necesita. Sólo he añadido en la línea 60, 'FIXES', porque este es el nombre del fichero de cambios de las fuentes.

La única cosas que no debe cambiar son los nombre de las reglas, porque es necesario que se llamen de esta forma para que todas las herramientas que las usan utilicen estos mismos nombres, esto se obliga en nuestro manual de Empaquetamiento.

Las últimas dos líneas (junto con otras que no se explican) son cosas más o menos necesarios, sobre las que puede leer en el manual de make, y en el Manual de Empaquetamiento. Por ahora no necesita saberlas.

# Chapter 5

# Otros ficheros en el directorio debian/.

Verá que existen otros ficheros en el subdirectorio debian/, muchas de las cuales tendrán el sufijo '.ex', que indica que son ejemplos. Si quieres o deseas usar cualquiera de éstas, lee la información relativa (pista: Manual de Política), renombre–lo para eliminar el sufijo '.ex' , y modifíquelo y el fichero de reglas si fuera necesario. Algunos de estos ficheros, los más usados, se explican en las siguientes secciones.

## 5.1 README.debian (LEEME.debian, n. del t.)

Cualquier detalle extra en discrepancias entre el programa original y su versión debianizada debería estar documentada aquí. Dh_make crea una por defecto, y éste es su aspecto:

```
gentoo for DEBIAN
----------------------

<possible notes regarding this package - if none, delete this file>

Josip Rodin <jrodin@jagor.srce.hr>, Wed, 11 Nov 1998 21:02:14 +0100
```

Dado que no tenemos que poner nada aquí – está permitido borrarla. Por cierto, sí, puede renombrar el fichero a README.Debian :–)

## 5.2 conffiles

Una de las cosas más molestas de los programas es cuando pasas mucho tiempo y esfuerzo adaptando un programa y una actualización destroza todos tus cambios. Debian arregla este problema marcando los ficheros de configuración de forma que cuando actualizas un paquete no se le pregunta si desea mantener la nueva configuración o no. Lo consigue poniendo la ruta completa a cada fichero de configuración (se encuentran generalmente en /etc) una por línea, en un fichero llamado 'conffiles'.

Gentoo tiene un fichero de configuración, /etc/gentoorc, y meteremos éste en el fichero 'conffiles'. No es necesario tener este último fichero si su programa no tiene ningún fichero de configuración.

## 5.3 dirs

Este fichero especifica los directorios que se necesitan pero que por alguna razón no crea un proceso de instalación normal (make install). Por defecto, tiene éste aspecto:

```
usr/bin
usr/sbin
```

Observe que la barra precedente no está incluida. Normalmente lo cambiaríamos para gentools a algo así:

```
usr/X11R6/bin
usr/X11R6/man/man1
```

pero estos directorios ya se crean en el Makefile, así que no necesitaremos este fichero y lo podremos borrar.

## 5.4 manpage.1.ex

Los ficheros que terminan en *.ex son ejemplos de cómo añadir ese tipo de soporte en el paquete. Si usa uno de ellos, editelo y elimine la extensión .ex. Si no lo quiere usar, simplemente borrelo.

Su programa debería tener una página de manual. Si no la tiene, éste es un esqueleto que puede rellenar. Lea las páginas de manual para `man(7)` para una breve descripción de cómo crear una página de manual. Asegúrese de renombrar este fichero al nombre del programa y modificar la extensión para indicar la sección a la que debería ir. Aquí hay una corta lista:

```
       Section |      Deguiónion         |      Notes
          1     Comandos de Usuario       Programas ejecutables o guiones.
          2     Llamadas al Sistema       Funciones dadas por el kernel.
          3     Llamadas a Librerías      Funciones dadas por las libr-
erías del sistema.
          4     Ficheros Especiales      Generalmente se encuentra en /dev
          5     Formatos de Fichero      Por ejemplo, el formato del /etc/passwd
          6     Juegos                   U otros programas frívolos.
          7     Paquetes de Macros       Como las macros de man.
          8     Administración del Sist. Programas que sólo ejecuta el superusuario
          9     Rutinas del Kernel      Llamadas al sistema no estándar.
```

Así que la página de manual de gentoo debería llamarse gentoo.1, o gentoo.1x porque es un programa de X11. Cómo no había página de manual en las fuentes originales la escribí del ejemplo.

## 5.5 menu.ex

Los usuarios de X Windows tendrán un gestor de ventanas con menús que pueden adaptarse para lanzar programas. Si tienen instalado el paquete de menu de Debian, se creará un conjunto de menús para cada programa del sistema para ellos. No se exige en la política de Debian, pero los usuarios seguramente lo apreciarán. Podemos añadir a Gentoo a los menús editando este fichero. Aquí está el fichero que dh_make crea por defecto:

```
?package(gentoo):needs=X11|text|vc|wm section=Apps/vea-manual-menu\
  title="gentoo" command="/usr/bin/gentoo"
```

El primer campo especifica qué tipo de interfaz necesita el programa (esto es, texto o X11). La siguiente es el menu y submenu dónde debería aparecer. La lista actual de secciones está en /usr/share/doc/debian–policy/menu–policy.html/ch2.html#s2.1 El tercero es el nombre del programa. El cuarto es el icono para el programa o nada si no hay ninguno. El quinto es el texto que aparecerá en el menu. Finalmente, el sexto es la orden que ejecuta el programa.

Ahora cambiaremos la entrada de menu a ésta:

```
?package(gentoo):needs=X11 section=Apps/Misc \
  title="gentoo" command="/usr/X11R6/bin/gentoo"
```

Vea `menufile(5)`, `update-menus(1)` y /usr/share/doc/debian–policy/menu–policy.html/ para más información.

## 5.6 watch.ex

Puede usar este fichero, junto con los programas `uscan(1)` y `uupdate(1)` (en el paquete devscripts) para vigilar el servidor de donde obtuvo las fuentes originales.

Esto es lo que he puesto yo:

```
# watch control file for uscan
# Site Directory Pattern Version Guión
ftp.obsession.se /gentoo gentoo-(.*)\.tar\.gz debian uupdate
```

Pista: conectese a Internet, e intente ejecutar el programa “uscan” en el directorio donde ha creado el fichero. Vea la página de manual para más detalles.

## 5.7 ex.doc–base

Si su paquete tiene documentación en HTML o de otro tipo (excepto páginas de manual y documentos info), debería usar el fichero ‘doc–base’ para registrarlo para que el usuario pueda encontrarlo con por ejemplo `dhelp(1)` o `dwww(1)`.

Este es el aspecto del fichero doc–base:

```
Document: gentoo
Title: Gentoo Manual
Author: Emil Brink
Abstract: This manual describes what Gentoo is, and how it can be used.
Section: Apps/Tools

Format: HTML
Index: /usr/share/doc/gentoo/html/index.html
Files: /usr/share/doc/gentoo/html/*.html
```

Para información sobre el formato del fichero, vea `install-docs(8)` y el manual doc–base en /usr/doc/doc–base/doc–base.html/index.html.

## 5.8 postinst.ex, preinst.ex, postrm.ex, prerm.ex

Estos ficheros se llaman guiones de gestión, y debería intentar evitarlos si puede porque son demasiados complejos. Para más información lea el Manual de Empaquetamiento.

Por ahora, debería intentar evitar editar manualmente estos guiones si puede porque suelen hacerse muy complejos. Para más información lea el Manual de Empaquetamiento (sección 6) y eche un vistazo a los ejemplos dados por dh_make.

Ahora estamos preparados para construir el paquete.

# Chapter 6

# Pasos Finales

## 6.1 Construir el paquete.

Entre en el directorio principal de y ejecute el siguiente comando:

```
dpkg-buildpackage -rfakeroot
```

Esto hará todo por usted, sólo tendrá que dar su clave secreta PGP, dos veces. Una vez haya hecho esto, verá cuatro nuevos ficheros en el directorio anterior (~/debian/):

- *gentoo_0.9.12–1_i386.deb*

  es el paquete binario completo. Puede usar dpkg o dselect para instalar o eliminar éste como cualquier otro paquete.

- *gentoo_0.9.12.orig.tar.gz*

  Este es el código fuente original empaquetado de forma que si alguien quiere recrear su paquete desde cero puede hacerlo. O si alguien no está usando el sistema de paquetes de Debian quiere descargarse las fuentes y compilarlo manualmente.

- *gentoo_0.9.12–1.dsc*

  Este es un sumario de los contenidos del código fuente. Este fichero es generado con el fichero gentoo–0.9.12/debian/control y se usa cuando se descomprime las fuentes con `dpkg-source(1)`. Éste fichero está firmado con PGP de forma que cualquier puede estar seguro de que es realmente suyo.

- *gentoo_0.9.12–1.diff.gz*

  Este fichero comprimido contiene todos y cada uno de los cambios que hizo al código fuente original, en un formato conocido como "diff unificado". Está hecho y lo utiliza `dpkg-source(1)`.

- *gentoo_0.9.12–1_i386.changes*

  Este fichero describe todos los cambios hechos en la revisión actual del paquete, y se usa por los programas de gestión del archivo FTP para instalar los paquetes binarios y fuentes en él. Se genera parcialmente gracias al fichero gentoo–0.9.12/debian/changelog y el fichero .dsc.

  Mientras sigue trabajando en el paquete, cambiará su comportamiento y se le añadirán nuevas funciones. Las personas que descarguen su paquete pueden mirar este fichero y ver qué ha cambiado. Las largas listas de números son las sumas MD5 para los ficheros. Las personas que descarguen estos ficheros pueden probarlos con `md5sum(1)` y si los números no coinciden, sabrán que el fichero está corrupto o ha sido modificado. Éste fichero está firmado con PGP de forma que cualquiera puede estar aún más seguro de que es realmente suyo.

Con un paquete grande, puede que no quiere recompilar desde cero cada vez que toca un detalle en el fichero debian/rules. Para propósitos de prueba, puede hacer un fichero .deb sin necesidad de recompilar las fuentes originales de esta forma:

```
fakeroot debian/rules binary
```

Sólo asegúrese de que su regla 'install' *no* tiene un hijo 'install–stamp' (éste es el valor por defecto actualmente), para asegurarse de que se ejecuta 'dh_clean –k' cada vez. Y una vez que ha terminado con los cambios, no se olvide de reconstruir el paquete siguiendo el procedimiento habitual, para poder enviarlo de la forma correcta.

## 6.2 Comprobar su paquete para encontrar errores.

Ejecute `lintian(1)` sobre su fichero de cambios .changes; éste programa comprobará muchos errores comunes al empaquetar. El comando es:

```
lintian -i gentoo_0.9.12-1_i386.changes
```

Por supuesto, cambia el nombre de fichero con el nombre del fichero de cambios generado por su paquete. Si parece que hay algunos errores (líneas que comienzan por E:), lea la explicación (líneas N:), corrija errores, y reconstruya como se describe en 'Construir el paquete.' on the preceding page. Si hay líneas que comienzan con W:, son sólo avisos (warnings, n. del t.), así que puede estar seguro de que su paquete está bien (pero seguramente necesita algún ajuste fino).

Observe que puede construir el paquete con dpkg–buildpackage y ejecutar lintian todo de una sola vez con `debuild(1)`.

Mire dentro del paquete usando un gestor de ficheros como `mc(1)`, o descomprimalo en algún punto temporal usando `dpkg-deb(1)`. Esté atento a ficheros innecesarios extra y algunas otras cosas que no se limpiaron. Pista: 'zgrep ^+++ ../gentoo_0.9.12–1.diff.gz' le dará una lista de sus cambios/modificaciones a las fuentes, y 'dpkg–deb –c gentoo_0.9.12–1_i386.deb' listará los ficheros en el paquete.

Instale el paquete para probarlo usted mismo, esto es usando la orden `debi(1)` como superusuario. Intente instalarlo y ejecutarlo en otras máquinas distintas de la suya y esté atento para detectar errores o avisos tanto en la instalación como en la ejecución del programa.

Más tarde, cuando construya una nueva versión, debería hacer lo siguiente para asegurar la actualización básica del paquete:

- actualízese de la versión previa (y de la versión en la última versión de Debian),
- vuelva a la versión anterior de nuevo,
- instale el paquete como uno nuevo (esto es, sin ninguna versión instalada previamente),
- desintálelo, reinstálelo y luego purgelo.

## 6.3 Enviando su paquete.

Ahora que ha probado su nuevo paquete en profundidad, necesitará enviar estos ficheros a master.debian.org, usando `dupload(1)`. Primero debe editar el fichero de configuración de dupload, ~/.dupload.conf. Copie los valores por defecto del /etc a su directorio personal: Puede poner algo así:

```
package config;
$default_host = "master";

$cfg{"master"}{"method"} = "scpb";
$cfg{"master"}{"login"} = "joy";
$cfg{"master"}{"visibleuser"} = "jrodin";
$cfg{"master"}{"visiblename"} = "jagor.srce.hr";
$cfg{"master"}{"fullname"} = "Josip Rodin";

$cfg{"non-us"}{"method"} = "scpb";
$cfg{"non-us"}{"login"} = "joy";
$cfg{"non-us"}{"visibleuser"} = "jrodin";
$cfg{"non-us"}{"visiblename"} = "jagor.srce.hr";
$cfg{"non-us"}{"fullname"} = "Josip Rodin";

1;
```

Por supuesto, cambie mis valores personales a los suyos, y lea la página de manual `dupload.conf(5)` para comprender qué significa cada una de estas opciones.

Entonces conecte con su proveedor de Internet, asegurese una vez mas de que está en el directorio /usr/local/src/gentoo, y ejecute la orden:

```
dupload --to master gentoo_0.9.12-1_i386.changes
```

Dupload comprobará que las sumas md5 coinciden con aquellas en el fichero .changes, y le avisará de rehacer el paquete como se describe en 'Construir el paquete.' on page 31 para poder enviarlo correctamente.

Dupload le preguntará su password en master.debian.org, envíe sus paquetes, y hará un pequeño anuncio sobre su envío en `<debian-devel-changes@lists.debian.org>`.

Si vive en Europa, puede usar otras colas de envío en lugar de master. Para más detalles mire lea `dupload(1),dupload(5)` y la Referencia del Desarrollador.

## 6.4 Actualizar su paquete

Supongamos que se ha registrado un informe de bug contra su paquete, #54321, y que describe un problema que puede solucionar. Para crear una nueva revisión del paquete, necesita:

- Corregir, por supuesto, el problema en las fuentes del paquete.
- Añadir una nueva revisión en el fichero de cambios (changelog, n. del t.) de Debian, con 'dch –i', e incluir una breve descripción del error y su solución seguido por: "Closes: #54321". De esta forma, el informe del error será automágicamente cerrado por los programas de gestión del archivo en el momento en que su paquete se acepte en e el archivo de Debian.
- Repita lo que hizo en 'Construir el paquete.' on page 31, 'Comprobar su paquete para encontrar errores.' on page 32, y 'Enviando su paquete.' on the preceding page. La diferencia es que esta vez ,las fuentes originales del archivo no serán incluidas, dado que no han cambiado y ya existen en el archivo de Debian.

Ahora consideremos una situación diferente, algo más complicada: una versión nueva de las fuentes originales ha salido, y, por supuesto, desea empaquetarla. Debe hacer lo siguiente:

- Descargar las nuevas fuentes y ponerlas en un archivo tar (esto es llamado 'gentoo–0.9.13.tar.gz') un directorio por encima de su antiguo árbol de fuentes (por ejemplo ~/debian/).
- Entre en el antiguo directorio y ejecute:

  ```
  uupdate -u gentoo-0.9.13.tar.gz
  ```

  Por supuesto, reemplace este nombre de fichero con el nombre de las fuentes de su programa. `uupdate(1)` renombrará apropiadamente este fichero tar, intente aplicar los cambios de su fichero .diff.gz previo y actualizar el nuevo fichero debian/changelog.
- Cambie el directorio a '../gentoo–0.9.13', el nuevo directorio fuente del paquete, y repita la operación que hizo en 'Construir el paquete.' on page 31, 'Comprobar su paquete para encontrar errores.' on page 32, y 'Enviando su paquete.' on the page before.

Observe que si ha puesto el fichero 'debian/watch' como se describe en 'watch.ex' on page 29, puede ejecutar automáticamente `uscan(1)` para ver fuentes revisadas, descargarlas, y ejecutar uupdate.

## 6.5 Dónde pedir ayuda.

Antes de que decida preguntar en lugares públicos, por favor simplemente RTFM ("Lea el Jodido Manual", n. del t.). Esto incluye documentación en /usr/share/doc/dpkg, /usr/share/doc/debian, /usr/share/doc/package/* y las páginas de man/info para todos los programas mencionados en este artículo. Cuando reciba un aviso de fallo (bug report, n. del t.) (sí, avisos de fallos ¡de verdad!) sabrá que es el momento de indagar en el Sistema de Seguimiento de Fallos de Debian (`http://www.debian.org/Bugs/`) y leer la documentación allí, para poder tratar los informes de forma eficiente.

Si se une a la lista de distribución de Mentores de Debian en `<debian-mentors@lists.debian.org>` puede unirse con desarrolladores de Debian con experiencia que le ayudarán con las preguntas que pueda tener. Puede subscribirse a ella enviando un correo electrónico a `<debian-mentors-request@lists.debian.org` con la palabra 'subscribe' en el tema del mensaje.

Si aún tiene preguntas, hagalas en la lista de distribución de Desarrolladores de Debian en `<debian-devel@lists.debian.org>`. Puede subscribirse a ella enviando un correo electrónico a `<debian-devel-request@lists.debian.org>` con la palabra 'subscribe' en el tema del mensaje. Si ya es un desarrollador de Debian debería estar suscrito a ella de todas formas.

Aunque todo funcionara bien, es el momento de empezar a rezar. ¿Por qué? Por que en sólo unas horas (o días) usuarios de todo el mundo empezarán a usar su paquete, y si cometió algún error crítico será bombardeado por correos de centenares de usuarios furiosos de Debian.... Sólo bromeaba :–)

Relájese y prepárese para recibir reportes de fallos, porque hay mucho más trabajo que hacer antes de seguir completamente las políticas de Debian (una vez más lea la *documentación real* para detalles). ¡Buena suerte!