Guía del Nuevo Desarrollador de Debian

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

versión 1.0 (translation 1.0), 25 de enero de 2000

Nota de Copyright

Copyright © 1998, 1999, 2000 Josip Rodin. Translation Copyright © 1999, 2000, 2001 Javier Fernández–Sanguino Peña.

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

Este documento se ha hecho usando estos dos documentos como ejemplo:

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>.

Índice General

Capítulo 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 necesitarás instalar programas que no estén así. Puedes preguntarte cómo harás tus propios paquetes y que quizás ésta sea una tarea demasiada difícil. Bueno, si eres 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. Observa que en la lista no están incluidos paquetes ‘esenciales’ o ‘necesarios’ que se suponen ya instalados.

Este documento fue escrito cuando la distribución ‘slink’ 2.1 era la 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 tengas (junto con los paquetes de los que dependen). Aún así, deberías comprobarlo con ‘dpkg –s <paquete>‘.

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

- *cpp* – el preprocesador de C. (ve a `cpp(1)`)
- *cpio* – este es un programa archivador como tar o zip. (ve a `cpio(1)`)
- *dpkg–dev* – este paquete contiene las herramientas necesarias para desempaquetar, construir y enviar paquetes fuente de Debian. (ve a `dpkg-source(1)`)
- *file* – este útil programa puede determinar de qué tipo es un fichero. (ve a `file(1)`)
- *gcc* – el compilador de C de GNU. La mayor parte de los programas de Linux están escritos en C. Si tu programa está escrito en algún otro lenguaje de programación como C++, Fortran o Pascal, deberías instalar g++, g77, o gpc, respectivamente. (lee `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 recomiendan y/o usan libc5, te sugiero que utilices la nueva versión (libc6). (revisa ‘info libc‘)
- *make* – habitualmente la creación de un programa tiene varios pasos. En lugar de ejecutar los mismos comandos una y otra vez, puedes utilizar este programa para automatizar el proceso, creando ‘Makefile’s. 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 puedes necesitarlos también. (ve a ‘info make‘, `imake(1)`, `xmkmf(1)`, `autoconf(1)`, `automake(1)`)
- *patch* – esta utilidad es muy práctica, ya que permite tomar un fichero que contiene un listado de diferencias (producido por el programa diff) y aplicárselas al fichero original, produciendo una versión “parcheada”. (ve a `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”. (ve `perl(1)`)

De la sección ‘devel’ de la distribución posiblemente necesites instalar esto tú 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. (ve a `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. (ve a /usr/share/doc/devscripts/README.gz)
- *fakeroot* – esta utilidad te permite emular ser root (superusuario, n. del t.) lo cual es necesario para ciertas partes del proceso de construcción. (ve a `fakeroot(1)`)
- *lintian* – este es el gestor de paquetes de Debian, que te 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. (ve a `lintian(1)`, `diffstat(1)`, /usr/share/doc/lintian/lintian.html/index.html)

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 del Sistema de Ficheros (Filesystem Hierarchy Standard, n. del t.), y, lo más importante (para ti) es que describe los requisitos que debe satisfacer cada paquete para ser incluido en la distribución. (mira /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, cómo renombrar, abandonar, coger paquetes, cómo hacer NMUs (Non–Maintainer Uploads, o envíos por personas distintas del desarrollador, n. del t.), como gestionar los errores que los usuarios envíen, y cómo y cuando enviar los paquetes, etc. (lee /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. (lee /usr/share/doc/packaging–manual/packaging.html/index.html)

También necesitarás un paquete criptográfico, o bien PGP (los paquetes pgp–*) o GPG (el paquete gnupg), para *firmar* digitalmente tu paquete. Esto es especialmente importante si quieres distribuir tu paquete a otras personas (y harás precisamente esto si tu paquete se incluye en la distribución de Debian). Sin embargo, debido a una estúpida ley de exportación de los EEUU, no puedes simplemente obtenerlo de tu servidor 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/). Tu servidor FTP tendrá un fichero llamado README.non–us, que te indicará dónde está la réplica más cercana de este servidor.

Las breves descripciones dadas anteriormente sólo sirven para introducirte a lo que hace cada paquete. Antes de continuar, por favor, lee la documentación de cada programa, al menos para su uso normal. Puede parecerte algo duro ahora, pero más adelante estarás *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. Lee el manual de Debmake (`http://www.debian.org/~jaldhar/`) para más información.

1.2 Más información.

Puedes 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 mezcles los términos como ’fuentes de un programa’ y el ’paquete fuente de un programa’! Por favor, lee los otros manuales si necesitas 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 tú haces un programa, y quieres incluirlo en Debian, tienes total libertad para solicitar convertirte en desarrollador.

Después de construir tu paquete (o mientras lo haces), deberás convertirte en un desarrollador oficial de Debian si deseas que tu 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, léela.

Capítulo 2

Primeros pasos.

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

2.1 Elige tu programa.

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

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

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

- consulta la página WNPP (paquetes en Perspectiva o que Necesitan Ayuda) (`http://www.debian.org/devel/wnpp/`) y los archivos de la lista de correo debian–devel para ver si alguna otra persona está construyendo ese mismo programa. Si es así, contacta con el desarrollador actual si crees que lo necesita mantener. Si no, intenta buscar otro programa interesante que nadie mantenga.
- el programa **debe** tener una licencia, si es posible, libre en el sentido marcado por las Directrices de Debian para el software libre (`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 estás seguro sobre en qué lugar debería ir, pregunta en `<debian-legal@lists.debian.org>`.

- el programa **no** debería ejecutarse con setuid root, o aún mejor: no debería tener ningún programa 'setuid' ni 'setgid'.
- el programa no debería ser un demonio, o algo que vaya en los directorios */sbin.
- el programa debería estar compuesto por binarios ejecutables, no lo intentes aún con librerías.
- debería tener una documentación buena, o al menos legible (para cualquiera).
- deberías contactar con el/los autor/es del programa para comprobar si están de acuerdo con que se empaquete. Esto es importante para consultar al/los autor/es sobre el programa en caso de que haya problemas específicos, así que no intentes empaquetar programas que no estén mantenidos.
- y por último pero no menos importante, deberías saber cómo funciona, y haberlo utilizado durante algún tiempo.

Por supuesto, esta lista es para tomar medidas de seguridad, y con la intención de salvarte de usuarios enfurecidos si haces algo mal con algún demonio setuid... Pero cuando tengas más experiencia en empaquetar cosas, podrás 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 te ayudará gustosamente.

Para más ayuda sobre esto, lee la Referencia del Desarrollador.

2.2 Obtén el programa, y pruébalo.

La primera cosa a hacer es encontrar y descargar el paquete original. Supongo que ya tienes el código fuente que obtuviste de la página del autor. Las fuentes de los programas gratuitos de Linux generalmente vienen en formato tar/gzip, con extensión .tar.gz, y generalmente contienen un subdirectorio llamado programa–versión con todas las fuentes en él. Si tu programa viene en otro tipo de archivo (por ejemplo, el fichero termina en ".Zº ".zip"), descomprímelo con las herramientas adecuadas, o pregunta en debian–mentors si no estás seguro de cómo se puede desempaquetar correctamente (pista: prueba 'file archivo.extensión').

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

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

Ahora tienes otro subdirectorio, llamado 'gentoo–0.9.12'. Muévete a ese directorio y lee **en profundidad** la documentación que encuentres. Está generalmente en ficheros que se llaman README*, INSTALL*, *.lsm o *.html. Allí encontrarás instrucciones de cómo compilar e instalar el programa (muy probablemente asumirán que lo quieres instalar en el directorio /usr/local/bin: no harás esto, pero eso lo veremos más adelante en 'Instalación en un subdirectorio.' en la página 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 tu sistema y se asegura de que tu 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 comprobaciones automáticas. Generalmente se instalarán en sus directorios de destino ejecutando 'make install'.

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

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

2.3 Cosas antes de 'dh_make'.

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

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

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

Comprueba también 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, eres 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 de las fuentes es el 19 de diciembre de 1998, puedes utilizar la abreviatura norteamericana 0.0.19981219 como número de versión. Aún así habrá algunos que ni siquiera estén numerados, en cuyo caso deberás contactar con el 'desarrollador fuente' para ver si tienen algún otro sistema de seguimiento de revisiones.

2.4 Ejecutando 'dh_make'.

Asegúrate que estás en el directorio donde están las fuentes del programa, y ejecuta lo siguiente:

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

Por supuesto, cambia la cadena "tu.dirección@de.desarrollador" 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. Lee `dh_make(1)` para más detalles.

Saldrá alguna información. Te preguntará qué tipo de paquete deseas 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’. Comprueba la información que aparece en la pantalla y confirma 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.

Ten en cuenta que deberías ejecutar dh_make **sólo una vez**, y que no se comportará correctamente si lo haces otra vez en el mismo directorio, ya “debianizado”. Esto también significa que usarás un método distinto para crear una nueva revisión o una nueva versión de tu paquete en el futuro. Lee más sobre esto más adelante en ‘Actualizar tu paquete.’ en la página 32.

Capítulo 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 usuarios). Esto significa que tienes que mirar el sistema de construcción de tu programa, generalmente empezando por el Makefile. Éste es el guión que `make(1)` usará para automatizar la creación de este programa. Para más detalles sobre Makefiles, mira en 'El fichero 'rules'.' en la página 19.

Date cuenta que si tu programa usa GNU `automake(1)` y/o `autoconf(1)`, lo que quiere decir que las fuentes incluyen ficheros Makefile.am y Makefile.in, respectivamente, necesitarás modificar esos ficheros, porque cada invocación de automake reescribirá los Makefile.in's con información generada a partir de los Makefile.am's, y cada llamada a ./configure hará lo mismo con los Makefile's, con información de los Makefile.in's. Editar los ficheros Makefile.am requiere algunos conocimientos de automake, sobre lo que puedes leer en la entrada de info para automake, mientras que editar Makefile.in's es casi lo mismo que editar ficheros Makefile, simplemente basta con poner atención en las variables, es decir, cualquier cadena que empiece y acabe por '@'s, como por ejemplo @CFLAGS@ o @LN_S@, que se rellenan con cada vez que se ejecute ./configure.

Ten 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 tu sistema, para que los binarios sean incluidos en tu $PATH, y para que encuentre la documentación y páginas de manual en los lugares habituales. Tienes que asegurarte que esto se realice de forma correcta, pero tienes que hacer que se instale en un directorio temporal que se creará bajo tu 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, debes 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 utilicen la herramienta GNU autoconf, ésto será bien sencillo, porque dh_make pondrá las opciones necesarias para hacer esto automáticamente, así que podrías dejar de leer esta sección. Pero con otros programas, puede ser necesario que examines y edites los Makefiles.

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

Antes de esto deberías insertar dos nuevas líneas que digan:

```
        # Edited for Debian GNU/Linux.
        DESTDIR =
```

porque el proceso de construcción lo necesita (explicado más tarde, en 'El fichero 'rules'.' en la página 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 especificadas en el Estándar de la Jerarquía del Sistema de Ficheros (Filesystem Hierarchy Standard, n. del t.) (/usr/share/doc/debian–policy/fhs). 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 tendremos un situación un poco más complicada. Si dejas alguna línea como:

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

que se ajusta a la política de Debian, deberás editar algunos ficheros de fuentes reales de C. Pero, ¿dónde buscar? Puedes probar a encontrarlos usando:

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

(en cada subdirectorio que contenga ficheros .c y .h). Grep dirá el nombre del fichero y la línea, cuando encuentra una ocurrencia. Ahora edita esos ficheros y cambia en esas líneas usr/local/lib con usr/share – y ya está. Simplemente reemplaza usr/local/lib por tu localización, y sé muy cuidadoso para no tocar el resto del código, si no sabes mucho sobre cómo programar en C. :–)

Después de esto deberías encontrar el objetivo 'install' (busca una línea que comience por 'install:') y renombra todas las referencias a directorios distintos de los definidos al comienzo del Makefile. Anteriormente el objetivo 'install' decía:

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

# You're going to have to be root to do this!
install:        gentoo
                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/share/doc/gentoo/html
                cp -a docs/* $(DESTDIR)/usr/share/doc/gentoo/html
```

Un lector atento se dará cuenta de que he cambiado 'gentoo' a 'gentoo–target' en la línea 'install:'. A eso se le llama arreglar un fallo en el programa. :–)

Siempre que hagas cambios que no estén específicamente relacionados con el paquete Debian, asegúrese de que los envías al desarrollador original para que éste los incluya en la próxima revisión del programa. Observa que no tienes que enviar ninguno de los ficheros debian/* al desarrollador original, pero deberías hacerlo con cualquier otro parche. Y sé amable haciendo que tus cambios no sean 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 exista en Debian y sirva para el mismo propósito. La mejor forma es comentar esas líneas porque puede 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 (o Makefile.in) de tu programa que dice algo como lo siguiente (y tu programa no compila):

```
LIBS = -lcurses -lcosas -lmáscosas
```

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

```
LIBS = -lncurses -lcosas -lmáscosas
#LIBS = -lcurses -lcosas -lmáscosas
```

Capítulo 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 son necesarios 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}
10 Description: <insertar unos 60 caracteres de descripción>
11  <inserta una descripción ligera, indentada con espacios>
```

(He añadido los números de línea).

Las líneas 1–5 son la información de control para el paquete fuente. La línea 1 es el nombre del paquete fuente.

La línea 2 es la sección de la distribución dentro de la que estará este paquete. Como puedes haber notado, Debian está dividida en secciones: main (principal, n. del t.) (el software libre), non–free (no libre, n. del t.) (el software que realmente no es libre) y contrib (software libre que depende de software no libre). Bajo ellas, hay subdivisiones lógicas que describen en una palabra qué paquetes hay dentro. Así que tenemos 'admin' para programas que sólo usa un administrador, 'base' para las herramientas básicas, 'devel' para las herramientas de programación, 'doc' para la documentación, 'libs' para las bibliotecas o librerías, 'mail' para lectores y demonios de correo–e, 'net' para aplicaciones y demonios de red, 'x11' para programas específicos de X11, y muchos más.

Vamos a cambiarla para que ponga x11.

La línea 3 describe cómo de importante es para el usuario la instalación de este paquete. La sección y prioridad actualmente las usa sólo dselect cuando ordena paquetes y selecciona valores por defecto, y pueden ser modificadas (y generalmente lo serán) por nuestros administradores del FTP. Lee 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 (al menos, los dos primeros números de la versión del paquete debian–policy actual).

Si se necesita un compilador no estándar o alguna otra herramienta para construir el paquete, deberías añadir aquí una línea 'Build–Depends' y listar los paquetes necesarios. Para más información sobre ésto, lee 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)` la rellenará con el valor apropiado cuando se compile este paquete (lea la Referencia del Desarrollador para una explicación de qué es portar paquetes). Si tu paquete es independiente de la arquitectura (por ejemplo, un documento, un programa de shell o un guión en Perl), cambia esto a "all", y lee más adelante 'El fichero 'rules'.' en la página 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.).

Las herramientas de gestión de paquetes como dpkg, dselect o apt (y sus respectivos front–ends) normalmente se comportan de la misma forma cuando tratan con esas relaciones entre paquetes; si no es así, se explicará en cada caso. (ve `dpkg(8)`, `dselect(8)`, `apt(8)`, `console-apt(8)`, `gnome-apt(8)`)

Esto es lo que generalmente significan:

- Depends:

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

- Recommends:

  Dselect no te permitirá instalar el paquete a menos que los paquetes que recomienda estén instalados. Dpkg y apt, sin embargo, sí te dejarán. Usa 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 le hacen caso a esto. Utiliza esto para paquetes que funcionarán bien con tu 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*. Utiliza esto **muy** poco y sólo después de haberlo discutido en la lista de distribución de debian–devel. En resumidas cuentas: no lo utilices en absoluto. :–)

- Conflicts:

  El paquete no será instalado hasta que todos los paquetes con los que entra en conflicto hayan sido eliminados. Utiliza esto si tu programa no funcionará en absoluto (o causará daños) si un paquete en particular está presente.

- Provides:

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

- Replaces:

  Use esto si tu 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 tuyo.

Todos estos campos tienen una sintaxis uniforme. Son 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 aplicación a versiones determinadas de cada paquete nombrado. Esto se hace listando después de cada nombre de paquete individual las versiones entre paréntesis, e indicando antes del número de versión una relación de la siguiente lista. Las relaciones permitidas son: <<, <=, =, >= y >> para estrictamente anterior, anterior o igual, exactamente igual, posterior o igual o estrictamente posterior, respectivamente.

La última funcionalidad que necesitas conocer es $(shlibs:Depends). Ésta se generará automáticamente gracias a `dh_shlibdeps(1)` y será rellenada por `dh_gencontrol(1)` con los nombres de cualquier librería compartida, como libc6 o xlib6g, que use tu programa, así que no necesitas especificar esto usted mismo. Habiendo dicho todo esto, puedes 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" ("Un gestor de ficheros GTK+ completamente configurable por GUI").

La línea 12 es donde va la descripción larga del paquete. Debería ser al menos un párrafo que dé 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>
       5  Standards-Version: 3.0.1
       6
       7  Package: gentoo
       8  Architecture: any
       9  Depends: ${shlibs:Depends}
       10 Suggests: file
       11 Description: A fully GUI configurable GTK+ file manager
       12  gentoo is a file manager for Linux written from scratch in pu-
re C. It
       13  uses the GTK+ toolkit for all of its interface needs. gentoo provides
       14  100% GUI configurability; no need to edit config files by hand and re-

       15  start the program. gentoo supports identifying the type of various
       16  files (using extension, regular expressions, or the 'file' com-
mand),
       17  and can display files of different types with different co-
lors and icons.
       18  .
       19  gentoo borrows some of its look and feel from the classic Ami-
ga file
       20  manager "Directory OPUS" (written by Jonathan Potter).
```

(He añadido los números de línea).

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 números de línea).

Las cosas importantes que se deben añadir a este fichero son el lugar de donde obtuviste el paquete y la nota de copyright y licencia originales. Debes 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', donde basta referirse al fichero apropiado en el directorio /usr/share/common–licenses/ que existe en todo sistema Debian. Gentoo está gobernado 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 lo que 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 es el formato que usan dpkg y otros programas para obtener el número de versión, revisión, distribución y urgencia de tu paquete.

Para ti es también importante, ya que es bueno tener documentados todos los cambios que hayas hecho. Le ayudará a las personas que se descarguen tu 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:

```
1  gentoo (0.9.12-1) unstable; urgency=low
2
3   * Initial Release.
4
5  -- Josip Rodin <jrodin@jagor.srce.hr> Wed, 11 Nov 1998 21:02:14 +0100
6
7  Local variables:
8  mode: debian-changelog
9  End:
```

(He añadido los números de línea).

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 se documentan los cambios hechos en esta revisión del paquete (no los cambios en las fuentes originales – 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 insertarse 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:

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

 * 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:
```

(He añadido los números de línea).

Puede leer más sobre nuevas versiones/revisiones de paquetes más adelante en ‘Actualizar tu paquete.’ en la página 32.

4.4 El fichero ‘rules’.

Ahora necesitamos mirar las reglas exactas que `dpkg-buildpackage(1)` utilizará para crear el paquete. Este fichero es en realidad otro Makefile, ya que se ejecuta con ‘make –f’, pero diferente al que viene en las fuentes originales.

Cada fichero ‘rules’ (de reglas, n. del t.), como muchos otros Makefiles, se compone de varias reglas que especifican cómo tratar las fuentes. Cada regla se compone de objetivos, ficheros o nombres de acciones que se deben llevar a cabo (por ejemplo, ‘build:’ o ‘install:’). Las reglas que quieras ejecutar deberían llamarse como argumentos de la línea de órdenes (por ejemplo, ‘./debian/rules build’ o ‘make –f rules install’). Después del nombre del objetivo, puedes nombrar las dependencias, programas o ficheros de los que la regla dependa. 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. Las líneas múltiples en blanco, y las líneas que empiezan por almohadillas (’#’) se tratan como comentarios y se ignoran.

Probablemente ya te hayas perdido, pero todo quedará más claro después de ver un fichero ‘rules’ que dh_make pone por defecto. Deberías también leer la entrada de ‘make’ en info para más información.

La parte importante que debes 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 te asustes y añade o quita cosas de éste para ajustarlo a tus necesidades. Una cosa que no debes 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 por Joey Hess.
# Algunas líneas son de debmake, por Christoph Lameter.
```

```

# Quítele el comentario para activar el modo explícito.
#export DH_VERBOSE=1

# Esta es la versión de compatibilidad con debhelper que usaremos.
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
```

```
51 dh_installexamples
52 dh_installmenu
53 # dh_installemacsen
54 # dh_installpam
55 # dh_installinit
56 dh_installcron
57 dh_installmanpages
58 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 estés familiarizado con líneas como la 1 de guiones hechos en shell o Perl. Esto significa que el fichero debe ejecutarse con /usr/bin/make.

Las líneas 12 a la 19 describen la regla ‘build’ (construcción, n. del t.), que ejecuta el Makefile del propio programa para compilarlo.

La regla ‘clean’ (limpiar, 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 *esté* limpio!), así que, por favor, usa las opciones que fuercen a hacer cosas (por ejemplo para rm, sería ‘–f’), o ignora 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 tu paquete es del tipo ‘Architecture: all’ necesitarás incluir todas las órdenes para crear el paquete bajo esta regla, y dejar la siguiente regla (‘binary–arch’) vacía en su lugar.

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

Los nombres comienzan 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,
- `dh_installmanpages(1)` copia todas las páginas de manual que puede encontrar en el paquete fuente en el paquete (cuidado, esto hace lo dice: *todas* las páginas),
- `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_installdeb(1)` copia los ficheros relativos al paquete (es decir, los guiones del mantenedor) 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 lee sus páginas de manual respectivas. Hay otros guiones con la misma nomenclatura (dh_*) que no se han mencionado aquí, pero pueden serte útiles. Si los necesitas, lee 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 sección binary–arch es donde deberías comentar las líneas que llamen a funciones que no necesites. En este caso he comentado las líneas sobre testversion, emacsen, pam, init, cron, manpages, info, undocumented, suidregister, makeshlibs, y 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.

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

Capítulo 5

Otros ficheros en el directorio debian/.

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

5.1 README.debian. (LÉEME.debian, n. del t.)

Cualquier detalle extra o discrepancias entre el programa original y su versión debianizada debería documentarse 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 borrarlo.

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 resuelve este problema marcando los ficheros de configuración de forma que cuando actualizas un paquete no se te pregunta si deseas mantener la nueva configuración o no. Se 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 tu programa no tiene ningún fichero de configuración.

5.3 dirs.

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

```
1 usr/bin
2 usr/sbin
```

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

```
1 usr/X11R6/bin
2 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 usas uno de ellos, edítalo y elimina la extensión .ex. Si no lo quieres usar, simplemente bórralo.

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

```
       Sección |      Descripción          |      Notas
          1     Comandos de Usuario          Programas o guiones ejecutables.
          2     Llamadas al Sistema          Funciones que ofrece el núcleo.
          3     Llamadas a Librerías         Funciones dadas por las libre-
rías del sistema.
          4     Ficheros Especiales          Generalmente se encuentran 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 suele eje-
cutar el superusuario.
          9     Rutinas del Núcleo           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. Como no había página de manual en las fuentes originales la escribí a partir 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 "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 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/see-menu-manual\
  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 menú y submenú 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 "none" 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 menú a ésta:

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

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

5.6 watch.ex.

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

Esto es lo que he puesto yo:

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

Pista: conéctate a Internet, e intenta ejecutar el programa "uscan" en el directorio donde has creado el fichero. Ve la página de manual para más detalles.

5.7 ex.doc–base

Si tu paquete tiene documentación en HTML o de otro tipo (excepto páginas de manual y documentos info), deberías 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, revisa `install-docs(8)` y el manual de doc–base en /usr/share/doc/doc–base/doc–base.html/index.html.

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

Estos ficheros se llaman guiones del desarrollador (maintainer scripts, n. del t.), y son guiones que se colocan en el área de control del paquete y que dpkg ejecuta cuando tu paquete se instala, se actualiza o se quita.

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

Ahora estamos preparados para construir el paquete.

Capítulo 6

Pasos Finales

6.1 Construir el paquete.

Entra en el directorio principal del programa y ejecuta el siguiente comando:

```
dpkg-buildpackage -rfakeroot
```

Esto hará todo por ti, sólo tendrás que dar tu clave secreta PGP dos veces. Una vez hayas hecho esto, verás cuatro nuevos ficheros en el directorio anterior (~/debian/):

- *gentoo_0.9.12–1_i386.deb*

  Este es el paquete binario completo. Puedes usar dpkg o dselect para instalar o eliminar tanto este 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 no está usando el sistema de paquetes de Debian, pero necesita descargarse las fuentes y compilarlo manualmente.

- *gentoo_0.9.12–1.dsc*

  Este es un sumario de los contenidos del código fuente. Este fichero se genera a partir del fichero gentoo–0.9.12/debian/control y se usa cuando se descomprimen las fuentes con `dpkg-source(1)`. Este fichero está firmado con PGP de forma que cualquiera 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". El que lo hace y lo usa es `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 a partir del fichero gentoo–0.9.12/debian/changelog y el fichero .dsc.

  Mientras sigues trabajando en el paquete, cambiará su comportamiento y se le añadirán nuevas funciones. Las personas que descarguen tu 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 comprobarlos con `md5sum(1)` y si los números no coinciden, sabrán que el fichero está corrupto o ha sido modificado. Este fichero está firmado con PGP de forma que cualquiera puede estar aún más seguro de que es realmente tuyo.

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

```
fakeroot debian/rules binary
```

Sólo asegúrate de que tu 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 hayas terminado con los cambios, no te olvides de reconstruir el paquete siguiendo el procedimiento habitual, para poder enviarlo de la forma correcta.

6.2 Comprobar tu paquete para encontrar errores.

Ejecuta `lintian(1)` sobre tu fichero de cambios .changes; este 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 tu paquete. Si parece que hay algunos errores (líneas que comienzan por E:), lee la explicación (líneas N:), corrige errores, y reconstruye como se describe en 'Construir el paquete.' en la página anterior. Si hay líneas que comienzan con W:, son sólo avisos (warnings, n. del t.), así que puedes estar seguro de que tu paquete está bien (pero seguramente necesita algún ajuste fino).

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

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

Instala el paquete para probarlo tú mismo, p. ej. usando la orden `debi(1)` como superusuario. Intenta instalarlo y ejecutarlo en otras máquinas distintas de la tuya y estáte atento para detectar errores o avisos tanto en la instalación como en la ejecución del programa.

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

- actualízate a partir de la versión previa (y de la versión en la última versión de Debian),
- vuelve a la versión anterior de nuevo,
- instala el paquete como uno nuevo (esto es, sin ninguna versión instalada previamente),
- desinstálalo, reinstálalo y luego púrgalo.

6.3 Enviando el paquete.

Ahora que has probado tu nuevo paquete en profundidad, necesitarás enviar estos ficheros a master.debian.org, usando `dupload(1)`. Primero debes editar el fichero de configuración de dupload, ~/.dupload.conf. Puedes 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, cambia mis valores personales a los tuyos, y lee la página de manual `dupload.conf(5)` para comprender qué significa cada una de estas opciones.

Entonces conecta con tu proveedor de Internet, y ejecuta la orden:

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

Dupload comprueba que las sumas md5 coinciden con aquellas en el fichero .changes, y te avisará de rehacer el paquete como se describe en 'Construir el paquete.' en la página 29 para poder enviarlo correctamente.

Dupload te preguntará tu contraseña en master.debian.org, enviará tus paquetes, y hará un pequeño anuncio sobre su envío en `<debian-devel-changes@lists.debian.org>` si es necesario.

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

6.4 Actualizar tu paquete.

Supongamos que se ha creado un informe de un fallo en tu paquete, de número #54321, y que describe un problema que puedes solucionar. Para crear una nueva revisión del paquete, necesitas:

- 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 tu paquete se acepte en el archivo de Debian.
- Repite lo que hiciste en 'Construir el paquete.' en la página 29, 'Comprobar tu paquete para encontrar errores.' en la página 30, y 'Enviando el paquete.' en la página anterior. La diferencia es que esta vez, las fuentes originales del archivo no se incluirán, dado que no han cambiado y ya existen en el archivo de Debian.

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

- Descarga las nuevas fuentes y pon el archivo tar (pongamos que se llama 'gentoo–0.9.13.tar.gz') un directorio por encima del antiguo árbol de fuentes (por ejemplo ~/debian/).
- Entra en el antiguo directorio y ejecuta:

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

  Por supuesto, reemplaza este nombre de fichero con el nombre de las fuentes de tu programa. `uupdate(1)` renombrará apropiadamente este fichero tar, intentará aplicar los cambios de tu fichero .diff.gz previo y actualizará el nuevo fichero debian/changelog.
- Cambia al directorio '../gentoo–0.9.13', el nuevo directorio fuente del paquete, y repite la operación que hiciste en 'Construir el paquete.' en la página 29, 'Comprobar tu paquete para encontrar errores.' en la página 30, y 'Enviando el paquete.' en la página anterior.

Observa que si has puesto el fichero 'debian/watch' como se describe en 'watch.ex.' en la página 27, puedes ejecutar `uscan(1)` para buscar automágicamente fuentes revisadas, descargarlas, y ejecutar uupdate.

6.5 Dónde pedir ayuda.

Antes de que te decidas a preguntar en lugares públicos, por favor, simplemente RTFM ("Lee El Jodido Manual", n. del t.). Esto incluye la 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 recibas un aviso de fallo (bug report, n. del t.) (sí, avisos de fallos ¡de verdad!) sabrás que es el momento de indagar en el Sistema de Seguimiento de Fallos de Debian (`http://www.debian.org/Bugs/`) y leer la documentación de allí, para poder tratar los informes de forma eficiente.

Si te unes a la lista de distribución de Mentores de Debian en `<debian-mentors@lists.debian.org>` podrás tener a desarrolladores de Debian con experiencia que te ayudarán con las preguntas que pueda tener. Puedes subscribirte 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, házlas en la lista de distribución de Desarrolladores de Debian en `<debian-devel@lists.debian.org>`. Puedes subscribirte a ella enviando un correo electrónico a `<debian-devel-request@lists.debian.org>` con la palabra 'subscribe' en el tema del mensaje. Si ya eres un desarrollador de Debian deberías 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 tu paquete, y si cometiste algún error crítico serás bombardeado por correos de centenares de usuarios furiosos de Debian... Sólo bromeaba :–)

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