

Guia del nou desenvolupador de Debian

Dret de reproducció © 1998-2002 Josip Rodin

Dret de reproducció © 2005-2015 Osamu Aoki

Dret de reproducció © 2010 Craig Small

Dret de reproducció © 2010 Raphaël Hertzog

This document may be used under the terms of the GNU General Public License version 2 or higher.

Aquest document s'ha escrit fent servir aquest dos documents com a exemples:

- Making a Debian Package (AKA the Debmake Manual), copyright © 1997 Jaldhar Vyas.
- The New-Maintainer's Debian Packaging Howto, copyright © 1997 Will Lowe.

COLLABORATORS

	<i>TITLE :</i> Guia del nou desenvolupador de Debian		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Josip Rodin, Osamu Aoki, i Innocent De Marchi	3 de juliol de 2020	
Traducció al català		3 de juliol de 2020	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Índex

1	Començant correctament.	1
1.1	Dinamisme social a Debian	1
1.2	Programes necessaris per treballar.	3
1.3	Documents necessaris per treballar.	4
1.4	On demanar ajuda.	4
2	Primers passos.	6
2.1	Pla de treball de la construcció de paquets Debian	6
2.2	Seleccionar el programa.	7
2.3	Aconseguir el programa i prova'l.	9
2.4	Sistemes de compilació simples	10
2.5	Sistemes de compilació populars	10
2.6	Nom del paquet i versió.	11
2.7	Configuració de quilt	12
2.8	Paquet Debian no nadiu inicial.	12
3	Modificar les fonts originals.	14
3.1	Configuració de quilt	14
3.2	Apedaçant el codi font.	14
3.3	La instal·lació dels arxius al seu destí	15
3.4	Diferències a les biblioteques.	17
4	Fitxers necessaris en el directori debian.	19
4.1	El fitxer control	19
4.2	El fitxer copyright	23
4.3	El fitxer changelog	24
4.4	El fitxer rules	25
4.4.1	Objectius del fitxer rules	25
4.4.2	Fitxer rules predeterminat.	26
4.4.3	Personalització del fitxer rules	29

5	Altres fitxers del directori debian.	32
5.1	README.Debian (LLEGEIX-ME.debian)	32
5.2	Fitxer compat.	33
5.3	Fitxer conffiles.	33
5.4	Fitxers <i>nom_del_paquet.cron.*</i>	33
5.5	Fitxer dirs.	34
5.6	Fitxer <i>nom_del_paquet.doc-base</i> .	34
5.7	Fitxer docs.	34
5.8	Fitxer emacs-en-*.	34
5.9	Fitxer <i>nom_del_paquet.examples</i> .	35
5.10	Fitxers <i>nom_del_paquet.init nom_del_paquet.default</i>	35
5.11	Fitxer install.	35
5.12	Fitxer <i>nom_del_paquet.info</i> .	35
5.13	Fitxer <i>nom_del_paquet.links</i> .	36
5.14	Fitxers { <i>nom_arxiu</i> .source/}lintian-overrides	36
5.15	Fitxers manpage.*.	36
5.15.1	Fitxers manpage.1.ex	36
5.15.2	Fitxer manpage.sgml.ex.	37
5.15.3	Fitxer manpage.xml.ex.	37
5.16	Fitxer <i>nom_del_paquet.manpages</i>	37
5.17	Fitxer NEWS.	37
5.18	Fitxers {pre,post}{inst,rm}	38
5.19	Fitxer <i>nom_del_paquet.symbols</i> .	38
5.20	Fitxer TODO.	38
5.21	Fitxer watch.	38
5.22	Fitxer source/format.	39
5.23	Fitxer source/local-options.	39
5.24	Fitxer source/options	39
5.25	Fitxers patches/*.	40
6	Construir el paquet.	41
6.1	Reconstrucció completa.	41
6.2	«Autobuilder».	42
6.3	L'ordre debuild .	43
6.4	El paquet pbuilder .	43
6.5	git-buildpackage command and similar	45
6.6	Reconstrucció ràpida.	46
6.7	Jerarquia d'ordres	46

7	Com comprovar el teu paquet per trobar errors.	47
7.1	Canvis sospitosos	47
7.2	Comprovació de la instal·lació del paquet	47
7.3	Comprovar els <i>guions del desenvolupador</i> («maintainer scripts»).	47
7.4	El paquet <code>lintian</code> .	48
7.5	L'ordre <code>debc</code> .	49
7.6	L'ordre <code>debdiff</code> .	49
7.7	L'ordre <code>interdiff</code> .	49
7.8	L'ordre <code>mc</code> .	49
8	Actualitzar el paquet.	50
8.1	Nova revisió Debian del paquet.	50
8.2	Inspecció d'una nova versió de l'autor.	51
8.3	Nova versió del programa font.	51
8.4	Actualitzar el format del paquet.	52
8.5	Conversió a UTF-8	53
8.6	Recordatori per actualitzar paquets.	53
9	Enviar el paquet.	54
9.1	Enviar el paquet al repositori de Debian.	54
9.2	Incloure el fitxer <code>orig.tar.gz</code> per a la transferència del paquet al repositori.	55
9.3	Enviaments discontinuats.	55
A	Tècniques avançades	56
A.1	Biblioteques compartides.	56
A.2	Gestionant <code>debian/nom_del_paquet.symbols</code>	57
A.3	Multi-arquitectura	58
A.4	Construint un paquet de biblioteca compartit	59
A.5	Paquet nadiu Debian	60

Capítol 1

Començant correctament.

Aquest document descriurà com es construeix un paquet Debian GNU/Linux per a un usuari comú Debian i per a futurs desenvolupadors, fent servir un llenguatge informal i incloent molts exemples. Un antic dit romà diu *Longum iter est per preaecepta, breve et efficax per exempla!* («És un llarg camí amb les normes, però curt i eficient amb els exemples»).

The rewrite of this tutorial document with updated contents and more practical examples is available as [Guide for Debian Maintainers](https://www.debian.org/doc/devel-manuals#debmake-doc) (<https://www.debian.org/doc/devel-manuals#debmake-doc>) . Please use this new tutorial as the primary tutorial document.

This document is made available for the Debian **Buster** release since this offers many translations. This document will be dropped in the following releases since contents are getting outdated. ¹

Una de les característiques que fan de Debian una de les distribucions més importants del mercat és el seu sistema de paquets. Tot i què hi ha una gran quantitat de programes disponibles en paquets de Debian, de vegades pots haver de fer servir programes que no estan disponibles en aquest format. Pot ésser que et demanis com construir els teus paquets i que pensis que és una tasca difícil. Bé doncs, si ets un principiant en GNU/Linux, pot ésser difícil, però si acabes de començar amb aquest sistema, no és recomanable que comencis per aquí :-). Cal saber algunes coses sobre programació en Unix, tot i què no és necessari ésser un mestre ².

Tot i així, una cosa és del tot segura: per construir i fer el manteniment de paquets Debian adequadament, calen moltes hores. Per a què el nostre sistema no tenguí errors, cal que els nostres desenvolupadors siguin tècnicament competents.

If you need some help with packaging, please read Secció 1.4.

Les noves versions d'aquest document es publiquen a <http://www.debian.org/doc/maint-guide/> i en el paquet `maint-guide`. Les traduccions es poden aconseguir en els paquets del tipus `maint-guide-ca`. Teniu en compte que les traduccions poden estar sense actualitzar.

Com es tracta d'un tutorial, he decidit explicar detalladament cada etapa dels temes importants. Alguns d'ells poden semblar-te irrelevants. Sigues pacient. També he omès intencionadament alguns casos extrems i sempre punters només per mantenir senzill aquest document.

1.1 Dinamisme social a Debian

Aquest és un recull de les meves observacions sobre la interacció social a Debian: espero que t'ajudin en la teva interacció amb Debian:

- Tots som voluntaris.

¹En el document s'assumeix que fas servir la versió `jessie`. Si vols fer servir aquest document amb versions anteriors (incloses sistemes Ubuntu i d'altres), cal que tenguís instal·lats (com a mínim) els paquets `dpkg` i `debhelper` .

²Podràs aprendre les operacions bàsiques dels sistemes Debian a [Debian Reference](http://www.debian.org/doc/manuals/debian-reference/) (<http://www.debian.org/doc/manuals/debian-reference/>) . També s'expliquen alguns aspectes de la programació en sistemes Unix.

- No podeu imposar als altres el que s’ha de fer.
- Cal estar motivat per fer les coses per tu mateix.
- La cooperació amistosa és la força motriu.
 - La teva contribució no ha de sobre-esforçar als altres.
 - La teva contribució és valuosa només quan els altres t’ho agraeixen.
- Debian no és la teva escola on automàticament reps atenció dels docents.
 - Cal ésser capaç d’aprendre moltes coses per un mateix.
 - L’atenció d’altres voluntaris és un recurs molt escàs.
- Debian millora contínuament.
 - S’espera que construeixis paquets d’alta qualitat.
 - Cal adaptar-se al canvi.

En el desenvolupament i manteniment de Debian, es fan servir les següents expressions per referir-se a les persones implicades en la construcció i manteniment dels programes i paquets:

- **autor original** («upstream author»): és la persona que ha escrit el codi original del programa (o l’original del document en el cas de paquets de documentació).
- **desenvolupador original** («upstream maintainer»): la persona actualment responsable del manteniment del codi original.
- **empaquetador (desenvolupador)** («maintainer»): la persona encarregada de la construcció i actualització dels paquets per a Debian.
- **patrocinador** («sponsor»): la persona (cal que sigui un DD o DM) que transfereix els paquets construïts pels desenvolupadors al repositori de Debian després de comprovar que el paquet s’ajusta a las normes de Debian.
- **mentor**: la persona que ajuda als desenvolupadors principiants a iniciar-se en la construcció i manteniment del paquet.
- **desenvolupador de Debian (DD)** («Debian Developer»): la persona que es membre de Debian. Té autorització (sense limitacions) per transferir paquets al repositori oficial de Debian.
- **empaquetador/responsable de Debian (DM)**(«Debian Maintainer»): la persona amb autorització limitada per transferir paquets al repositori oficial de Debian.

No és possible arribar a ésser **desenvolupador oficial de Debian (DD)** de la nit al dia degut a què no és suficient ésser tècnicament competent. Cal no desanimar-se per això. Pots aconseguir que el teu paquet passi al repositori, si és d’utilitat als usuaris, com a **responsable** mitjançant un **patrocinador** o un **responsable de Debian**.

No és necessari construir un nou paquet per ésser desenvolupador oficial de Debian. Una opció és contribuir en el manteniment d’algun dels paquets de la distribució. Hi ha molts paquets pendents d’algú que s’en faci càrrec (consulta Secció 2.2).

Com que ens concentrem només en els aspectes tècnics dels paquets en el present document, consulta el següent per saber com funciona Debian i com pots participar-hi:

- **Debian: 17 years of Free Software, "do-ocracy", and democracy** (<http://upsilon.cc/~zack/talks/2011/20110321-taipei.pdf>) (PDF amb diapositives d’introducció en anglès)
- **How can you help Debian?** (<http://www.debian.org/intro/help>) (document oficial)
- **The Debian GNU/Linux FAQ, Chapter 13 - 'Contributing to the Debian Project'** (<https://www.debian.org/doc/manuals/debian-faq/contributing.en.html>) (document semi-oficial)
- **Debian Wiki, HelpDebian** (<http://wiki.debian.org/HelpDebian>) (documentació complementària)
- **Debian New Member site** (<https://nm.debian.org/>) (document oficial)
- **Debian Mentors FAQ** (<http://wiki.debian.org/DebianMentorsFaq>) (documentació complementària)

1.2 Programes necessaris per treballar.

Abans de començar, cal instal·lar els paquets necessaris per a la construcció de paquets. A la llista de paquets no estan inclosos paquets amb prioritat «essencial» o «requerit» (`essential` i `required`) degut a què segurament ja estan instal·lats.

The following packages come with the standard Debian installation, so you probably have them already (along with any additional packages they depend on). Still, you should check them with `aptitude show package` or with `dpkg -s package`.

El paquet imprescindible per al desenvolupament és `build-essential`. Quan s'instal·la, *també s'instal·len* altres paquets necessaris, obtenint-se la instal·lació bàsica per a la construcció de paquets.

Per a la construcció d'alguns paquets, amb això seria suficient, però n'hi ha d'altres que, sense ésser imprescindibles, pot ésser útil instal·lar-los o, fins i tot, necessaris:

- `autoconf`, `automake` i `autotools-dev` - molts programes fan servir fitxers de configuració i fitxers `Makefile` que es gestionen amb aquests programes (llegeix `info autoconf`, `info automake`). El paquet `autotools-dev` permet mantenir versions actualitzades dels fitxers «autoconf» i «automake» i conté documentació per aprendre a fer servir amb eficàcia aquest tipus de fitxers.
- `dh-make` i `debhelper` - `dh-make` es fa servir en la construcció de l'esquelet dels paquets, i es faran servir algunes eines de `debhelper` per construir alguns paquets. Tot i que no són imprescindibles per a la construcció de paquets és del *tot* recomanable per als nous desenvolupadors. Faciliten molt el procés de construcció inicial així com el manteniment posterior (llegiu `dh_make(8)`, `debhelper(1)`, `/usr/share/doc/debhelper/README`)³.
- La nova ordre `debmake` es pot fer servir com a alternativa a l'ordre `dh-make`. Compte amb més funcionalitats i una documentació extensa en format HTML amb exemples per a la construcció de paquets a `debmake-doc`.
- `devscripts` - aquest paquet conté alguns guions útils per als desenvolupadors, però no són necessaris en la construcció de paquets. Val la pena mirar els paquets recomanats i suggerits per aquest paquet (llegeix `/usr/share/doc/devscripts/README.gz`).
- `fakeroot` - this utility lets you emulate being root, which is necessary for some parts of the build process. (See `fakeroot(1)`.)
- `file` - aquest programa és molt útil per saber de quin tipus és un fitxer determinat (consulta `file(1)`).
- `gfortran` - el compilador GNU per a Fortran 95, necessari si el programa està escrit en Fortran (llegeix `gfortran(1)`).
- `git` - aquest paquet instal·la el popular sistema de control de versions, dissenyat per al seguiment de grans projectes amb eficàcia i rapidesa; es fa servir amb molts projectes de codi lliure importants entre els quals el nucli de Linux (llegeix `git(1)`, `git Manual` (`/usr/share/doc/git-doc/index.html`)).
- `gnupg` - a tool that enables you to digitally *sign* packages. This is especially important if you want to distribute packages to other people, and you will certainly be doing that when your work gets included in the Debian distribution. (See `gpg(1)`.)
- `gpc` - el compilador GNU de Pascal, necessari si el programa fa servir aquest llenguatge. També hi ha el compilador `fp-compiler`, de codi lliure per a Pascal (llegeix `gpc(1)` i `ppc386(1)`).
- `lintian` - this is the Debian package checker, which lets you know of any common mistakes after you build the package and explains the errors found. (See `lintian(1)`, *Lintian User's Manual* (<https://lintian.debian.org/manual/index.html>) .)
- `patch` - és una utilitat molt útil. Permet aplicar un arxiu amb una llista de diferències (produït pel programa `diff`) en el fitxer original, i obtenir una versió apedaçada del fitxer original (vegeu `patch(1)`).
- `patchutils` - aquest paquet proporciona programes per gestionar els pegats, entre d'altres `lsdiff`, `interdiff` i `filterdiff`.
- `pbuilder` - this package contains programs which are used for creating and maintaining a **chroot** environment. Building a Debian package in this **chroot** environment verifies the proper build dependency and avoids FTBFS (Fails To Build From Source) bugs. (see `pbuilder(8)` and `pbuild(1)`)
- `perl` - Perl és un dels llenguatges interpretats usats amb més freqüència per als guions (o «scripts») en els sistemes Un*x. Tant és així que es fa servir l'expressió «navalla suïssa d'Unix» per fer referència en aquest llenguatge (llegeix `perl(1)`).

³Hi ha d'altres paquets semblants pel que fa a la seva funcionalitat però més específics, com és ara `dh-make-perl`, `dh-make-php`, etc.

- **python** - Python és una altre llenguatge interpretat per a fer guions a Debian degut a la combinació de funcionalitat i clara sintaxi (consulta `python(1)`).
- **quilt** - aquest paquet permet aplicar els pegats i fer el seguiment dels canvis realitzats. Fa les modificacions segons l'ordre establert i, amb ell, és possible aplicar («push»), desfer («pop») i actualitzar les modificacions fàcilment (llegeix `quilt(1)` i `/usr/share/doc/quilt/quilt.pdf.gz`).
- **xutils-dev** - alguns programes, si més no els escrits per a X11, fan servir aquest paquet per generar fitxers `Makefile` executant conjunts de funcions macro (llegeix `imake(1)` i `xmkmf(1)`).

Les descripcions anteriors només són un resum de la funcionalitat de cada paquet. Abans de continuar, si us plau, llegeix la documentació de cada programa, al menys pel que fa a la seva funcionalitat bàsica. Pot semblar una mica pesat, però et farà *molt profit* aquesta lectura.

1.3 Documents necessaris per treballar.

La documentació llistada a continuació és *imprescindible* llegir-la mentre es llegeix aquest document:

- **debian-policy** - a [Debian Policy Manual](http://www.debian.org/doc/devel-manuals#policy) (<http://www.debian.org/doc/devel-manuals#policy>) s'explica l'estructura i contingut del repositori, alguns aspectes del disseny del sistema operatiu, l'estàndard del sistema de fitxers («Filesystem Hierarchy Standard») i el què és més important, descriu els requisits per a què un paquet s'afegeixi a la distribució (llegix les còpies locals de `/usr/share/doc/debian-policy/policy.pdf.gz` i `/usr/share/doc/debian-policy/fhs/fhs-3.0.pdf.gz`).
- **developers-reference** - the [Debian Developer's Reference](http://www.debian.org/doc/devel-manuals#devref) (<http://www.debian.org/doc/devel-manuals#devref>) descriu totes les coses no específicament sobre els detalls tècnics d'empaquetament, com la estructura de l'arxiv, com canviar el nom, orfà, o adoptar paquets, com fer NMUs, com gestionar bugs, les millors pràctiques d'empaquetament, quan i on pujar, etc. (Vegeu la còpia local de `/usr/share/doc/developers-reference/developers-reference.pdf`.)

La documentació següent és *important* llegir-la mentre es llegeix aquest document:

- [Autotools Tutorial](http://www.lrde.epita.fr/~adl/autotools.html) (<http://www.lrde.epita.fr/~adl/autotools.html>) provides a very good tutorial for the GNU Build System known as the GNU Autotools, whose most important components are Autoconf, Automake, Libtool, and gettext.
- **gnu-standards** - aquest paquet proporciona dos documents del projecte GNU: «GNU Coding Standards» (http://www.gnu.org/prep/standards/html_node/index.html) i «Information for Maintainers of GNU Software» (http://www.gnu.org/prep/maintain/html_node/index.html). Encara que no són d'aplicació obligada a Debian, són útils com a orientació (llegeix les còpies locals de `/usr/share/doc/gnu-standards/standards.pdf.gz` i `/usr/share/doc/gnu-standards/maintain.pdf.gz`).

Si aquest document és contradictori amb algun aspecte amb els documents anteriors, serà d'aplicació l'establert per aquest darrers. En aquest cas, si us plau comunica l'error del paquet `maint-guide` fent servir **reportbug**.

The following is an alternative tutorial document that you may read along with this document:

- [Debian Packaging Tutorial](http://www.debian.org/doc/packaging-manuals/packaging-tutorial/packaging-tutorial) (<http://www.debian.org/doc/packaging-manuals/packaging-tutorial/packaging-tutorial>)

1.4 On demanar ajuda.

Before you decide to ask your question in some public place, please read this fine documentation:

- els arxius de `/usr/share/doc/package` per a cada paquet
- el contingut de **man** *command* per a cada una de les ordres

- el contingut de **info** *command* per a cada una de les ordres
- els continguts de l'arxiu de la llista de correu debian-mentors@lists.debian.org (<http://lists.debian.org/debian-mentors/>)
- el contingut de l'arxiu de la llista de correu debian-devel@lists.debian.org (<http://lists.debian.org/debian-devel/>)

Considera fer servir el motor de cerca web mitjançant la inclusió eficaç de cadenes de recerca, com ara `site:lists.debian.org` per limitar el domini.

Començar a treballar en un paquet petit és una bona opció per aprendre els detalls de la creació de paquets. Inspeccionar els paquets mantinguts per altres persones és una excel·lent manera d'aprendre.

Si encara tens preguntes sobre la construcció de paquets que no poden trobar resposta en la documentació disponible i els recursos web, pots fer consultes de forma interactiva:

- debian-mentors@lists.debian.org mailing list (<http://lists.debian.org/debian-mentors/>) . (Aquesta llista de correu és per a principiants.)
- llista de correu debian-devel@lists.debian.org (<http://lists.debian.org/debian-devel/>) . (Aquesta llista de correu és per a experts.)
- IRC (<http://www.debian.org/support#irc>) con és ara `#debian-mentors`.
- Equips centrats en un conjunt de paquets específic. (La llista completa està disponible a <https://wiki.debian.org/Teams> (<https://wiki.debian.org/Teams>))
- Llistes de correu específiques per a un idioma com per exemple `debian-devel-{french,italian,portuguese,spanish}@lists.debian.org` o `debian-devel@debian.or.jp`. (La llista completa està disponible a <https://lists.debian.org/devel.html> (<https://lists.debian.org/devel.html>) i <https://lists.debian.org/users.html> (<https://lists.debian.org/users.html>))

Els desenvolupadors de Debian amb més experiència t'ajudaran amb gust, si fas les preguntes després de fer el treball necessari.

En rebre un avís d'error (sí, un avís d'error de veritat!) serà el moment de donar un cop d'ull al [Sistema de seguiment d'errades de Debian](http://www.debian.org/Bugs/) (<http://www.debian.org/Bugs/>) i llegir la documentació per tal de conèixer el procediment adequat. Et recomano la lectura de la *Developer's Reference*, 5.8. 'Handling bugs' (<http://www.debian.org/doc/manuals/developers-reference/pkgs.html#bug-handling>) .

Even if it all worked well, it's time to start praying. Why? Because in just a few hours (or days) users from all around the world will start to use your package, and if you made some critical error you'll get mailbombed by numerous angry Debian users... Just kidding. :-)

Relaxa't i estiguis preparat per rebre informes d'errades, perquè el camí és llarg per aconseguir el compliment de las Normes de Debian (una vegada més llegeix la *documentació real* per a més detalls). Bona sort!

Capítol 2

Primers passos.

Començarem la construcció del teu paquet (o encara millor, pots adoptar un paquet ja existent).

2.1 Pla de treball de la construcció de paquets Debian

Si estàs treballant en la construcció d'un paquet Debian a partir de les fonts d'un programa, el pla de treball típic requereix la construcció d'arxius amb noms específics a cada una de les etapes de la següent manera:

- Obtindre una còpia del codi font del programa, normalment en un arxiu comprimit en format «tar».
– `nom_del_paquet-versió.tar.gz`
- Afegir les modificacions específiques de la construcció del paquet Debian a les fonts del programa en el directori `debian`, i generar un paquet font no nadiu (o sigui, el conjunt d'arxius d'entrada que es fan servir en la compilació del paquet Debian).
– `nom_del_paquet-versió.orig.tar.gz`
– `nom_del_paquet-versió-revisió.debian.tar.gz`¹
– `nom_del_paquet-versió-revisió.dsc`
- Construir paquets binaris Debian, habitualment paquets ordinaris per a la instal·lació en format `.deb` (o en format `.udeb`, fet servir per l'instal·lador de Debian) a partir del paquet font Debian.
– `nom_del_paquet-versió-revisió_arquitectura.deb`

Observa que el caràcter que separa `nom_del_paquet` i `versió` s'ha canviat de `-` (guió) a `_` (guió baix) en el nom del paquet Debian.

En els noms dels arxius anteriors, cal substituir `nom_del_paquet` pel **nom del paquet**, `versió` pel codi de **versió del codi font**, `revisió` pel codi de la **revisió Debian** i `arquitectura` per l'**arquitectura del paquet** com es defineix en el Manual de Normes de Debian ².

Cada etapa d'aquest esquema s'explica amb detall en seccions posteriors.

¹Per a paquets no nadius Debian construïts amb l'antic format 1.0, es fa servir `nom_del_paquet-versió-revisió.diff.gz`

²Consulta 5.6.1 "Source" (<http://www.debian.org/doc/debian-policy/ch-controlfields.html#s-f-Source>) , 5.6.7 "Package" (<http://www.debian.org/doc/debian-policy/ch-controlfields.html#s-f-Package>) , i 5.6.12 "Version" (<http://www.debian.org/doc/debian-policy/ch-controlfields.html#s-f-Version>) . El codi de l'**arquitectura del paquet** segueix el [Debian Policy Manual](#), 5.6.8 "Architecture" (<http://www.debian.org/doc/debian-policy/ch-controlfields.html#s-f-Architecture>) i s'assigna automàticament en el procés de construcció del paquet.

2.2 Seleccionar el programa.

Probablement ja tens clar quin paquet vols construir. Primer cal que comprovis si ja està inclòs en el repositori fent servir:

- l'ordre **aptitude**.
- la pàgina web **paquets Debian** (<http://www.debian.org/distrib/packages>) .
- pàgina web **Debian Package Tracker** (<https://tracker.debian.org/>) (Sistema de seguiment de paquets de Debian)

Si el paquet ja existeix, comença per instal·lar-ho. Si és un paquet **orfe** (si el responsable actual és el «Debian QA Group» (<http://qa.debian.org/>) , el grup de qualitat de Debian), pots adoptar-lo (convertir-te en el responsable del manteniment) si està disponible. Cal que ho comprovis a «Debian Bug report logs»: **errors en el paquet «wnpp» de la distribució de treball («inestable» o «sid»)** (<http://bugs.debian.org/wnpp>) . També pots adoptar un paquet si hi ha un avís de «sol·licitud d'adopció» («Request for Adoption» o **RFA**)³.

Hi ha diversos recursos de seguiment dels paquets:

- L'ordre **wnpp-alert** del paquet **devscripts**
- **Work-Needing and Prospective Packages** (<http://www.debian.org/devel/wnpp/>) (Paquets en què es treballa i futurs paquets)
- **Registre Debian d'Informes d'errors: errors en el pseudo-paquet wnpp a unstable** (<http://bugs.debian.org/wnpp>)
- **Paquets Debian que necessiten atenció** (<http://wnpp.debian.net/>)
- **Navegació en els errors del paquet wnpp basada en paraules clau** (<http://wnpp-by-tags.debian.net/>)

Cal tenir present que Debian incorpora paquets d'un gran nombre de programes de tot tipus i que la quantitat de paquets disponibles en el repositori de Debian és major que el nombre de col·laboradors amb autorització per afegir paquets al repositori. En conseqüència, la col·laboració en el manteniment de paquets que ja estan en el repositori es valora molt positivament (i és més fàcil trobar un patrocinador) per la resta de desenvolupadors⁴. Per col·laborar en el manteniment de paquets ja disponibles en el repositori tens les següents opcions:

- Fer-te càrrec de paquets orfes d'ús freqüent
- unir-te als **equips de desenvolupadors** (<http://wiki.debian.org/Teams>) .
- Selecciona errors dels paquets més populars.
- Selecciona **paquets QA o NMU** (<http://www.debian.org/doc/developers-reference/pkgs.html#nmu-qa-upload>) .

If you are able to adopt the package, get the sources (with something like `apt-get source packagename`) and examine them. This document unfortunately doesn't include comprehensive information about adopting packages. Thankfully you shouldn't have a hard time figuring out how the package works since someone has already done the initial setup for you. Keep reading, though; a lot of the advice below will still be applicable to your case.

Si el paquet és nou i decideixes que t'agradaria posar-lo a disposició dels usuaris de Debian, has de seguir els passos indicats a continuació:

- Aprèn el funcionament del programa i fes-ho servir una temporada (per comprovar la seva utilitat).
- You must check that no one else is already working on the package on the **Work-Needing and Prospective Packages** (<http://www.debian.org/devel/wnpp/>) site. If no one else is working on it, file an ITP (Intent To Package) bug report to the **wnpp** pseudo-package using **reportbug**. If someone's already on it, contact them if you feel you need to. If not —find another interesting program that nobody is maintaining.

³See **Debian Developer's Reference 5.9.5 "Adopting a package"** (<http://www.debian.org/doc/manuals/developers-reference/pkgs.html#adopting>) .

⁴Tot i així, hi ha nous programes que són d'interès per a Debian.

- Cal que el programa **tengui una llicència**:
 - Per a paquets de la secció `main` cal que el programa **s'ajusti a les Directrius de Debian per al programari (DFSG)** lliure (consulta **DFSG** (http://www.debian.org/social_contract#guidelines)) i **no ha de precisar la instal·lació de cap altre paquet** que no pertanyi a la secció `main` en la compilació o execució com requereix la directiva de Debian («Debian Policy»).
 - Per a paquets de la secció `contrib`, la llicència cal que compleixi tots els requisits de la DFSG però pot precisar la instal·lació d'altres paquets que no siguin de la secció `main` en la compilació o execució.
 - Per a paquets de la secció `non-free`, no és necessari que la llicència compleixi tots els requisits de la DFSG però **cal que permeti la distribució del programa**.
 - Si tens dubtes a l'hora d'assignar el paquet a una secció, envia un correu amb el text de la llicència (en anglès) adreçat a debian-legal@lists.debian.org (<http://lists.debian.org/debian-legal/>).
- The program should **not** introduce security and maintenance concerns into the Debian system.
 - The program should be well documented and its code needs to be understandable (i.e., not obfuscated).
 - Cal que contactis amb l'autor o autors del programa per comprovar que accepten la construcció del paquet. És important que els autors continuïn amb el manteniment del programa de manera que puguis fer consultes sobre problemes específics del programa. No comencis a empaquetar un programa que no sigui mantingut pels autors.
 - El programa **no** s'ha d'executar com a «setuid root»: cal que no sigui «setuid» ni «setgid».
 - Cal que el programa no sigui un dimoni, o que s'instal·li en els directoris `*/sbin` o obrir un port com a usuari administrador.

Of course, the last one is just a safety measure, and is intended to save you from enraging users if you do something wrong in some setuid daemon... When you gain more experience in packaging, you'll be able to package such software.

Com a nou desenvolupador, s'aconsella l'adquisició d'experiència amb la construcció de paquets senzills i no s'aconsella la construcció de paquets complicats.

- Paquets simples
 - un únic paquet binari, arquitectura = totes (col·lecció de dades com per exemple gràfics de fons de pantalla)
 - un únic paquet binari, arquitectura = totes (executables escrits en un llenguatge interpretat com per exemple POSIX)
- Paquets de complexitat intermèdia
 - un únic paquet binari, arquitectura = totes (executables binaris ELF escrits en un llenguatge compilat com és ara C i C++)
 - paquet múltiple binari, arquitectura = qualsevol i totes (paquets amb executables binaris ELF i documentació)
 - el format de l'arxiu font no és ni `tar.gz` ni `tar.bz2`
 - paquets amb parts de les fonts que no es poden distribuir.
- Paquets molt complexes
 - paquets amb mòduls d'interprets fets servir per altres paquets
 - paquets de biblioteques genèriques ELF que fan servir altres paquets
 - paquets amb múltiples binaris incloent paquets de biblioteques ELF
 - paquets de fonts amb fonts originals diverses
 - paquets amb mòduls del nucli
 - paquets amb pegats del nucli.
 - qualsevol paquet amb guions del desenvolupador no trivials

Construir paquets molt complexes no és molt difícil, però requereix tenir més coneixements. Hauràs de buscar una orientació específica per a cada funció complexa. Per exemple, alguns llenguatges tenen els seus propis documents de normes:

- Perl policy (<http://www.debian.org/doc/packaging-manuals/perl-policy/>)

- Normes per a Python (<http://www.debian.org/doc/packaging-manuals/python-policy/>)
- Normes per a Java (<http://www.debian.org/doc/packaging-manuals/java-policy/>)

There is another old Latin saying: *fabricando fit faber* (practice makes perfect). It is *highly* recommended to practice and experiment with all the steps of Debian packaging with simple packages while reading this tutorial. A trivial upstream tarball, `hello-sh-1.0.tar.gz`, created as follows may offer a good starting point:⁵

```
$ mkdir -p hello-sh/hello-sh-1.0; cd hello-sh/hello-sh-1.0
$ cat > hello <<EOF
#!/bin/sh
# (C) 2011 Foo Bar, GPL2+
echo "Hello!"
EOF
$ chmod 755 hello
$ cd ..
$ tar -cvzf hello-sh-1.0.tar.gz hello-sh-1.0
```

2.3 Aconsegueix el programa i prova'l.

La primera passa és trobar i descarregar el codi font original del programa. Les fonts dels programes lliures de GNU/Linux és habitual que es distribueixin en fitxers amb format **tar+gzip** amb extensió `.tar.gz` o amb format **tar+bzip2** amb extensió `.tar.bz2`, i generalment tenen un directori amb el nom *programa-versió* amb tots els fitxers del codi font.

If the latest version of the source is available through a Version Control System (VCS) such as Git, Subversion, or CVS, you need to get it with `git clone`, `svn co`, or `cvcs co` and repack it into **tar+gzip** format yourself by using the `--exclude-vcs` option.

Si el programa està disponible en una altre format (pot ésser un arxiu acabat en `.Z` o `.zip`⁶), ho descomprimeixes amb l'eina adequada i el tornes a empaquetar adequadament.

Si algun dels continguts del codi font del teu programa no compleix les directrius DFSG, has de descomprimir-ho per eliminar-los hi tornar a comprimir-los afegint `dfsg` a la cadena de la versió del codi original.

Com exemple, faré servir el programa **gentoo**, un gestor de fitxers de X11 fet amb GTK+⁷.

Fes un nou directori al teu directori d'usuari amb el nom **debian** o **deb** o el que trobis més adequat (p.e. `~/gentoo/` és molt adequat). Copia en aquest nou directori el fitxer del codi font i extreu el seu contingut executant `tar xzf gentoo-0.9.12.tar.gz`. Comprova que no hi ha hagut errors, fins i tot els més *irrellevants*, degut a què és possible que hi hagi errors en desempaquetar-ho en sistemes d'altres persones que no ignorin aquestes errades. En el terminal hauràs fet el següent:

```
$ mkdir ~/gentoo ; cd ~/gentoo
$ wget http://www.example.org/gentoo-0.9.12.tar.gz
$ tar xvzf gentoo-0.9.12.tar.gz
$ ls -F
gentoo-0.9.12/
gentoo-0.9.12.tar.gz
```

Ara tens un nou directori amb el nom **gentoo-0.9.12**. Accedeix en aquest directori i llegeix *atentament* la documentació del programa, normalment en fitxers amb el nom `README*`, `INSTALL*`, `*.lsm` o `*.html`. Hi haurà instruccions per a la compilació i instal·lació del programa (probablement la instal·lació es farà a `/usr/local/bin`, però veurem a Secció 3.3 que no és la destinació correcta).

Per començar la construcció del paquet, cal fer-ho amb el directori del codi font «net», o simplement només amb els fitxers de l'arxiu comprimit del codi font.

⁵No et preocupis per l'arxiu `Makefile` perdut. Pots instal·lar l'ordre **hello** fent servir l'ordre **debhelper** com s'ha explicat a Secció 5.11 o modificant les fonts originals afegint un nou `Makefile` amb l'objectiu `install` com a Capítol 3.

⁶Pots identificar el format de l'arxiu fent servir l'ordre **file**.

⁷Aquest programa ja està empaquetat. La [versió actual](http://packages.qa.debian.org/g/gentoo.html) (<http://packages.qa.debian.org/g/gentoo.html>) fa servir «Autotools» en la seva construcció i ha canviat molt respecte a la versió 0.9.12 feta servir en els exemples.

2.4 Sistemes de compilació simples

Alguns programes tenen un arxiu `Makefile` i és possible compilar-los simplement executant `make`⁸. Molts d'ells permeten executar `make check`, per fer comprovacions. La instal·lació en el directori de destí es fa executant `make install`.

Una vegada s'ha compilat el programa, prova'l i comprova que tot funciona correctament i que no es genera cap errada en el sistema en l'execució i/o instal·lació.

L'execució de l'ordre `make clean` (o `make distclean`) eliminarà del directori els fitxers generats per a la compilació i que són innecessaris. Habitualment, serà possible des-instal·lar el programa amb l'ordre `make uninstall`.

2.5 Sistemes de compilació populars

La majoria de programes lliures estan escrits en llenguatge `C` i `C++`. Molts fan servir les «Autotools» i «CMake» per compilar en diferents arquitectures. Aquestes eines generen un arxiu `Makefile` i d'altres fitxers necessaris per a la compilació. Així, molts programes es compilen i instal·len amb l'execució de `make; make install`.

Les **Autotools** són el sistema de compilació GNU que comprèn [Autoconf](#), [Automake](#) (si l'entrada no existeix a la Viquipèdia, s'ha posat l'enllaç a la versió en castellà), [Libtool](#) i [gettext](#). Confirmaràs que el programa fa servir les «autotools» per la presència dels arxius `configure.ac`, `Makefile.am`, i `Makefile.in`⁹.

La primera etapa en l'ús de les «Autotools» és l'execució de l'ordre (per l'autor del codi font) `autoreconf -i -f` amb la qual es generen, fent servir el fitxer font (a l'esquerra del diagrama) els fitxers (a la dreta del diagrama) que després farà servir l'ordre «configure».

```
configure.ac-----+> autoreconf --> configure
Makefile.am -----+      |      +> Makefile.in
src/Makefile.am --      |      +> src/Makefile.in
                        |      +> config.h.in
                        |
                        |      automake
                        |      aclocal
                        |      aclocal.m4
                        |      autoheader
```

Per a l'edició dels fitxers `configure.ac` i `Makefile.am` cal conèixer el funcionament de **autoconf** i **automake**. Consulta `info autoconf` i `info automake` (executant les ordres en el terminal).

La segona etapa del pla de treball amb «Autotools» és l'obtenció de les fonts i l'execució de `./configure && make` en el directori del codi font per compilar el programa generant el fitxer **binari**.

```
Makefile.in -----+      +> Makefile -----+> make -> binary
src/Makefile.in --> ./configure --> src/Makefile --+
config.h.in -----+      +> config.h -----+
                        |
                        |      config.status --+
                        |      config.guess --+
```

Pots fer canvis en el fitxer `Makefile` com és el directori d'instal·lació predeterminat fent servir les opcions `./configure --prefix=/usr`.

Tot i què no és necessari, l'actualització del fitxer `configure` i dels altres fitxers amb l'ordre `autoreconf -i -f` és la forma més adient per comprovar la compatibilitat del codi font¹⁰.

CMake (en el moment de fer aquesta traducció, aquesta entrada no existeix a la Viquipèdia en català) és un sistema de compilació alternatiu. Els programes que fan servir aquest sistema de compilació tenen un arxiu `CMakeLists.txt` en el codi font.

⁸Many modern programs come with a script named `configure`, which when executed creates a `Makefile` customized for your system.

⁹Autotools is too big to deal with in this small tutorial. This section is meant to provide keywords and references only. Please make sure to read the [Autotools Tutorial](#) (<http://www.lrde.epita.fr/~adl/autotools.html>) and the local copy of `/usr/share/doc/autotools-dev/README.Debian.gz`, if you need to use it.

¹⁰Pots automatitzar el procés utilitzant `dh-autoreconf`. Consulta Secció 4.4.3.

2.6 Nom del paquet i versió.

Si el codi font te el nom `gentoo-0.9.12.tar.gz`, pots fer servir `gentoo` com a **nom del paquet** (de les fonts) i `0.9.12` com a codi de la **versió del codi font**. Aquestes denominacions es fan servir a l'arxiu `debian/changelog` que es descriu més endavant a Secció 4.3

Although this simple approach works most of the time, you may need to adjust **package name** and **upstream version** by renaming the upstream source to follow Debian Policy and existing convention.

You must choose the **package name** to consist only of lower case letters (a-z), digits (0-9), plus (+) and minus (-) signs, and periods (.). It must be at least two characters long, must start with an alphanumeric character, and must not be the same as existing packages. It is a good idea to keep its length within 30 characters.¹¹

Si l'autor original fa servir termes genèrics com `prova` de nom, és convenient canviar el nom de forma que identifiqui el seu contingut i evitar «embrutar» l'espai de noms¹².

You should choose the **upstream version** to consist only of alphanumerics (0-9A-Za-z), plus signs (+), tildes (~), and periods (.). It must start with a digit (0-9).¹³ It is good idea to keep its length within 8 characters if possible.¹⁴

If upstream does not use a normal versioning scheme such as `2.30.32` but uses some kind of date such as `11Apr29`, a random codename string, or a VCS hash value as part of the version, make sure to remove them from the **upstream version**. Such information can be recorded in the `debian/changelog` file. If you need to invent a version string, use the `YYYYMMDD` format such as `20110429` as upstream version. This ensures that `dpkg` interprets later versions correctly as upgrades. If you need to ensure smooth transition to the normal version scheme such as `0.1` in the future, use the `0~YYMMDD` format such as `0~110429` as the upstream version.

El codi de versió¹⁵ serà comparat per `dpkg(1)` de la següent manera:

```
$ dpkg --compare-versions versió_1 op versió_2
```

La norma de comparació de versions es pot resumir com:

- Les cadenes es comparen des del cap fins la cua.
- Les lletres són anteriors als dígit.
- Els nombres es comparen com enters.
- Les lletres es comparen per l'ordre del seu codi ASCII.
- Hi ha normes especials per al punt (.), signe més (+) i ~ com es mostra a continuació:

$$0.0 < 0.5 < 0.10 < 0.99 < 1 < 1.0\sim rc1 < 1.0 < 1.0+b1 < 1.0+nmu1 < 1.1 < 2.0$$

Un cas complicat es produeix quan s'allibera una versió del codi font `gentoo-0.9.12-ReleaseCandidate-99.tar.gz` com a versió prèvia de `gentoo-0.9.12.tar.gz`. Cal assegurar que l'actualització funcionarà correctament canviant el nom de codi font per `gentoo-0.9.12~rc99.tar.gz`.

¹¹La longitud predeterminada pel nom del paquet de l'ordre **aptitude** és 30. Per a més del 90% dels paquets, el nom del paquet té menys de 24 caràcters.

¹²If you follow the [Debian Developer's Reference 5.1. "New packages"](http://www.debian.org/doc/developers-reference/pkgs.html#newpackage) (<http://www.debian.org/doc/developers-reference/pkgs.html#newpackage>), the ITP process will usually catch this kind of issue.

¹³Aquesta norma més estricta hauria d'ajudar a evitar noms d'arxius confusos.

¹⁴La longitud predeterminada pel codi de versió de l'ordre **aptitude** és 10. El codi de revisió Debian precedit per un guió en consumeix 2. Per a més del 80% dels paquets, la versió del codi font té una longitud inferior a 8 caràcters i el codi de la revisió Debian menys de 2. Per a més del 90% dels paquets, la versió del codi font té menys de 10 caràcters i la revisió Debian menys de 3.

¹⁵Les cadenes de versió poden ésser la **versió del codi font** (*versió*), la **revisió Debian** (*revisió*), o **versió** (*versió-revisió*). Consulta Secció 8.1 per saber com cal incrementar la **revisió Debian**.

2.7 Configuració de quilt.

Comença per configurar les variables d'entorn del shell Bash `$DEBEMAIL` i `$DEBFULLNAME` que faran servir les eines de manteniment de Debian per tal d'obtenir el teu nom i adreça electrònica com s'indica a continuació¹⁶.

```
$ cat >>~/.bashrc <<EOF
DEBEMAIL="your.email.address@example.org"
DEBFULLNAME="Firstname Lastname"
export DEBEMAIL DEBFULLNAME
EOF
$ . ~/.bashrc
```

2.8 Paquet Debian no nadiu inicial.

Els paquets normals Debian són paquets de Debian no nadius construïts a partir dels programes originals. Si vols construir un paquet no nadiu Debian del codi font `gentoo-0.9.12.tar.gz`, pots construir el paquet no nadiu inicial Debian fent servir l'ordre **dh_make** de la següent manera:

```
$ cd ~/gentoo
$ wget http://example.org/gentoo-0.9.12.tar.gz
$ tar -xvzf gentoo-0.9.12.tar.gz
$ cd gentoo-0.9.12
$ dh_make -f ../gentoo-0.9.12.tar.gz
```

Caldrà canviar el nom del fitxer amb el codi font¹⁷. Consulta `dh_make(8)` per a una descripció més detallada.

You should see some output asking you what sort of package you want to create. Gentoo is a single binary package—it creates only one binary package, i.e., one `.deb` file—so we will select the first option (with the `S` key), check the information on the screen, and confirm by pressing `ENTER`.¹⁸

Després d'executar l'ordre **dh_make**, es fa una còpia del fitxer amb el codi original amb el nom `gentoo_0.9.12.orig.tar.gz` en el directori arrel per facilitar la construcció del paquet de fonts no nadiu de Debian amb el fitxer `debian.tar.gz`:

```
$ cd ~/gentoo ; ls -F
gentoo-0.9.12/
gentoo-0.9.12.tar.gz
gentoo_0.9.12.orig.tar.gz
```

Posa atenció en els dos canvis en el nom del fitxer `gentoo_0.9.12.orig.tar.gz`:

- El nom del paquet i de la versió estan separats per «`_`».
- S'ha afegit `.orig` abans de l'extensió `.tar.gz`.

Fitxa't que hi ha nous fitxers (de plantilla) en el directori `debian` dels quals es parlarà en Capítol 4 i Capítol 5. El procés d'empaquetament no està totalment automatitzat. La modificació dels fitxers Debian s'explicarà a Capítol 3. A continuació es compilarà el paquet Debian en l'apartat Capítol 6, la revisió del resultat a l'apartat Capítol 7 i la transferència del paquet al repositori a Capítol 9. S'explicarà cada etapa a continuació.

¹⁶El text mostrat a continuació dona per suposat que fas servir «Bash» com a intèrpret d'ordres. Si fas servir altres intèrprets, com és ara «Z shell», hauràs de fer servir els seus arxius de configuració en lloc de `~/.bashrc`.

¹⁷If the upstream source provides the `debian` directory and its contents, run the **dh_make** command with the extra option `--addmissing`. The new source 3.0 (`quilt`) format is robust enough not to break even for these packages. You may need to update the contents provided by the upstream version for your Debian package.

¹⁸Les opcions sobre el tipus de paquet són les següents: `s` per a un binari, `i` per a un paquet independent de l'arquitectura (paquet de codi font o de documentació), `m` per a paquets amb més d'un arxiu binari, `l` per a biblioteques, `k` per a un mòdul del nucli («kernel»), `n` per a un pegat del nucli i `b` per a paquets `cdb`s. Aquest document es centra amb l'ús del paquet `debhelper` i l'ordre **dh** per a la construcció de paquets amb un binari i tracta només parcialment el seu ús en la construcció de paquets independents de l'arquitectura i amb més d'un arxiu binari. El paquet `cdb`s proporciona guions alternatius a l'ordre **dh** i no s'explica en aquest document.

Si casualment elimines algun dels fitxers de plantilles del directori «debian», pots tornar a generar-los executant **dh_make** amb l'opció **--admissing** des del directori de les fonts.

L'actualització d'un paquet ja construït és un procés més complex: es probable que s'hagi construït amb procediments distints als que es fan servir actualment. Es millor treballar amb paquets actualitzats per aprendre els procediments bàsics de la construcció de paquets. Per empaquetar una revisió o una nova versió del teu paquet en el futur, faràs servir un procediment distint. Tot això s'explicarà a la secció Capítol 8.

Please note that the source file does not need to contain any build system discussed in Secció 2.4 and Secció 2.5. It could be just a collection of graphical data, etc. Installation of files may be carried out using only **debhelper** configuration files such as **debian/install** (see Secció 5.11).

Capítol 3

Modificar les fonts originals.

No hi ha espai en aquest document per explicar *tots* els detalls de les modificacions que poden haver de fer-se a les fonts originals. Tot i així, a continuació s'expliquen els problemes més freqüents.

3.1 Configuració de quilt.

El programa **quilt** proporciona un mètode bàsic per aplicar i conservar les modificacions del codi font en la construcció de paquets Debian. Atès que és desitjable fer alguna modificació a la configuració predeterminada, anem a crear un àlies **dquilt** per a la construcció de paquets Debian afegint la següent línia a l'arxiu `~/.bashrc`. La segona línia garanteix que l'àlies tindrà la mateixa funció d'auto-completar del intèrpret d'ordres per l'ordre **dquilt** que l'ordre **quilt**:

```
alias dquilt="quilt --quiltrc=${HOME}/.quiltrc-dpkg"
complete -F _quilt_completion -o filenames dquilt
```

Ara genera l'arxiu `~/.quiltrc-dpkg` de la següent manera:

```
d=. ; while [ ! -d $d/debian -a $(readlink -e $d) != / ]; do d=$d/..; done
if [ -d $d/debian ] && [ -z $QUILT_PATCHES ]; then
    # if in Debian packaging tree with unset $QUILT_PATCHES
    QUILT_PATCHES="debian/patches"
    QUILT_PATCH_OPTS="--reject-format=unified"
    QUILT_DIFF_ARGS="-p ab --no-timestamps --no-index --color=auto"
    QUILT_REFRESH_ARGS="-p ab --no-timestamps --no-index"
    QUILT_COLORS="diff_hdr=1;32:diff_add=1;34:diff_rem=1;31:diff_hunk=1;33:diff_ctx=35: ↵
        diff_cctx=33"
    if ! [ -d $d/debian/patches ]; then mkdir $d/debian/patches; fi
fi
```

Consulta `quilt(1)` i `/usr/share/doc/quilt/quilt.pdf.gz` per fer servir **quilt**.

3.2 Apedaçant el codi font.

Let's assume you find an error in the upstream Makefile as follows, where `install: gentoo` should have been `install: gentoo-target`.

```
install: gentoo
    install ./gentoo $(BIN)
    install icons/* $(ICONS)
    install gentoorc-example $(HOME)/.gentoorc
```

Anem a adobar l'errada amb l'ordre **dquilt** desant les modificacions a realitzar en el fitxer `fix-gentoo-target.patch`¹:

```
$ mkdir debian/patches
$ dquilt new fix-gentoo-target.patch
$ dquilt add Makefile
```

Ara fes els canvis necessaris en el fitxer `Makefile` (amb un editor) original i deixa'l així:

```
install: gentoo-target
    install ./gentoo $(BIN)
    install icons/* $(ICONS)
    install gentoorc-example $(HOME)/.gentoorc
```

A continuació executa **dquilt** (com s'indica a l'exemple següent) per actualitzar el pedaç generant el fitxer `debian/patches/fix-gentoo-target.patch` i afegeix la descripció de la modificació realitzada com es descriu a [DEP-3: Patch Tagging Guidelines](http://dep.debian.net/deps/dep3/) (<http://dep.debian.net/deps/dep3/>):

```
$ dquilt refresh
$ dquilt header -e
... descriu la modificació
```

3.3 La instal·lació dels arxius al seu destí

En general, els programes s'instal·len en el directori `/usr/local`. Però els paquets Debian no poden fer servir aquest directori d'ús privat de l'usuari administrador: les instal·lacions es fan en els directoris del sistema com és ara `/usr/bin` com estableix la normativa de jerarquia del sistema de fitxers («Filesystem Hierarchy Standard» [Filesystem Hierarchy Standard](http://www.debian.org/doc/packaging-manuals/fhs/fhs-3.0.html) (<http://www.debian.org/doc/packaging-manuals/fhs/fhs-3.0.html>) (FHS)).

És freqüent fer servir l'ordre `make(1)` per a la construcció (compilació) automatitzada del programa i l'execució de l'ordre `make install` instal·la directament el programa en el directori desitjat executant la secció `install` del fitxer `Makefile`. En la construcció dels paquets binaris de Debian, el sistema de construcció simula la instal·lació del programa en una reconstrucció de l'estructura de directoris del programa en un directori temporal en lloc de fer-ho en la destinació real.

Aquestes diferències entre la instal·lació del programa i la «simulació» d'instal·lació en el procés d'empaquetament Debian, és gestionada de manera transparent pel paquet `debhelper` amb les ordres **dh_auto_configure** i **dh_auto_install** si es compleixen els següents requisits:

- Cal que el fitxer `Makefile` segueixi les convencions GNU i accepti la variable `$(DESTDIR)`².
- Cal que el codi font segueixi l'estàndard de la jerarquia del sistema de fitxers («Filesystem Hierarchy Standard» o FHS).

Els programes que fan servir **autoconf** de GNU compleixen *automàticament* amb les convencions GNU i el seu empaquetament es quasi *automàtic*. Amb aquests requisits i l'heurística emprada pel paquet `debhelper`, és possible empaquetar sense fer canvis en el sistema de compilació. L'empaquetament no es tan complicat com pot semblar.

Per fer canvis en el fitxer `Makefile`, assegura't que permet fer servir la variable `$(DESTDIR)`. La variable `$(DESTDIR)` no es defineix a l'arxiu i s'afegirà a totes les adreces de directoris que es facin servir en la instal·lació del programa. El guió d'empaquetament estableix el valor de la variable `$(DESTDIR)` al valor del directori temporal d'instal·lació del programa en el procés de construcció del paquet.

Per a un paquet de fonts que generi només un paquet binari, el directori temporal fet servir per l'ordre **dh_auto_install** és `debian/nom_del_paquet` per a paquets amb un arxiu binari³. El contingut complet del directori temporal s'instal·larà en

¹L'anterior execució de **dh_make** hauria d'haver generat el directori `debian/patches`. En cas contrari o bé si estàs treballant en l'actualització d'un paquet, caldrà executar la primera ordre «`mkdir debian/patches`».

²Consulta [GNU Coding Standards: 7.2.4 DESTDIR: Support for Staged Installs](http://www.gnu.org/prep/standards/html_node/DESTDIR.html#DESTDIR) (http://www.gnu.org/prep/standards/html_node/DESTDIR.html#DESTDIR).

³Per a un paquet de fonts que generi més d'un paquet binari, l'ordre **dh_auto_install** fa servir el directori temporal `debian/tmp` i l'ordre **dh_install** amb l'ajuda dels fitxers `debian/paquet-1.install` i `debian/paquet-2.install` distribuirà el contingut del directori `debian/tmp` en els directoris temporals `debian/paquet-1` i `debian/paquet-2` per tal de construir els paquets binaris `package-1_*.deb` i `package-2_*.deb`.

el sistema de l'usuari quan aquest instal·li el paquet, amb la diferència que amb **dpkg** la instal·lació es fa des del directori arrel del sistema (en lloc del directori «debian/» o «debian/tmp» fet servir en la construcció del paquet).

Tots els fitxers i directoris que s'instal·len en el directori `debian/nom_de_l_paquet` en la construcció del paquet, hauran de poder instal·lar-se correctament des del directori arrel quan s'instal·lin fent servir el fitxer `.deb`. No s'ha de fer servir cadenes del tipus `/home/el_meu/deb/nom_de_l_paquet-versió/usr/share/nom_de_l_paquet` en la construcció del paquet.

Aquest és la part important del fitxer `Makefile` del paquet `gentoo`⁴:

```
# On s'instal·laran els executables amb 'make install'?
BIN      = /usr/local/bin
# On s'instal·laran els icones amb 'make install'?
ICONS    = /usr/local/share/gentoo
```

We see that the files are set to install under `/usr/local`. As explained above, that directory hierarchy is reserved for local use on Debian, so change those paths as follows:

```
# On s'instal·laran els executables amb 'make install'?
BIN      = $(DESTDIR)/usr/bin
# On s'instal·laran els icones amb 'make install'?
ICONS    = $(DESTDIR)/usr/share/gentoo
```

El destí correcte dels fitxers binaris, icones, documentació, etc, s'explica en el document «Estàndard de la jerarquia del sistema de fitxers». Cal que llegeixis les seccions que poden ésser d'aplicació al teu paquet.

Així doncs, hauríem d'instal·lar els arxius executables en el directori `/usr/bin` en lloc de `/usr/local/bin` i la pàgina de manual a `/usr/share/man/man1` en lloc de `/usr/local/man/man1`. No hi ha cap referència a la pàgina de manual en el fitxer `Makefile` de `gentoo`, però a Debian cal que cada programa tenguin la seva pàgina de manual, així que més endavant en farem una i la instal·larem a `/usr/share/man/man1`.

Alguns programes no fan servir variables en el fitxer `makefile` per definir les ubicacions dels fitxers. Això significa que hauràs d'editar alguns dels fitxers de codi C per solucionar això per tal de que es facin servir els directoris correctes. Però, on cal buscar?, i exactament, què? Prova d'executar el següent:

```
$ grep -nr --include='*.[c|h]' -e 'usr/local/lib' .
```

grep buscarà recursivament en l'estructura de directoris i indicarà el nom del fitxer i la línia on hi hagi alguna concordança.

Ara edita els fitxers i canvia la cadena `usr/local/lib` per `usr/lib`. La següent ordre hauria de fer-ho automàticament:

```
$ sed -i -e 's#usr/local/lib#usr/lib#g' \
    $(find . -type f -name '*.[c|h]')
```

Si voleu confirmar cada una de les substitucions en el seu lloc, això es pot fer de forma interactiva de la següent manera:

```
$ vim '+argdo %s#usr/local/lib#usr/lib#gce|update' +q \
    $(find . -type f -name '*.[c|h]')
```

Fet això hauries de trobar (en el fitxer `Makefile`) l'objectiu `install`: (busca una línia que comenci per «install:») i canviar el nom de totes les referències a directoris distints dels definits a l'inici de l'arxiu `Makefile`.

A l'original, l'objectiu «install» de `gentoo` posava:

```
install: gentoo-target
    install ./gentoo $(BIN)
    install icons/* $(ICONS)
    install gentoorc-example $(HOME)/.gentoorc
```

Corregirem el problema i conservarem les modificacions en el fitxer `debian/patches/install.patch` amb l'ordre **dquilt**.

⁴Només és un exemple del contingut de l'arxiu `Makefile`. Si el fitxer `Makefile` es construeix amb l'ordre `./configure`, el procediment correcte per solucionar això en l'arxiu `Makefile` és executar l'ordre `./configure` des de l'ordre `dh_auto_configure` amb les opcions predeterminades afegint l'opció `--prefix=/usr`.

```
$ dquilt new install.patch
$ dquilt add Makefile
```

Anem a canviar això per al paquet Debian amb l'editor:

```
install: gentoo-target
        install -d $(BIN) $(ICONS) $(DESTDIR)/etc
        install ./gentoo $(BIN)
        install -m644 icons/* $(ICONS)
        install -m644 gentoorc-example $(DESTDIR)/etc/gentoorc
```

Fixa't que hi ha la primera línia que és nova (`install -d`) abans de les altres. El fitxer `Makefile` original no estava aquesta ordre degut a què els directoris `/usr/local/bin` ja existeixen en el sistema quan s'executa `make install`. Per això, com farem la instal·lació en un directori buit (o inexistent), caldrà assegurar-se que els directoris necessaris ja existeixen.

També podem afegir altres coses al final de la regla, com la instal·lació de la documentació addicional que de vegades els autors originals poden haver descuidat:

```
install -d $(DESTDIR)/usr/share/doc/gentoo/html
cp -a docs/* $(DESTDIR)/usr/share/doc/gentoo/html
```

Després de comprovar que tot sigui correcte, fes que **dquilt** actualitzi la modificació en el fitxer `debian/patches/install.patch` i afegeix la descripció en la capçalera del fitxer:

```
$ dquilt refresh
$ dquilt header -e
... descriu la modificació
```

Ja tens un parell de pegats del paquet:

1. Correcció d'un error en el codi font: `debian/patches/fix-gentoo-target.patch`
2. Una modificació específica de l'empaquetat Debian: `debian/patches/install.patch`

Whenever you make changes that are not specific to the Debian package such as `debian/patches/fix-gentoo-target.patch`, be sure to send them to the upstream maintainer so they can be included in the next version of the program and be useful to everyone else. Also remember to avoid making your fixes specific to Debian or Linux —or even Unix! Make them portable. This will make your fixes much easier to apply.

No cal enviar cap dels fitxers del directori `debian/*` a l'autor original.

3.4 Diferències a les biblioteques.

There is one other common problem: libraries are often different from platform to platform. For example, a `Makefile` can contain a reference to a library that doesn't exist on the Debian system. In that case, we need to change it to a library that does exist in Debian, and serves the same purpose.

Anem a suposar que a l'arxiu `Makefile` (o `Makefile.in`) del teu programa hi ha una línia que posa alguna cosa com el següent.

```
LIBS = -lfoo -lbar
```

Si el teu programa no es compila degut a que la biblioteca `foo` no està present i el seu equivalent és proporcionat per la biblioteca `foo2` en els sistemes Debian, pots corregir aquest problema de compilació canviant `foo` per `foo2` a l'arxiu `debian/patches/foo2.patch`⁵:

⁵Si hi ha canvis a l'API de la biblioteca `foo` a la biblioteca `foo2`, els canvis en el codi font s'hauran de fer de manera que siguin compatibles amb la nova API.

```
$ dquilt new foo2.patch
$ dquilt add Makefile
$ sed -i -e 's/-lfoo/-lfoo2/g' Makefile
$ dquilt refresh
$ dquilt header -e
.. descriu el canvi
```


Capítol 4

Fitxers necessaris en el directori debian.

Ara tens un nou directori en el directori principal del programa («gentoo-0.9.12»), amb el nom `debian`. Hi ha alguns fitxers en aquest directori que caldrà editar per adaptar-los. Els fitxers més importants són `control`, `changelog`, `copyright` i `rules`, tots ells s'han d'incloure a tots els paquets ¹.

4.1 El fitxer `control`.

La informació d'aquest fitxer es fa servir per `dpkg`, `dselect`, `apt-get`, `apt-cache`, `aptitude` i altres eines de gestió de paquets. El seu contingut està explicat a «[Debian Policy Manual, 5 'Control files and their fields'](http://www.debian.org/doc/debian-policy/ch-controlfields.html)» (<http://www.debian.org/doc/debian-policy/ch-controlfields.html>).

Aquest és el contingut del fitxer `control` generat per `dh_make`:

```
1 Source: gentoo
2 Section: unknown
3 Priority: optional
4 Maintainer: Josip Rodin <joy-mg@debian.org>
5 Build-Depends: debhelper (>=10)
6 Standards-Version: 4.0.0
7 Homepage: <insert the upstream URL, if relevant>
8
9 Package: gentoo
10 Architecture: any
11 Depends: ${shlibs:Depends}, ${misc:Depends}
12 Description: <insert up to 60 chars description>
13 <insert long description, indented with spaces>
```

(S'han afegit els números de les línies).

Lines 1–7 are the control information for the source package. Lines 9–13 are the control information for the binary package.

La línia 1 és el nom del paquet font.

La línia 2 és la secció de la distribució a la qual s'assignarà el paquet.

Debian està dividit en seccions: `main` (principal, per als programes de codi lliure), `non-free` (propietaris, per als programes propietaris) i `contrib` (per als programes que depenen de programari propietari). A cada secció hi ha subdivisions segons el tipus de programari. Tenim `admin` per a programes que es fan servir per l'administració del sistema (habitualment només per l'usuari «root»), `base` per a les eines bàsiques del sistema, `devel` per al programari de desenvolupament, `doc` per a la

¹En aquest capítol, es farà referència als arxius ubicats en el directori `debian` sense afegir el prefix `debian/` per a simplificar el text i sempre que no hi hagi possibilitat de confusió.

documentació, `libs` per a les biblioteques, `mail` per a lectors de correu i dimonis de correu electrònic, `net` per a programes i dimonis de xarxa, `x11` per a programes específics de X11 i molts d'altres ².

Anem a canviar-ho per `x11` (el prefix `main/` és implícit, pel que podem ometre'l).

La línia 3 descriu la importància que pot tenir per a l'usuari (o el sistema) la instal·lació del paquet ³.

- La prioritat `optional` es fa servir per a paquets nous que no tenen conflictes amb altres paquets amb prioritat `required`, `important` o `standard`.

«Section» i «Priority» es fan servir per les interfícies com **aptitude** quan ordenen els paquets i seleccionen els predeterminats. Una vegada el teu paquet s'afegeixi a Debian, el contingut d'aquests camps pot ésser canviat per les persones responsables del repositori, que t'informaran per correu electrònic.

Com és un paquet de prioritat normal i no té conflictes amb cap altre paquet, deixarem la prioritat a `optional` (opcional).

A la línia 4 hi ha el nom i l'adreça electrònica del desenvolupador. Assegura't que l'adreça electrònica és vàlida per al camp `A`: el sistema de seguiment d'errades («Bug Tracking System») la farà servir per enviar-te els missatges de les errades del paquet. No facis servir «&», el signe «&» i parèntesi.

Line 5 includes the list of packages required to build your package as the `Build-Depends` field. You can also have the `Build-Depends-Indep` field as an additional line here. ⁴ Some packages like `gcc` and `make` which are required by the `build-essential` package are implied. If you need to have other tools to build your package, you should add them to these fields. Multiple entries are separated with commas; read on for the explanation of binary package dependencies to find out more about the syntax of these lines.

- A tots els paquets construïts amb l'ordre **dh** en el fitxer `debian/rules`, caldrà posar `debhelper` (`>=9`) en el camp `Build-Depends` per seguir les normes Debian respecte a l'objectiu `clean`.
- Source packages which have binary packages with `Architecture: any` are rebuilt by the autobuilder. Since this autobuilder procedure installs only the packages listed in the `Build-Depends` field before running `debian/rules build` (see Secció 6.2), the `Build-Depends` field needs to list practically all the required packages, and `Build-Depends-Indep` is rarely used.
- Els paquets de fonts amb binaris del tipus `Architecture: all`, el camp `Build-Depends-Indep` inclourà tots els paquets necessaris amb excepció dels llistats en el camp `Build-Depends` per tal de seguir els requeriments de les normes Debian respecte a l'objectiu `clean`.

Si tens dubtes, fes servir el camp `Build-Depends` ⁵.

Si vols saber quins paquets són necessaris per compilar el teu programa, executa l'ordre (si el programa fa servir «configure» per compilar-se):

```
$ dpkg-depcheck -d ./configure
```

Si vols saber les dependències de compilació d'un paquet `/usr/bin/nom_del_paquet`, executa:

```
$ objdump -p /usr/bin/nom_del_paquet | grep NEEDED
```

and for each library listed (e.g., `libfoo.so.6`), execute

```
$ dpkg -S libfoo.so.6
```

²Consulta [Debian Policy Manual, 2.4 'Sections'](http://www.debian.org/doc/debian-policy/ch-archive.html#s-subsections) (<http://www.debian.org/doc/debian-policy/ch-archive.html#s-subsections>) and [Llistat de seccions a sid](http://packages.debian.org/unstable/) (<http://packages.debian.org/unstable/>).

³Consulta [Manual de normes de Debian, 2.5 'Prioritat'](http://www.debian.org/doc/debian-policy/ch-archive.html#s-priorities) (<http://www.debian.org/doc/debian-policy/ch-archive.html#s-priorities>).

⁴Consulta [Debian Policy Manual, 7.7 'Relationships between source and binary packages - Build-Depends, Build-Depends-Indep, Build-Conflicts, Build-Conflicts-Indep'](http://www.debian.org/doc/debian-policy/ch-relationships.html#s-sourcebinarydeps) (<http://www.debian.org/doc/debian-policy/ch-relationships.html#s-sourcebinarydeps>).

⁵Aquesta situació una mica estranya és una característica ben documentada a la «[Debian Policy Manual, Footnotes 55](http://www.debian.org/doc/debian-policy/footnotes.html#f55)» (<http://www.debian.org/doc/debian-policy/footnotes.html#f55>). En tot cas, això és degut al funcionament de **dpkg-buildpackage**, no pel fet de fer servir l'ordre **dh** en el fitxer `debian/rules`. Tot això també s'explica a «[auto build system for Ubuntu](https://bugs.launchpad.net/launchpad-build/+bug/238141)» (<https://bugs.launchpad.net/launchpad-build/+bug/238141>).

Cal fer servir la versió `-dev` de cada un dels paquets llistats en el camp `Build-Depends`. Si fas servir el programa **ldd** per saber les dependències, també t'informarà de les dependències de les biblioteques indirectes, la qual cosa pot fer que la llista sigui massa llarga.

gentoo fa servir `xlibs-dev`, `libgtk1.2-dev` i `libglib1.2-dev` en la compilació, així que caldrà afegir-ho tot a continuació de `debhelper`.

La línia 6 és la versió dels estàndards definits en les normes Debian i que s'han seguit en la construcció del paquet, és a dir, la versió del manual de normes que has seguit per empaquetar (consulta «[Debian Policy Manual](http://www.debian.org/doc/devel-manuals#policy)» (<http://www.debian.org/doc/devel-manuals#policy>)).

A la línia 7 està l'adreça URL del programa.

La línia 9 és el nom del paquet binari. Normalment és el mateix que el paquet del codi original, tot i que no és imprescindible que sigui així.

A la línia 10 es descriu les architectures en les quals pot ésser compilat el paquet. Aquest valor és un dels següents depenent del tipus de paquet binari ⁶:

- **Architecture:** `any`

- El paquet binari generat és depenent de l'arquitectura, generalment quan es tracta d'un programa escrit en un llenguatge compilat.

- **Architecture:** `all`

- El paquet binari generat és independent de l'arquitectura, generalment quan es tracta d'arxius de text, imatges o guions escrits en un llenguatge interpretat.

Eliminarem la línia 10 degut a que el programa està escrit en C. `dpkg-gencontrol(1)` omplirà el valor apropiat d'arquitectura per a cada màquina en què és possible compilar el paquet de fonts.

Si el teu paquet és independent de l'arquitectura (serà el cas d'un document o d'un guió escrit en Perl), canvia el valor a `all` i consulta més endavant Secció 4.4 per saber com fer servir la regla `binary-indep` en lloc de `binary-arch` per a la construcció del paquet.

La línia 11 mostra una de les més poderoses possibilitats del sistema de paquets de Debian. És possible relacionar els paquets entre ells. A més de `Depends` (depèn de) altres camps de relació són `Recommends` (recomana), `Suggests` (suggereix), `Pre-Depends` (pre-depèn de), `Breaks` (trenca a), `Conflicts` (entre en conflicte amb), `Provides` (proveeix), `Replaces` (reemplaça a).

Les eines de gestió de paquets tracten de la mateixa manera les relacions entre els paquets; els casos en què no és així s'explicaran en el seu moment (consulta `dpkg(8)`, `dselect(8)`, `apt(8)`, `aptitude(1)`, etc.).

Heus aquí una descripció simplificada de les relacions entre paquets ⁷:

- **Depends**

El programa no s'instal·larà fins què s'instal·lin els paquets dels quals depèn. Fes servir aquesta opció si el teu programa no funciona (o es trenca fàcilment) sense un paquet determinat.

- **Recommends**

Aquesta opció és per a paquets que no són imprescindibles per al funcionament del programa però que es fan servir juntament amb ell. Quan els usuaris instal·lin el teu paquet, totes les interfícies d'instal·lació aconsellaran la instal·lació dels paquets recomanats. **aptitude** i **apt-get** instal·len els paquets recomanats (però l'usuari pot decidir no fer-ho). **dpkg** ignora el contingut d'aquest camp.

⁶Consulta [Debian Policy Manual](http://www.debian.org/doc/debian-policy/ch-controlfields.html#s-f-Architecture), 5.6.8 "Architecture" (<http://www.debian.org/doc/debian-policy/ch-controlfields.html#s-f-Architecture>) per a més detalls.

⁷Consulta [Debian Policy Manual](http://www.debian.org/doc/debian-policy/ch-relationships.html), 7 "Declaring relationships between packages" (<http://www.debian.org/doc/debian-policy/ch-relationships.html>) .

- **Suggests**

Fes servir aquesta opció per a paquets que funcionaran bé amb el teu programa però que no són en absolut necessaris.. Quan l'usuari instal·li el teu programa, probablement no se li demanarà que s'instal·lin els paquets suggerits. És possible configurar **aptitude** per a què instal·li els paquets suggerits (no és l'opció per defecte). **dpkg** i **apt-get** ignoren aquestes dependències. Fes servir aquesta opció per a paquets que funcionaran bé amb el teu programa però que no són en absolut necessaris.

- **Pre-Depends**

Aquesta opció és més radical que **Depends**. El paquet no s'instal·larà fins que els paquets dels quals pre-depen estiguin instal·lats i *correctament configurats*. Fes servir aquesta opció *poques vegades* i només després d'haver-ho discutit a la llista de distribució debian-devel@lists.debian.org (<http://lists.debian.org/debian-devel/>) . Més clarament: no facis servir aquesta opció :-).

- **Conflicts**

El paquet no s'instal·larà fins què tots el paquets amb els quals entre en conflicte siguin eliminats. Fes servir aquesta opció si el teu programa no funcionarà amb absolut (o fallarà fàcilment) si un paquet en concret no està instal·lat.

- **Breaks**

Si el paquet s'instal·la, tots els paquets de la llista es trencaran. Normalment, els paquets inclosos en la llista **Breaks** tenen una clàusula de versió anterior. La solució és fe servir eines de gestió de paquets més sofisticades per actualitzar els paquets de la llista.

- **Provides**

For some types of packages where there are multiple alternatives, virtual names have been defined. You can get the full list in the [virtual-package-names-list.txt.gz](http://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt) (<http://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt>) file. Use this if your program provides a function of an existing virtual package.

- **Replaces**

Fes servir aquesta opció només si el teu programa substitueix fitxers d'un altra paquet o el paquet complet (generalment es fa servir aquesta opció conjuntament amb **Conflicts**). S'eliminaran els fitxers dels paquets indicats abans d'instal·lar el teu.

Tots aquests camps tenen una sintaxi uniforme: es tracta d'una llista de noms de paquets separats per comes. Aquests noms de paquets també poden ésser llistes de paquets alternatius, separats amb una barra vertical «|» (símbol de tub).

Els camps poden restringir la seva aplicació a versions determinades del paquet llistat. Aquestes versions s'enumeren entre parèntesi després de cada nom de paquet individual, i ha de contenir una relació de la següent llista seguit pel nombre de versió. Les relacions possibles són: <<, <=, =, >= i >> per a estrictament anterior, anterior o igual, exactament igual, posterior o igual i estrictament posterior, respectivament. Per exemple:

```
Depends: foo (>= 1.2), libbar1 (= 1.3.4)
Conflicts: baz
Recommends: libbaz4 (> 4.0.7)
Suggests: quux
Replaces: quux (<< 5), quux-foo (<= 7.6)
```

L'última funcionalitat que necessites conèixer és `${shlibs:Depends}`, `${perl:Depends}`, `${misc:Depends}`, etc.

`dh_shlibdeps(1)` determina les dependències de biblioteques compartides pels paquets binaris. Genera una llista d'executables **ELF** i biblioteques compartides per a cada paquet binari. Aquesta llista es farà servir per substituir `${shlibs:Depends}`.

`dh_perl(1)` comprova les dependències Perl. Genera una llista de dependències de `perl` o `perlapi` per a cada paquet binari. Aquesta llista es fa servir per reemplaçar `${perl:Depends}`.

Algunes ordres de `debhelper` determinen les dependències dels paquets llistats anteriorment. Les ordres generen una llista dels paquets necessaris per a cada un dels paquets binaris. La llista d'aquests paquets reemplaçarà a `${misc:Depends}`.

`dh_gencontrol(1)` genera l'arxiu `DEBIAN/control` per a cada paquet binari substituint `${shlibs:Depends}`, `${perl:Depends}` `${misc:Depends}`, etc.

La línia **Depends** pot quedar així com està però afegirem una línia **Suggests**: `file`, degut a què el paquet `gentoo` fa servir algunes funcions del paquet `file`.

La línia 9 conté la URL del programa. S'assumeix que és <http://www.obsession.se/gentoo/>.

La línia 12 és una descripció curta del programa. La major part dels monitors dels usuaris són de 80 columnes d'amplada (els terminals, és clar!) per la qual cosa la descripció no hauria de tenir més de 60 caràcters. Canviaré la descripció per a **fully GUI-configurable, two-pane X file manager**. («Interfície gràfica d'usuari GTK+ de gestió de fitxers completament configurable»).

La línia 13 és per fer una descripció més llarga del paquet. Cal que sigui almenys d'un paràgraf amb detalls sobre el paquet. Cada línia començarà amb un espai buit. No hi pot haver línies en blanc, però es pot simular amb un `.` (punt) a la segona columna. Tampoc hi pot haver més d'una línia en blanc al final de la descripció completa ⁸.

Afegirem els camps `Vcs - *` per a documentar la localització del sistema de control de versions (VCS) entre les línies 6 i 7 ⁹. Se suposa que el paquet `gentoo` està allotjat en el servei «Alioth Git» de Debian a `git://git.debian.org/git/collab-maint/gentoo.git`.

Finalment, el fitxer `control` actualitzat quedarà així:

```
1 Source: gentoo
2 Section: x11
3 Priority: optional
4 Maintainer: Josip Rodin <joy-mg@debian.org>
5 Build-Depends: debhelper (>=10), xlibs-dev, libgtk1.2-dev, libglib1.2-dev
6 Standards-Version: 4.0.0
7 Vcs-Git: https://anonscm.debian.org/git/collab-maint/gentoo.git
8 Vcs-browser: https://anonscm.debian.org/git/collab-maint/gentoo.git
9 Homepage: http://www.obsession.se/gentoo/
10
11 Package: gentoo
12 Architecture: any
13 Depends: ${shlibs:Depends}, ${misc:Depends}
14 Suggests: file
15 Description: fully GUI-configurable, two-pane X file manager
16  gentoo is a two-pane file manager for the X Window System. gentoo lets the
17  user do (almost) all of the configuration and customizing from within the
18  program itself. If you still prefer to hand-edit configuration files,
19  they're fairly easy to work with since they are written in an XML format.
20  .
21  gentoo features a fairly complex and powerful file identification system,
22  coupled to an object-oriented style system, which together give you a lot
23  of control over how files of different types are displayed and acted upon.
24  Additionally, over a hundred pixmap images are available for use in file
25  type descriptions.
26  .
29  gentoo was written from scratch in ANSI C, and it utilizes the GTK+ toolkit
30  for its interface.
```

(S'han afegit els números de les línies).

4.2 El fitxer `copyright`.

Aquest arxiu conté la informació de la llicència i drets d'autor de les fonts originals del paquet. [Debian Policy Manual, 12.5 Copyright information](http://www.debian.org/doc/debian-policy/ch-docs.html#s-copyrightfile) (http://www.debian.org/doc/debian-policy/ch-docs.html#s-copyrightfile) concreta el seu contingut i [DEP-5: Machine-parseable debian/copyright](http://dep.debian.net/deps/dep5/) (http://dep.debian.net/deps/dep5/) proporciona directrius per al seu format.

dh_make proporciona una plantilla per a l'arxiu `copyright`. Amb l'opció `--copyright gpl2` s'aconsegueix la plantilla per al paquet `gentoo` amb la llicència GPL-2.

⁸Aquestes descripcions s'han d'escriure en anglès. De la traducció s'encarrega el [The Debian Description Translation Project - DDTP](http://www.debian.org/intl/l10n/ddtp) (http://www.debian.org/intl/l10n/ddtp).

⁹Consulta [Developer's Reference, 6.2.5. "Version Control System location"](http://www.debian.org/doc/manuals/developers-reference/best-pkging-practices.html#bpp-vcs) (http://www.debian.org/doc/manuals/developers-reference/best-pkging-practices.html#bpp-vcs).

You must fill in missing information to complete this file, such as the place you got the package from, the actual copyright notice, and the license. For certain common free software licenses (GNU GPL-1, GNU GPL-2, GNU GPL-3, LGPL-2, LGPL-2.1, LGPL-3, GNU FDL-1.2, GNU FDL-1.3, Apache-2.0, 3-Clause BSD, CC0-1.0, MPL-1.1, MPL-2.0 or the Artistic license), you can just refer to the appropriate file in the `/usr/share/common-licenses/` directory that exists on every Debian system. Otherwise, you must include the complete license.

Resumint, el fitxer `copyright` del paquet `gentoo` hauria de ser més o menys com el següent exemple:

```
1 Format: https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
2 Upstream-Name: gentoo
3 Upstream-Contact: Emil Brink <emil@obsession.se>
4 Source: http://sourceforge.net/projects/gentoo/files/
5
6 Files: *
7 Copyright: 1998-2010 Emil Brink <emil@obsession.se>
8 License: GPL-2+
9
10 Files: icons/*
11 Copyright: 1998 Johan Hanson <johan@tiq.com>
12 License: GPL-2+
13
14 Files: debian/*
15 Copyright: 1998-2010 Josip Rodin <joy-mg@debian.org>
16 License: GPL-2+
17
18 License: GPL-2+
19 This program is free software; you can redistribute it and/or modify
20 it under the terms of the GNU General Public License as published by
21 the Free Software Foundation; either version 2 of the License, or
22 (at your option) any later version.
23 .
24 This program is distributed in the hope that it will be useful,
25 but WITHOUT ANY WARRANTY; without even the implied warranty of
26 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
27 GNU General Public License for more details.
28 .
29 You should have received a copy of the GNU General Public License along
30 with this program; if not, write to the Free Software Foundation, Inc.,
31 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
32 .
33 On Debian systems, the full text of the GNU General Public
34 License version 2 can be found in the file
35 '/usr/share/common-licenses/GPL-2'.
```

(S'han afegit els números de les línies).

Consulta el text «COM» redactat pel «ftpmasters» i enviat a «debian-devel-announce»: <http://lists.debian.org/debian-devel-announce/2006/03/msg00023.html>.

4.3 El fitxer `changelog`.

Aquest és un arxiu imprescindible amb un format especial descrit a les normes Debian, [Debian Policy Manual, 4.4 "debian/changelog"](http://www.debian.org/doc/debian-policy/ch-source.html#s-dpkgchangelog) (<http://www.debian.org/doc/debian-policy/ch-source.html#s-dpkgchangelog>). El seu especial format és degut a què **dpkg** i d'altres programes el fan servir per obtenir informació de la versió, revisió, distribució i urgència del paquet.

També és important aquest arxiu: facilita la documentació dels canvis fets en la construcció del paquet. El fitxer informarà als usuaris que descarreguin el paquet sobre els problemes que pugui tenir el paquet i que ells hagin de saber. El fitxer es conserva com `/usr/share/doc/gentoo/changelog.Debian.gz` en el paquet.

dh_make construeix un arxiu predeterminat amb el següent text:

```

1 gentoo (0.9.12-1) unstable; urgency=medium
2
3 * Initial release. (Closes: #nnnn) <nnnn is the bug number of your ITP>
4
5 -- Josip Rodin <joy-mg@debian.org> Mon, 22 Mar 2010 00:37:31 +0100
6

```

(S’han afegit els números de les línies).

Line 1 is the package name, version, distribution, and urgency. The name must match the source package name; distribution should be `unstable`, and urgency should be set to `medium` unless there is any particular reason for other values.

Lines 3-5 are a log entry, where you document changes made in this package revision (not the upstream changes —there is a special file for that purpose, created by the upstream authors, which you will later install as `/usr/share/doc/gentoo/change.log.gz`). Let’s assume your ITP (Intent To Package) bug report number was 12345. New lines must be inserted just below the uppermost line that begins with `*` (asterisk). You can do it with `dch(1)`. You can edit this manually with a text editor as long as you follow the formatting convention used by the `dch(1)`.

In order to prevent a package being accidentally uploaded before completing the package, it is a good idea to change the distribution value to an invalid distribution value of `UNRELEASED`.

Aquest serà el contingut del fitxer:

```

1 gentoo (0.9.12-1) UNRELEASED; urgency=low
2
3 * Initial Release. Closes: #12345
4 * This is my first Debian package.
5 * Adjusted the Makefile to fix $(DESTDIR) problems.
6
7 -- Josip Rodin <joy-mg@debian.org> Mon, 22 Mar 2010 00:37:31 +0100
8

```

(S’han afegit els números de les línies).

Quan estiguis satisfet dels canvis i els hakis documentat en el fitxer `change.log`, ja pots canviar el valor de la distribució de `UNRELEASED` pel nom de la distribució de destinació `unstable` (o bé per `experimental`).¹⁰

A la secció Capítol 8 s’explicaran alguns aspectes més de l’actualització del fitxer `change.log`.

4.4 El fitxer `rules`.

Now we need to take a look at the exact rules that `dpkg-buildpackage(1)` will use to actually create the package. This file is in fact another `Makefile`, but different from the one(s) in the upstream source. Unlike other files in `debian`, this one is marked as executable.

4.4.1 Objectius del fitxer `rules`.

Cada arxiu `rules`, igual que els arxius `Makefile`, consta de diversos objectius i les seves regles¹¹. Cada regla comença amb la declaració dels objectius a la primera columna. Les següents línies comencen amb una tabulació (codi ASCII 9) i les seves regles. Les línies buides i les començades amb `#` es tracten com a comentaris i s’ignoren¹².

A rule that you want to execute is invoked by its target name as a command line argument. For example, `debian/rules build` and `fakeroot make -f debian/rules binary` execute rules for `build` and `binary` targets, respectively.

A continuació tens una explicació simplificada dels objectius:

¹⁰Si fas servir l’ordre `dch -r` per fer efectiu aquest darrer canvi, assegura’t que deses el fitxer `change.log` des del editor.

¹¹You can start learning how to write a `Makefile` from [Debian Reference, 12.2. “Make”](http://www.debian.org/doc/manuals/debian-reference-ch12#_make) (http://www.debian.org/doc/manuals/debian-reference-ch12#_make). The full documentation is available as http://www.gnu.org/software/make/manual/html_node/index.html or as the `make-doc` package in the non-free archive area.

¹²[Debian Policy Manual, 4.9 “Main building script: debian/rules”](http://www.debian.org/doc/debian-policy/ch-source.html#s-debianrules) (<http://www.debian.org/doc/debian-policy/ch-source.html#s-debianrules>) explica els detalls.

- **clean** (obligatori): elimina tots els fitxers generats, compilats o innecessaris de l'estructura de directoris de les fonts.
- **build** (obligatori): construeix els fitxers executables o els documents amb format a partir dels fitxers de les fonts.
- objectiu **build-arch** (obligatori): per compilar les fonts en programes compilats dependents de l'arquitectura en l'arbre de directoris de compilació.
- objectiu **build-indep** (obligatori) : per compilar les fonts en documents amb format independents de l'arquitectura en l'arbre de directoris de compilació.
- **install** (opcional): fa la instal·lació en l'estructura de directoris temporal en el directori **debian** dels fitxers que constitueixen els paquets binaris. Si hi ha un objectiu **binary***, dependrà d'aquest.
- **binary** (obligatori): per a la construcció de cada un dels paquets binaris (combinat amb els objectius **binary-arch** i **binary-indep**)¹³.
- **binary-arch** (obligatori): per a la construcció de paquets dependents de l'arquitectura (els que tenen el camp **Architecture:** **all** en el fitxer «control»)¹⁴.
- **binary-indep** (obligatori): per a la construcció de paquets independents de l'arquitectura (**Architecture:** **all** en el fitxer «control»)¹⁵.
- **get-orig-source** (opcional): per determinar la versió més recent de les fonts originals des del magatzem de l'autor.

Probablement l'exposició no és gens clara, però tot quedarà més clar després de veure el fitxer **rules** que **dh_make** construeix com a plantilla.

4.4.2 Fitxer **rules** predeterminat.

La nova versió de **dh_make** genera un arxiu **rules** molt simple però poderós que fa servir l'ordre **dh**:

```
1 #!/usr/bin/make -f
2 # See debhelper(7) (uncomment to enable)
3 # output every command that modifies files on the build system.
4 #DH_VERBOSE = 1
5
6 # see FEATURE AREAS in dpkg-buildflags(1)
7 #export DEB_BUILD_MAINT_OPTIONS = hardening=+all
8
9 # see ENVIRONMENT in dpkg-buildflags(1)
10 # package maintainers to append CFLAGS
11 #export DEB_CFLAGS_MAINT_APPEND = -Wall -pedantic
12 # package maintainers to append LDFLAGS
13 #export DEB_LDFLAGS_MAINT_APPEND = -Wl,--as-needed
14
15
16 %:
17     dh $@
```

(He afegit els números de les línies i simplificat els comentaris. En el fitxer **debian/rules** els espais inicials de les línies són tabulacions).

Probablement estàs familiaritzat amb línies com la primera per a guions escrits en shell o Perl. Aquesta línia indica que el fitxer s'executa amb **/usr/bin/make**.

Cal eliminar la marca de comentari a la línia 4 per assignar el valor 1 a la variable **DH_VERBOSE**. En aquest cas, l'ordre **dh** escriurà les ordres **dh_*** en el terminal així com siguin executades per **dh**. Pots afegir la línia **export DH_OPTIONS=-v** aquí.

¹³Aquest objectiu és utilitzat per **dpkg-buildpackage** com en Secció 6.1.

¹⁴Aquest objectiu és utilitzat per **dpkg-buildpackage -B** com en Secció 6.2.

¹⁵Aquest objectiu és utilitzat per **dpkg-buildpackage -A**.

Això farà que cada ordre **dh_*** escrigui la sortida d'allò que fa. Això t'ajudarà a entendre com funciona el fitxer **rules** i a solucionar errades. La nova ordre **dh** és part fonamental de les eines **debhelper** i no t'amaguen res.

Tota la feina del fitxer la fan les línies 16 i 17 executant una regla implícita fent servir la regla patró. El símbol de percentatge substitueix «qualsevol objectiu» per a continuació executar únicament **dh** amb el nom de l'objectiu (com a opció) ¹⁶. L'ordre **dh** és un guió que executa les seqüències necessàries d'ordres **dh** * segons els seus paràmetres com es descriu a continuació ¹⁷:

- `debian/rules clean` executa `dh clean`, que al seu torn executa el següent:

```
dh_testdir
dh_auto_clean
dh_clean
```

- `debian/rules` build runs `dh build`, which in turn runs the following:

```
dh_testdir
dh_auto_configure
dh_auto_build
dh auto test
```

- `fakeroot debian/rules binary` runs `fakeroot dh binary`, which in turn runs the following¹⁸:

```
dh_testroot
dh_prep
dh_installdirs
dh_auto_install
dh_install
dh_installdocs
dh_installchangelogs
dh_installexamples
dh_installman
dh_installdcatalogs
dh_installcron
dh_installdebconf
dh_installemacsen
dh_installifupdown
dh_installinfo
dh_installinit
dh_installmenu
dh_installemime
dh_installdmodules
dh_installdlogcheck
dh_installdlogrotate
dh_installpam
dh_installppp
dh_installdudev
dh_installwm
dh_installdxfonts
dh_bugfiles
dh_lintian
dh_gconf
dh_icons
```

¹⁶This uses the new `debhelper` v7+ features. Its design concepts are explained in [Not Your Grandpa's Debhelper](http://joey.kitenet.net/talks/debhelper/-debhelper-slides.pdf) (<http://joey.kitenet.net/talks/debhelper/-debhelper-slides.pdf>) presented at DebConf9 by the `debhelper` upstream. Under Lenny, `dh_make` created a much more complicated `rules` file with explicit rules and many `dh_*` scripts listed for each one, most of which are now unnecessary (and show the package's age). The new `dh` command is simpler and frees us from doing the routine work "manually". You still have full power to customize the process with `override_dh_*` targets. See [Secció 4.4.3](#). It is based only on the `debhelper` package and does not obfuscate the package building process as the `cdbs` package tends to do.

¹⁷ You can verify the actual sequences of **dh_*** programs invoked for a given *target* without really running them by invoking **dh target --no-act** or **debian/rules --'target' --no-act**.

¹⁸El següent exemple assumeix que el teu fitxer `debian/compat` té un valor igual o major a 9 per tal d'evitar invocar qualsevol ordre «python» de suport automàticament.

```
dh_perl
dh_usrlocal
dh_link
dh_compress
dh_fixperms
dh_strip
dh_makeshlibs
dh_shlibdeps
dh_installdeb
dh_gencontrol
dh_md5sums
dh_builddeb
```

- `fakeroot debian/rules binary-arch` runs `fakeroot dh binary-arch`, which in turn runs the same sequence as `fakeroot dh binary` but with the `-a` option appended for each command.
- `fakeroot debian/rules binary-indep` runs `fakeroot dh binary-indep`, which in turn runs almost the same sequence as `fakeroot dh binary` but excluding `dh_strip`, `dh_makeshlibs`, and `dh_shlibdeps` with the `-i` option appended for each remaining command.

La funció de les ordres `dh_*` pot deduir-se del seu nom ¹⁹. A continuació es resumeix la funció de les ordres més importants per al cas de compilació basada en un arxiu `Makefile` ²⁰:

- **`dh_auto_clean`** executa el següent si hi ha un arxiu `Makefile` amb l'objectiu `distclean` ²¹.

```
make distclean
```

- **`dh_auto_configure`** executa el següent si es fa servir el fitxer `./configure` (els arguments s'ha abreviat per facilitar la lectura):

```
./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var ...
```

- **`dh_auto_build`** executa el següent per executar el primer objectiu del fitxer `Makefile` (si es fa servir aquest arxiu).

```
make
```

- **`dh_auto_test`** executa el següent si existeix l'objectiu `test` en el fitxer `Makefile` ²².

```
make test
```

- **`dh_auto_install`** executa el següent si existeix l'objectiu `install` en el fitxer `Makefile` (s'ha truncat la línia per facilitar la lectura):

```
make install \
DESTDIR=/directori/del/paquet_versió-revisió/debian/nom_del_paquet
```

Els objectius que cal executar amb l'ordre **`fakeroot`** contenen **`dh_testroot`**. Si no fas servir l'ordre «`fakeroot`» per simular l'execució per l'usuari «`root`», es produirà una errada que aturarà l'execució.

És important tenir present que el fitxer `rules` construït per **`dh_make`** només és un exemple. Serà útil per a la majoria dels paquets simples, però no deixis de fer les adaptacions necessàries.

Malgrat que `install` no és un objectiu obligatori, està acceptat. `fakeroot dh install` es comporta com `fakeroot dh binary` però s'atura després de **`dh_fixperms`**.

¹⁹Per a una descripció més completa de la funció de cada guió `dh_*` i les seves opcions, consulta els manuals respectius així com la documentació de `debhelper`.

²⁰These commands support other build environments, such as `setup.py`, which can be listed by executing `dh_auto_build --list` in a package source directory.

²¹Mira el primer objectiu disponible a l'arxiu `Makefile` com `distclean`, `realclean` o `clean` a l'arxiu `Makefile` i l'executa.

²²En realitat executa el primer dels objectius `test` o `check` del fitxer `Makefile`.

4.4.3 Personalització del fitxer `rules`.

Pots fer molts canvis per adaptar el fitxer `rules` construït per l'ordre `dh`.

L'ordre `dh $@` permet les següents adaptacions ²³:

- Afegir funcionalitat per a l'ordre **`dh_python2`** (la millor opció per a Python) ²⁴.
 - Afegeix el paquet `python` en el camp `Build-Depends`.
 - Fes servir `dh $@ --with python2` en el seu lloc.
 - Així es gestiona el mòdul Python fent servir el entorn («framework») de `python`.
- Afegir funcionalitat per a l'ordre **`dh_pysupport`** (obsolet).
 - Afegeix el paquet `python-support` en el camp `Build-Depends`.
 - Fes servir `dh $@ --with pysupport` en el seu lloc.
 - Així es gestiona el mòdul Python fent servir el entorn («framework») de `python-support`.
- Afegir funcionalitat per a l'ordre **`dh_pycentral`** (obsolet).
 - Afegeix el paquet `python-central` en el camp `Build-Depends`.
 - Fes servir `dh $@ --with python-central` en el seu lloc.
 - També es desactiva l'ordre **`dh_pysupport`**.
 - I es gestiona el mòdul Python fent servir l'entorn («framework») de `python-central`.
- Afegir funcionalitat per a l'ordre **`dh_installtex`**.
 - Afegeix el paquet `tex-common` en el camp `Build-Depends`.
 - Fes servir `dh $@ --with tex` en el seu lloc.
 - Així es registren els tipus de lletra «Type 1», els patrons per a la separació de paraules («hyphenation patterns») o els formats TeX.
- Afegir funcionalitat per a les ordres **`dh_quilt_patch`** i **`dh_quilt_unpatch`**.
 - Afegeix el paquet `quilt` en el camp `Build-Depends`.
 - Fes servir `dh $@ --with quilt` en el seu lloc.
 - Així s'apliquen o es desfan, en els fitxers de les fonts originals, les modificacions dels fitxers del directori `debian/patches` en els paquets construïts amb el format 1.0.
 - Aquesta adaptació no és necessària per als paquets construïts amb el format 3.0 (`quilt`).
- Afegir funcionalitat per a l'ordre **`dh_dkms`**.
 - Afegeix el paquet `dkms` en el camp `Build-Depends`.
 - Fes servir `dh $@ --with dkms` en el seu lloc.
 - Així es controla correctament l'ús de DKMS en la construcció de paquets del nucli.
- Afegir funcionalitat per a les ordres **`dh_autotools-dev_updateconfig`** i **`dh_autotools-dev_restoreconfig`**.
 - Afegeix el paquet `autotools-dev` en el camp `Build-Depends`.
 - Fes servir `dh $@ --with autotools-dev` en el seu lloc.
 - Així s'actualitza i restaura `config.sub` i `config.guess`.

²³Si un paquet instal·la el fitxer `/usr/share/perl5/Debian/Debhelper/Sequence/nom_arxiu.pm` pots activar la funció adaptada amb `dh $@ --with nom_arxiu`.

²⁴És preferible fer servir l'ordre **`dh_python2`** abans que l'ordre **`dh_pysupport`** o **`dh_pycentral`**. No facis servir l'ordre **`dh_python`**.

- Afegir funcionalitat per a les ordres **dh_autoreconf** i **dh_autoreconf_clean**.
 - Afegeix el paquet **dh-autoreconf** en el camp **Build-Depends**.
 - Fes servir **dh \$@ --with autoreconf** en el seu lloc.
 - Així s'actualitzen els fitxers del sistema de compilació GNU i els restaura després de la compilació.
- Afegir funcionalitat per a l'ordre **dh_girepository**.
 - Afegeix el paquet **gobject-introspection** en el camp **Build-Depends**.
 - Fes servir **dh \$@ --with gir** en el seu lloc.
 - Això computa les dependències dels paquets d'enviament de dades d'introspecció de «GObject» i genera la substitució de la variable `${gir:Depends}` per les dependències del paquet.
- Afegir funcionalitat d'autocompletar a **bash**.
 - Afegeix el paquet **bash-completion** en el camp **Build-Depends**.
 - Fes servir **dh \$@ --with bash-completion** en el seu lloc.
 - Així s'instal·la la funció d'autocompletar de **bash** fent servir l'arxiu de configuració de `debian/nom_de_l_paquet.bash-compl`

Moltes de les ordres **dh_*** executades per la nova ordre **dh** es poden personalitzar amb fitxers de configuració ubicats en el directori **debian**. Consulta Capítol 5 i els manuals (les «manpage») de cada ordre.

Algunes ordres **dh_*** invocades per la nova ordre **dh** poden necessitar l'addició d'arguments o l'execució d'ordres addicionals. Per a aquests casos, hauràs d'afegir l'objectiu **override_dh_nom_ordre** en el fitxer **rules** per a l'ordre **dh_nom_ordre** que cal canviar. Es tracta de *execute'm a mi en el seu lloc* ²⁵.

Les ordres **dh_auto_*** fan més coses que les explicades en aquesta explicació simplificada. L'ús d'ordres equivalents més senzilles en el lloc d'aquestes en els objectius **override_dh_*** (amb l'excepció de l'objectiu **override_dh_auto_clean**) no és una bona idea ja que pot eliminar funcions intel·ligents de **debhelper**.

Si les dades de configuració del paquet **gentoo** es desen en el directori `/etc/gentoo` en lloc del directori habitual `/etc`, cal anular l'execució de l'argument predeterminat `--sysconfig=/etc` de l'ordre **dh_auto_configure** per **./configure** amb:

```
override_dh_auto_configure:
    dh_auto_configure -- --sysconfig=/etc/gentoo
```

The arguments given after `--` are appended to the default arguments of the auto-executed program to override them. Using the **dh_auto_configure** command is better than directly invoking the **./configure** command here since it will only override the `--sysconfig` argument and retain any other, benign arguments to the **./configure** command.

Si el **Makefile** de les fonts de **gentoo** requereix l'especificació de l'objectiu **build** per a la compilació ²⁶, pots afegir un objectiu **override_dh_auto_build** per tal d'anul·lar-lo.

```
override_dh_auto_build:
    dh_auto_build -- build
```

D'aquesta forma es garanteix l'execució de `$(MAKE)` amb tots els arguments predeterminats per l'ordre **dh_auto_build** i de l'argument **build**.

Si el fitxer **Makefile** de les fonts de **gentoo** requereix l'especificació de l'objectiu **packageclean** (per fer neteja després de la compilació), en lloc dels objectius **distclean** o **clean** a l'arxiu **Makefile** pots afegir un objectiu **override_dh_auto_clean** per habilitar l'ordre.

```
override_dh_auto_clean:
    $(MAKE) packageclean
```

Si el fitxer **Makefile** de les fonts de **gentoo** té un objectiu **test** i desitges que no s'executi en la construcció del paquet, pots fer servir l'objectiu **override_dh_auto_test** sense ordres per tal de que no es faci res.

²⁵En **lenny**, quan calia canviar un guió **dh_*** calia trobar la línia adequada en el fitxer **rules** i canviar-la. Ara és suficient afegir un objectiu.

²⁶**dh_auto_build** sense arguments executarà el primer objectiu del fitxer **Makefile**.

```
override_dh_auto_test:
```

Si el paquet **gentoo** inclou el poc freqüent arxiu de canvis de l'autor amb el nom **FIXES**, **dh_installchangelogs** no en farà la instal·lació (és l'opció predeterminada). L'ordre **dh_installchangelogs** requereix l'argument **FIXES** per tal d'instal·lar-lo ²⁷.

```
override_dh_installchangelogs:  
    dh_installchangelogs FIXES
```

Si fas servir el nou **dh**, la utilització explícita d'objectius com els llistats a Secció 4.4.1 (excepte `get-orig-source`) pot dificultar la correcta comprensió dels seus efectes. Si us plau, limita l'ús d'objectius explícits a objectius del tipus `override_dh_*` i de manera que siguin completament independents entre ells (sempre que sigui possible).

²⁷Els fitxers `debian/changelog` i `debian/NEWS` sempre s'instal·len automàticament. També es busca el fitxer de canvis de l'autor canviant el nom a minúscules i per la seva coincidència amb `changelog`, `changes`, `changelog.txt`, i `changes.txt`.

Capítol 5

Altres fitxers del directori `debian`.

Per tal de controlar el treball de `debhelper` en la compilació del paquet, pots afegir fitxers de configuració en el directori `debian`. En aquest capítol es resumeix el que pots fer amb cada un d'ells i el seu format. Consulta «[Debian Policy Manual](http://www.debian.org/doc/devel-manuals#policy)» (<http://www.debian.org/doc/devel-manuals#policy>) i «[Debian Developer's Reference](http://www.debian.org/doc/devel-manuals#devref)» (<http://www.debian.org/doc/devel-manuals#devref>) per a les directrius de construcció de paquets.

L'ordre `dh_make` construeix alguns fitxers de configuració com a plantilles i els col·loca en el directori `debian`. Alguns d'ells tenen el sufix `.ex` (de «example») en el nom. Altres tenen com a prefix (del nom) `nom_de_l_paquet` que estàs construint. Mira el contingut de tots ells ¹.

En altres casos, `dh_make` no pot construir plantilles de configuració per a `debhelper`. En aquests casos tu mateix escriuràs els fitxers amb un editor.

Si vols fer servir els fitxers generats en la construcció del paquet, fes el següent:

- elimina els sufix `.ex` o `.EX` de tots els fitxers de plantilla.
- reanomena els fitxers de configuració fent servir el nom del fitxer del paquet binari en lloc de `nom_de_l_paquet`.
- modifica el contingut dels fitxers de plantilla per adaptar-los a les teves necessitats.
- elimina els fitxers que no necessitis.
- fes les modificacions necessàries en el fitxer `control` (consulta Secció 4.1).
- modifica el fitxer `rules` (consulta Secció 4.4) si és necessari.

Els fitxers de configuració construïts per `debhelper` que no tenen el prefix `nom_de_l_paquet` com és ara `install` s'aplicaran al primer paquet binari. Si hi ha més d'un paquet binari, les seves configuracions s'especificaran amb el prefix del paquet binari corresponent al seu nom: `paquet-1.install`, `paquet-2.install`, etc.

5.1 `README.Debian` (LLEGEIX-ME.debian)

Qualsevol detall extra o discrepància entre el programa original i la seva versió Debian cal documentar-la en aquest arxiu.

`dh_make` en genera un predeterminat, que és el següent:

```
gentoo for Debian
-----
<possible notes regarding this package - if none, delete this file>
-- Josip Rodin <joy-mg@debian.org>, Wed, 11 Nov 1998 21:02:14 +0100
```

Atès que no tenim res per explicar, eliminarem el fitxer. Consulta `dh_installdocs(1)`.

¹En aquest capítol, es farà referència als arxius ubicats en el directori `debian` sense afegir el prefix `debian/` per a simplificar el text i sempre que no hi hagi possibilitat de confusió.

5.2 Fitxer compat.

The `compat` file defines the `debhelper` compatibility level. Currently, you should set it to the `debhelper` v10 as follows:

```
$ echo 10 > debian/compat
```

You may use `compat` level v9 in certain circumstances for compatibility with older systems. However, using any level below v9 is not recommended and should be avoided for new packages.

5.3 Fitxer conffiles.

Una de les coses més molestes dels programes és quan després de dedicar molt de temps i esforç a configurar un programa (com a usuari), una actualització del programa fa inútils els canvis. Debian resol aquesta situació marcant els fitxers de configuració² de manera que quan actualitzes un paquet, tens l'opció de mantenir la teva configuració.

Des de la versió 3 de `debhelper`, `dh_installdeb(1)` considera *automàticament* a tots els fitxers ubicats en el directori `/etc` com fitxers de configuració, de manera que si tots els fitxers de configuració del programa són en aquest directori, no es necessari que els incloguis en el fitxer «conffiles» (abreviatura d'arxiu de configuració). Per a la majoria de paquets, l'única ubicació dels arxius de configuració és `/etc` i en aquest cas pots eliminar aquest arxiu.

Si el teu programa fa servir fitxers de configuració però també els sobreescriu ell mateix (directament des del codi font) es millor no marcar-los com «conffiles». Si ho fas, `dpkg` informarà als usuaris que revisin els canvis realitzats en aquests fitxers en cada actualització.

Si el programa que empaqueta requereix que cada usuari modifiqui els fitxers de configuració del directori `/etc`, hi ha dues formes per a no marcar-los com fitxers «conffiles» i que no siguin manipulats per `dpkg`:

- Construir un enllaç simbòlic dels fitxers ubicats en `/etc` que apunti als fitxers ubicats en el directori `/var` generats per *guions del desenvolupador* («maintainer scripts»).
- Posar els fitxers generats pels *guions del desenvolupador* en el directori `/etc`.

Si vols més informació sobre els *guions del desenvolupador* consulta Secció 5.18.

5.4 Fitxers `nom_del_paquet.cron.*`

Si el teu paquet requereix la realització de tasques periòdiques per funcionar correctament, pots fer servir aquest arxiu. Pots establir la realització de tasques cada hora, dia, setmana o mes o qualsevol altre període de temps. Els noms dels fitxers són:

- `nom_del_paquet.cron.hourly` - instal·lats com `/etc/cron.hourly/nom_del_paquet`: s'executen cada hora.
- `nom_del_paquet.cron.daily` - instal·lats com `/etc/cron.daily/nom_del_paquet`: s'executen cada dia, habitualment a primera hora del matí.
- `nom_del_paquet.cron.weekly` - instal·lats com `/etc/cron.weekly/nom_del_paquet`: s'executen cada setmana, habitualment en el matí del diumenge.
- `nom_del_paquet.cron.hourly` - instal·lats com `/etc/cron.hourly/nom_del_paquet`: s'executen cada hora.
- `nom_del_paquet.cron.d` - instal·lats com `/etc/cron.d/nom_del_paquet`: per a qualsevol altre període de temps.

Els fitxers tenen el format de guions shell. Amb l'excepció dels fitxers `nom_del_paquet.cron.d` que s'ajustaran al format descrit a `crontab(5)`.

No és necessari explicitar un arxiu `cron.*` per a configurar la rotació de registres, per aquest cas consulta `dh_installogrotate(1)` i `logrotate(8)`. Elimina aquest arxiu si no s'ha de fer servir amb el teu paquet.

²Consulta `dpkg(1)` i *Debian Policy Manual*, "D.2.5 Conffiles" (<http://www.debian.org/doc/debian-policy/ap-pkg-controlfields.html#s-pkg-f-Conffiles>).

5.5 Fitxer `dirs`.

Aquest arxiu especifica els directoris necessaris però que per alguna raó no s'han creat en el procés d'instal·lació normal (`make install DESTDIR=...` executat per `dh_auto_install`). Generalment això és degut a un problema en el fitxer `Makefile`.

Els fitxers llistats en el fitxer `install` no requereixen la creació prèvia dels directoris. Consulta Secció 5.11.

És recomanable executar en primer lloc la instal·lació i només fer ús d'aquest arxiu si es produeix algú problema. No s'ha de posar la barra inicial en els noms dels directoris llistats a l'arxiu `dirs`.

5.6 Fitxer `nom_del_paquet.doc-base`.

Si el teu paquet té documentació a més de les pàgines de manual i d'informació, pots fer servir el fitxer `doc-base` per a registrar-la i així l'usuari la trobarà més fàcilment amb `dhhelp(1)`, `dwww(1)` o `doccentral(1)`.

La documentació inclourà fitxers HTML, PS i PDF ubicats en `/usr/share/doc/nom_del_paquet/`.

Aquest és el contingut del fitxer `gentoo.doc-base` de `gentoo`:

```
Document: gentoo
Title: Gentoo Manual
Author: Emil Brink
Abstract: This manual describes what Gentoo is, and how it can be used.
Section: File Management
Format: HTML
Index: /usr/share/doc/gentoo/html/index.html
Files: /usr/share/doc/gentoo/html/*.html
```

Per obtenir informació sobre el format del fitxer, consulta `install-docs(8)` i en el manual Debian `doc-base` en la còpia local `/usr/share/doc/doc-base/doc-base.html/index.html` proporcionat pel paquet `doc-base`.

Per a més detalls sobre la instal·lació de documentació addicional, consulta Secció 3.3.

5.7 Fitxer `docs`.

Aquest arxiu especifica els noms dels fitxers de documentació que `dh_installdocs(1)` instal·larà en el directori temporal.

L'opció predeterminada inclourà tots els fitxers existents en els directoris del nivell més alt del codi amb els noms `BUGS`, `README*`, `TODO` etc.

També n'inclouré alguns d'altres per a `gentoo`:

```
BUGS
CONFIG-CHANGES
CREDITS
ONEWS
README
README.gtkrc
TODO
```

5.8 Fitxer `emacsens-*`.

Si el teu paquet proporciona fitxers Emacs que es poden compilar a bytes en el moment de la instal·lació, pots fer servir aquests fitxers.

Aquests fitxers seran instal·lats en el directori temporal per `dh_installemacsens(1)`.

Elimina'ls si no els necessites.

5.9 Fitxer `nom_del_paquet.examples`.

L'ordre `dh_installexamples(1)` instal·la els fitxers i directoris llistats en aquest arxiu com a fitxers d'exemple.

5.10 Fitxers `nom_del_paquet.init` `nom_del_paquet.default`

Si el teu paquet és un dimoni que s'executa en iniciar-se el sistema, és que no has fet cas a la meva recomanació inicial, o no? :-)

El fitxer `nom_del_paquet.init` s'instal·la com un guió en `/etc/init.d/nom_del_paquet` pel guió «`init`» que engega i atura un dimoni. Es tracta d'una plantilla genèrica construïda per l'ordre `dh_make` amb el nom `init.d.ex`. Caldrà canviar-li el nom i fer molts canvis per assegurar-te que compleix amb les capçaleres de l'estàndard base de Linux (LSB, consulta [Linux Standard Base \(http://www.linuxfoundation.org/collaborate/workgroups/lsb\)](http://www.linuxfoundation.org/collaborate/workgroups/lsb)). `dh_installinit(1)` ho instal·la en el directori temporal.

The `package.default` file will be installed as `/etc/default/package`. This file sets defaults that are sourced by the `init` script. This `package.default` file is most often used to set some default flags or timeouts. If your `init` script has certain configurable features, you can set them in the `package.default` file, instead of in the `init` script itself.

Si el programa original inclou un arxiu pel guió «`init`», tu pots fer-ne ús o bé descartar-ho. Si decideixes no fer servir el guió «`init`» original, caldrà que en facis un de nou a `debian/nom_del_paquet.init`. Tot i així, si el guió «`init`» original té bona pinta i s'instal·la en la ubicació correcta, encara caldrà configurar el fitxer `rc*` d'enllaços simbòlics. Caldrà reescriure l'objectiu `dh_installinit` de l'arxiu `rules` com segueix:

```
override_dh_installinit:
    dh_installinit --onlyscripts
```

Elimina el fitxer si no el fas servir.

5.11 Fitxer `install`.

If there are files that need to be installed into your package but your standard `make install` won't do it, put the filenames and destinations into this `install` file. They are installed by `dh_install(1)`.³ You should first check that there is not a more specific tool to use. For example, documents should be in the `docs` file and not in this one.

En el fitxer `install` hi haurà una línia per a cada un dels fitxers a instal·lar, amb el seu nom (sempre relatiu al directori superior de la compilació) i després d'un espai, el directori d'instal·lació (relatiu al directori d'instal·lació). En el cas que no s'instal·li el fitxer binari `src/arxiu_binari` del `gentoo`, hauries d'escriure el següent:

```
src/arxiu_binari usr/bin
```

Quan s'instal·li el paquet, hi haurà una ordre executable `/usr/bin/arxiu_binari`.

En el fitxer `install` pots escriure el nom del fitxer sense el directori d'instal·lació sempre que la ruta relativa de directori no canviï. Aquest format es fa servir amb paquets grans que separen el resultat de la compilació en més d'un paquet binari fent servir el conjunt de fitxers `nom_del_paquet-1.install`, `nom_del_paquet-2.install`, etc.

L'ordre `dh_install` revisarà el directori `debian/tmp` per cercar els fitxers si no els troba en el directori actual (o en la ubicació establerta per a la recerca amb l'opció `--sourcedir`).

5.12 Fitxer `nom_del_paquet.info`.

Si el teu paquet fa servir fitxers «`info`» (sistema d'ajuda), pots instal·lar-los fent servir `dh_installinfo(1)` que utilitzarà el llistat del fitxer `package.info`.

³Aquesta ordre reemplaça l'ordre obsoleta `dh_movefiles(1)` que feia servir el fitxer `files`.

5.13 Fitxer `nom_del_paquet.links`.

Si has de generar enllaços simbòlics addicionals en el directori de compilació del paquet com a responsable del paquet, pots fer-ho fent servir `dh_link(1)` fent un llistat dels directoris complets de l'origen i destí dels fitxers a un fitxer `nom_del_paquet.links`.

5.14 Fitxers `{nom_arxiu.source/}lintian-overrides`

If lintian reports an erroneous diagnostic for a case where Debian policy allows exceptions to some rule, you can use `package.lintian-overrides` or `source/lintian-overrides` to quieten it. Please read the Lintian User's Manual (<https://lintian.debian.org/manual/index.html>) and refrain from abusing this.

El fitxer `nom_del_paquet.lintian-overrides` és per a un paquet binari amb el nom `nom_del_paquet` i s'instal·la a `usr/share/lintian/overrides/nom_del_paquet` per l'ordre **dh_lintian**.

El fitxer `source/lintian-overrides` és per als paquets de codi original i no s'instal·la.

5.15 Fitxers `manpage.*`.

Cal que el programa tengui una pàgina de manual. Si l'autor original no l'ha fet, l'hauràs de fer tu. L'ordre **dh_make** construeix alguns fitxers de plantilla per a les pàgines de manual. Caldrà editar-los i emplenar-los. Comprova que elimines els fitxers que no fas servir.

5.15.1 Fitxers `manpage.1.ex`

Las pàgines de manual s'escriuen normalment amb `nroff(1)`. L'exemple `manpage.1.ex` està escrit amb **nroff**. Consulta la pàgina de manual `man(7)` on s'explica com editar el fitxer.

El nom del fitxer de manual ha d'incloure el nom del programa corresponent: així canviaràs el nom de `manpage` a `gentoo`. El nom del fitxer inclou `.1` com a sufixe: aquest és el codi que indica que la pàgina de manual correspon a un programa d'usuari. Comprova que és la secció correcta. Aquí tens una petita llista de les seccions de les pàgines de manual.

Secció	Descripció	Notes
1	Ordres d'usuari	Ordres o guions executables
2	Crides del sistema	Funcions proporcionades pel nucli
3	Crides a biblioteques	Funcions de les biblioteques del sistema
4	Arxiu especials	En general es troben a <code>/dev</code>
5	Formats d'arxiu	p.ex. format de <code>/etc/passwd</code>
6	Jocs	Jocs o altres programes d'entreteniment
7	Paquets de macros	Com ara les macros mc .
8	Administració del sistema	Programes que només sol executar el superusuari
9	Rutines del nucli	Crides al sistema no estàndard i internes

En el cas del `gentoo`, el nom del fitxer serà `gentoo.1`. Com que no hi havia una pàgina de manual en els fitxers de fonts de `gentoo.1`, en vaig escriure una canviant el nom de la plantilla `manpage.1.ex` per `gentoo.1` i editant-la fent servir la informació de l'exemple i la documentació del programador original.

Fes servir l'ordre **help2man** per generar la pàgina de manual fent servir la informació aportada per les opcions «`-help`» i «`-version`» de cada programa ⁴.

⁴Cal tenir present que el marcadore de posició de pàgines de manual **help2man** reclamarà que la informació més detallada estigui disponible en el sistema d'informació. Si l'ordre no troba una pàgina **info**, caldrà que editis manualment la pàgina de manual construïda per **help2man**.

5.15.2 Fitxer `manpage.sgml.ex`.

Si prefereixes fer servir el format SGML en lloc de **nroff** pots fer servir la plantilla `manpage.sgml.ex`. En aquest cas hauràs de fer el següent:

- reanomenar el fitxer com `gentoo.sgml`.
- instal·lar el paquet `docbook-to-man`
- afegir `docbook-to-man` en el camp `Build-Depends` de l'arxiu `control`
- afegir l'objectiu `override_dh_auto_build` en el fitxer `rules`:

```
override_dh_auto_build:
    docbook-to-man debian/gentoo.sgml > debian/gentoo.1
dh_auto_build
```

5.15.3 Fitxer `manpage.xml.ex`.

I si prefereixes treballar en format XML en lloc de SGML, pots fer servir la plantilla `manpage.xml.ex`. En aquest cas hauràs de fer el següent:

- reanomenar el fitxer a `gentoo.1.xml`
- instal·lar el paquet `docbook-xsl` i un processador XSLT com `xsltproc` (recomanat)
- afegir els paquets `docbook-xsl`, `docbook-xml` i `xsltproc` a la línia de `Build-Depends` en el fitxer `control`
- afegir l'objectiu `override_dh_auto_build` en el fitxer `rules`:

```
override_dh_auto_build:
    xsltproc --nonet \
        --param make.year.ranges 1 \
        --param make.single.year.ranges 1 \
        --param man.charmap.use.subset 0 \
        -o debian/ \
    http://docbook.sourceforge.net/release/xsl/current/manpages/docbook.xsl\
    debian/gentoo.1.xml
dh_auto_build
```

5.16 Fitxer `nom_del_paquet.manpages`

Si el teu paquet té pàgines de manual, les instal·laràs amb `dh_installman(1)` llistant els fitxers corresponents en el fitxer `nom_del_paquet.manpages`.

Per instal·lar el fitxer `docs/gentoo.1` del paquet `gentoo` com el seu manual, escriu en el fitxer `gentoo.manpages` el següent:

```
docs/gentoo.1
```

5.17 Fitxer `NEWS`.

L'ordre `dh_installchangelogs(1)` instal·la aquest arxiu.

5.18 Fitxers {pre,post}{inst,rm}

Els arxius `postinst`, `preinst`, `postrm`, i `prerm`⁵ s'anomenen *guions del desenvolupador* («maintainer scripts»), i són guions que es col·loquen en l'àrea de control del paquet i són executats pel paquet **dpkg** quan el teu paquet s'instal·la, s'actualitza o és eliminat.

Hauries d'evitar editar manualment els *guions del desenvolupador* degut a la seva complexitat. Consulta «[Debian Policy Manual, 6 "Package maintainer scripts and installation procedure"](http://www.debian.org/doc/debian-policy/ch-maintainerscripts.html)» (<http://www.debian.org/doc/debian-policy/ch-maintainerscripts.html>) i examina els exemples contruïts per **dh_make**.

Si malgrat les meves advertències, adaptes els *guions del desenvolupador* per al teu paquet, comprova el seu funcionament en la **instal·lació, actualització, desinstal·lació i eliminació completa**.

Les actualitzacions a una nova versió cal que siguin «silencioses» i no intrusives (els usuaris només haurien de veure que s'han arreglat els errors i afegit noves funcions).

Quan l'actualització és necessàriament intrusiva (p. ex. fitxers de configuració dispersos en diversos directoris amb una estructura totalment modificada), s'haurien d'establir els valors predeterminats segurs (p. ex. desactivar els serveis) i facilitar la documentació apropiada establerta en les normes (fitxers `README.Debian` i `NEWS.Debian`) com a darrer recurs. Cal evitar molestar a l'usuari amb notes **debconf** invocades pels *guions del desenvolupador* en les actualitzacions.

El paquet **ucf** facilita el sistema *conffile-like* per preservar els canvis de configuració realitzats per l'usuari i per això no han de anomenar-se com *conffiles* els fitxers gestionats pels *guions del desenvolupador*. Així es minimitzen les incidències associades amb ells.

Aquests *guions del desenvolupador* són un exemple de les característiques de Debian que expliquen **perquè la gent elegeix Debian**. Has de ser acurat amb no molestar amb ells.

5.19 Fitxer `nom_del_paquet.symbols`.

Packaging of a library is not easy for a novice maintainer and should be avoided. Having said it, if your package has libraries, you should have `debian/package.symbols` files. See Secció [A.2](#).

5.20 Fitxer `TODO`.

L'ordre `dh_installdocs(1)` instal·la aquest arxiu.

5.21 Fitxer `watch`.

The `watch` file format is documented in the `uscan(1)` manpage. The `watch` file configures the **uscan** program (in the `devscripts` package) to watch the site where you originally got the source. This is also used by the [Debian Package Tracker](https://tracker.debian.org/) (<https://tracker.debian.org/>) service.

Aquest és el seu contingut:

```
# watch control file for uscan
version=3
http://sf.net/gentoo/gentoo-(.+)\.tar\.gz debian uupdate
```

Amb aquest arxiu `watch`, l'URL `http://sf.net/gentoo` es descarrega i es busca els enllaços de tipus ``. El nom base (just la part després del / final) d'aquests enllaços URL es comparen amb la funció «regexp» de Perl (consulta `perlre(1)`) amb el patró `gentoo-(.+)\.tar\.gz`. Dels fitxers trobats que s'ajusten al patró, es descarrega el de la versió més recent i el programa **uupdate** s'executa per actualitzar el codi original.

⁵Encara que faig servir les expressions **bash** abreujades `{pre,post}{inst,rm}` pels noms d'aquests arxius, has de fer servir la sintaxi pura POSIX per a aquests guions del desenvolupador per mantenir la compatibilitat el sistema «shell» **dash**.

Although this is true for other sites, the SourceForge download service at <http://sf.net> is an exception. When the `watch` file has a URL matching the Perl regexp `^http://sf\.net/`, the `uscan` program replaces it with `http://qa.debian.org/watch/sf.` and then applies this rule. The URL redirector service at <http://qa.debian.org/> is designed to offer a stable redirect service to the desired file for any `watch` pattern of the form `http://sf.net/project/tar-name-(.+)\.tar\.gz`. This solves issues related to periodically changing SourceForge URLs.

Si l'autor original publica la signatura criptogràfica del fitxer tarball, es recomana verificar la seva autenticitat fent servir l'opció `pgpsigur` `lman` `le` descrita a `uscan(1)`.

5.22 Fitxer `source/format`.

El fitxer `debian/source/format`, només conté una línia que indica el format de construcció del paquet (consulta `dpkg-source(1)` on hi ha la llista completa). Després de `squeeze` hauria de ser:

- 3.0 (native) per a paquets nadius de Debian o
- 3.0 (quilt) per als altres paquets.

El nou format 3.0 (quilt) registre els canvis realitzats en el paquet en fitxers de pegats **quilt** en el directori `debian/patches`. Aquests canvis s'aplique automàticament en l'extracció dels fitxers originals del paquet ⁶. Les modificacions es desen en el fitxer `debian.tar.gz` que conté tots els fitxers del directori `debian` utilitzat en la construcció del paquet. El nou format permet la inclusió d'arxius com els icones PNG sense necessitat de trucs ⁷.

Quan **dpkg-source** extreu un paquet amb les fonts amb el format 3.0 (quilt), automàticament s'apliquen tots els pegats listats en l'arxiu `debian/patches/series`. Això ho pots evitar amb l'opció `--skip-patches`.

5.23 Fitxer `source/local-options`.

When you want to manage Debian packaging activities under a VCS, you typically create one branch (e.g., `upstream`) tracking the upstream source and another branch (e.g., typically `master` for Git) tracking the Debian package. For the latter, you usually want to have unpatched upstream source with your `debian/*` files for the Debian packaging to ease merging of the new upstream source.

Un cop compilat el paquet, els pegats s'hauran aplicat a les fonts. Hauràs de desfer els pegats executant `dquilt pop -a` abans de sincronitzar-les amb la branca `master`. Pots automatitzar aquesta tasca afegint l'arxiu opcional `debian/source/local-options` el contingut del qual serà `unapply-patches`. Aquest arxiu no s'inclou en el paquet font generat i únicament canvia l'entorn local de construcció. Aquest fitxer també pot contenir la línia `abort-on-upstream-changes` (consulta `dpkg-source(1)`).

```
unapply-patches
abort-on-upstream-changes
```

5.24 Fitxer `source/options`

Els arxius generats automàticament a l'arbre del codi font poden ser bastant molests en la construcció de paquets ja que generen arxius grans de pegats. Hi ha mòduls personalitzats, com ara **dh_autoreconf** per alleujar aquest problema com es descriu a Secció 4.4.3.

Pots proporcionar una expressió regular en Perl per a l'opció `--extend-diff-ignore` de `dpkg-source(1)` per fer cas omís dels canvis realitzats en els arxius generats automàticament en crear el paquet font.

⁶Consulta [DebSrc3.0](http://wiki.debian.org/Projects/DebSrc3.0) (<http://wiki.debian.org/Projects/DebSrc3.0>) on es resumeix la informació sobre els formats 3.0 (quilt) i 3.0 (native).

⁷Actualment, el nou format també permet treballar amb més d'un arxiu «tar» amb les fonts originals i altres sistemes de compressió. Aquestes funcions no s'expliquen en aquest document.

Pots emmagatzemar aquestes opcions de l'ordre **dpkg-source** a l'arxiu `source/options` de les fonts del paquet com a solució genèrica per a fer front a aquest problema dels arxius generats automàticament. A l'exemple, s'evita la creació d'arxius de pegat pels arxius `config.sub`, `config.guess` i `Makefile`.

```
extend-diff-ignore = "(^|/)(config\.sub|config\.guess|Makefile)$"
```

5.25 Fitxers patches/*.

L'antic format 1.0 construïa un arxiu `diff.gz` amb el contingut dels fitxers de construcció del paquet del directori `debian` i els canvis a realitzar en les fonts. Aquest format per conservar els canvis era una mica enutjós quan es tractava d'inspeccionar i entendre cada modificació de les fonts. Ja no és eficaç.

El nou format 3.0 (**quilt**) de les fonts, desa les modificacions (els pegats) a fer als fitxers en el directori `debian/patches/` fent servir l'ordre **quilt**. Aquests pegats i altres dades del paquet ubicats en el directori `debian` es conserven en el fitxer `debian.tar.gz`. Des que l'ordre **dpkg-source** pot aplicar els pegats a les fonts amb el nou format 3.0 (**quilt**) sense el paquet **quilt**, no és necessari afegir el paquet **quilt** en el camp `Build-Depends` del fitxer `control` ⁸.

El funcionament de l'ordre **quilt** s'explica en [quilt\(1\)](#). Conserva les modificacions de les fonts en una col·lecció de fitxers de pegats -p1 en el directori `debian/patches` i les fonts originals romanen sense modificar fora del directori `debian`. L'ordre d'aplicació de les modificacions es conserva en el fitxer `debian/patches/series`. Pots executar («push»), desfer («pop») i actualitzar les modificacions fàcilment ⁹.

A Capítol 3, s'han construït tres fitxers de pegats en el directori `debian/patches`.

Com que els pegats s'ubiquen en `debian/patches`, comprova que has configurat correctament l'ordre **dquilt** com es descriu a Secció 3.1.

Quan algú (inclòs tu mateix), facilita un pegat `nom_pegat.patch` per a les fonts, una vegada construït el paquet, la modificació del paquet amb format 3.0 (**quilt**) és així de simple:

```
$ dpkg-source -x gentoo_0.9.12.dsc
$ cd gentoo-0.9.12
$ dquilt import ../foo.patch
$ dquilt push
$ dquilt refresh
$ dquilt header -e
... descripció de la modificació
```

Els pegats conservats amb el nou format de fonts 3.0 (**quilt**) han d'estar *exempts* de coses innecessàries. Pots comprovar-ho executant `dquilt pop -a; while dquilt push; do dquilt refresh; done`.

⁸Es fan servir altres mètodes d'aplicació de pegats a Debian. El sistema **quilt** és el recomanat. Altres sistemes són **dpatch**, **db**s, **cdb**s, etc. La majoria d'ells conserven els pegats en fitxers en el directori `debian/patches/`.

⁹Si has demanat a un patrocinador que afegixi el paquet al repositori Debian, aquest sistema de separació i documentació dels canvis és molt important per facilitar la revisió del paquet per part del patrocinador.

Capítol 6

Construir el paquet.

Ara hauríem d'estar preparats per construir el paquet.

6.1 Reconstrucció completa.

Per realitzar correctament la construcció (o la reconstrucció) completa d'un paquet, comprova que els següents paquets estan instal·lats:

- el paquet `build-essential`.
- els paquets llistats en el camp `Build-Depends` del fitxer «control» (consulta Secció 4.1).
- els paquets llistats en el camp `Build-Depends-indep` (també de l'arxiu «control», consulta Secció 4.1).

Accedeix al directori principal del codi font del programa i executa la següent ordre:

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

Aquesta ordre farà totes les tasques necessàries per construir els paquets binaris i de fonts. Més concretament:

- neteja l'arbre del codi (`debian/rules clean`).
- construeix el paquet de codi font (`dpkg-source -b`).
- compila el programa (`debian/rules build`).
- construeix el paquet binari (`fakeroot debian/rules binary`)
- genera el fitxer `.dsc`
- genera l'arxiu `.changes`, fent servir l'ordre **dpkg-genchanges**

If the build result is satisfactory, sign the `.dsc` and `.changes` files with your private GPG key using the **debsign** command. You need to enter your secret pass phrase, twice. ¹

Quan acabi el procés de construcció d'un paquet no nadiu Debian (p.ex.gentoo), veuràs els següents fitxers en el directori superior al directori de treball (`~/gentoo/`):

¹Aquesta clau GPG ha de ser signada per un desenvolupador de Debian per connectar-se a la web de confiança i cal registrar-la a l'anell de claus de Debian (<http://keyring.debian.org>) . Això permet que els paquets siguin acceptats a l'arxiu de Debian. Consulta [Creating a new GPG key \(http://keyring.debian.org/-creating-key.html\)](http://keyring.debian.org/-creating-key.html) i [Debian Wiki on Keysigning \(http://wiki.debian.org/Keysigning\)](http://wiki.debian.org/Keysigning) .

- `gentoo_0.9.12.orig.tar.gz`

This is the original upstream source code tarball, merely renamed to the above so that it adheres to the Debian standard. Note that this was created initially by the command `dh_make -f ../gentoo-0.9.12.tar.gz`.

- `gentoo_0.9.12-1.dsc`

Aquest és un sumari dels continguts del codi font. Aquest arxiu es genera amb la informació del fitxer `control` i es fa servir quan es descomprimeixen les fonts amb `dpkg-source(1)`.

- `gentoo_0.9.12-1.debian.tar.gz`

Aquest arxiu comprimit té tots els fitxers del directori `debian`. Les modificacions de les fonts originals es conserven en els fitxers de pegats **quilt** en el directori `debian/patches`.

Si una altra persona vol tornar a construir el paquet des de l'inici, pot fer-ho fàcilment fent servir aquests tres fitxers. El procés d'extracció és trivial: només cal copiar-los en un directori i executar `dpkg-source -x gentoo_0.9.12-1.dsc`².

- `gentoo_0.9.12-1_i386.deb`

Aquest és el paquet binari Debian. Pots fer servir **dpkg** per instal·lar o eliminar aquest paquet o qualsevol altre.

- `gentoo_0.9.12-1_i386.changes`

Aquest arxiu conté la descripció dels canvis fets en la revisió actual del paquet, i el fan servir els programes de gestió FTP del repositori Debian per instal·lar-hi els paquets binaris i de fonts. Se genera parcialment amb l'arxiu `changelog` i el fitxer `.dsc`.

Amb el temps, el paquet anirà canviat i s'afegiran funcions. Les persones que descarreguin el paquet poden llegir aquest arxiu i assabentar-se dels canvis. Els programes de manteniment del repositori Debian també envien el contingut d'aquest arxiu a la llista de correu debian-changes-announce@lists.debian.org (<http://lists.debian.org/debian-devel-changes/>).

The `gentoo_0.9.12-1.dsc` and `gentoo_0.9.12-1_i386.changes` files must be signed using the **debsign** command with your private GPG key in the `~/gnupg/` directory, before uploading them to the Debian FTP archive. The GPG signature provides the proof that these files are really yours, using your public GPG key.

The **debsign** command can be made to sign with your specified secret GPG key ID (good for sponsoring packages) with the following in the `~/devscripts` file:

```
DEBSIGN_KEYID=el_teu_identificador_de_la_clau_GPG
```

Les llistes de números en els fitxers `.dsc` i `.changes` són les sumes MD5/SHA1/SHA256 dels fitxers. Les persones que descarreguen els arxius poden comprovar-los amb `md5sum(1)`, `sha1sum(1)` o `sha256sum(1)` i si els nombres no coincideixen sabran que el fitxer està corrupte o s'ha modificat.

6.2 «Autobuilder».

Debian manté diverses [adaptacions «ports»](http://www.debian.org/ports/) (<http://www.debian.org/ports/>) amb la [xarxa de servidors de compilació automàtica](http://www.debian.org/devel/build/) (<http://www.debian.org/devel/build/>) que executa dimonis **buildd** en ordinadors d'arquitectura diferent. Encara que tu no hauràs de fer res de tot això, has de conèixer el processament del teu paquet. Veurem com es processa el paquet per compilar-ho en diferents arquitectures³.

Els paquets del tipus `Architecture: any`, són construïts pel sistema de compilació automàtica. El sistema garanteix la instal·lació de:

- el paquet `build-essential`, i
- els paquets llistats en el camp `Build-Depends` (consulta Secció 4.1).

²Pots evitar l'aplicació automàtica dels pegats per **quilt** en els paquets amb el format 3.0 (**quilt**) al final de l'extracció amb l'opció `--skip-patches`. També pots optar per desfer les modificacions, una vegada acabada l'extracció, executant `dquilt pop -a`.

³El funcionament del sistema actual de compilació automàtica és més complicat del que s'explica en aquest document. Molts dels detalls del seu funcionament no són objectiu d'aquest document.

A continuació s'executa la següent ordre en el directori de les fonts:

```
$ dpkg-buildpackage -B
```

D'aquesta manera, s'executa tot el necessari per a la construcció del paquet binari depenent de l'arquitectura per a cada arquitectura. Fa el següent:

- neteja l'arbre del codi (`debian/rules clean`).
- compila el programa (`debian/rules build`).
- construeix el paquet binari per a l'arquitectura (`fakeroot debian/rules binary-arch`).
- signa l'arxiu `.dsc` fent servir l'ordre **gpg**
- genera i signa l'arxiu `.changes`, fent servir les ordres **dpkg-genchanges** i **gpg**

Així, el paquet estarà disponible per a altres arquitectures.

Encara que és necessari instal·lar els paquets llistats en el camp **Build-Depends-indep** per a la construcció de paquets normal (consulta Secció 6.1), el sistema automàtic de construcció no ho requereix degut a què només construeix paquets binaris dependents de l'arquitectura ⁴. Aquestes diferències entre l'empaquetament normal i l'automàtic determinen si els paquets requerits per a la compilació s'han de llistar en el camp **Build-Depends** o en el camp **Build-Depends-indep** del fitxer `debian/control` (consulta Secció 4.1).

6.3 L'ordre **debuild**.

Pots automatitzar encara més el procés de construcció de paquets de **dpkg-buildpackage** seguit de l'ordre **debuild**. Consulta `debuild(1)`.

The **debuild** command executes the **lintian** command to make a static check after building the Debian package. The **lintian** command can be customized with the following in the `~/devscripts` file:

```
DEBUILD_DPKG_BUILDPACKAGE_OPTS="-us -uc -I -i"  
DEBUILD_LINTIAN_OPTS="-i -I --show-overrides"
```

Per exemple, netejar el codi i reconstruir el paquet des d'un compte d'usuari és tan simple com:

```
$ debuild
```

Pots eliminar els fitxers generats en la compilació executant:

```
$ debuild -- clean
```

6.4 El paquet **pbuilder**.

El paquet **pbuilder** és molt útil per aconseguir un entorn net (**chroot**) on comprovar les dependències ⁵. Això assegura una construcció neta des del codi per a la construcció automàtica en la distribució **sid** per a diferents arquitectures i evita errades serioses del tipus FTBFS (Petada en la construcció des de les fonts o «Fail to Build From Source»), que són sempre de tipus RC (errades crítiques per a la publicació o «release critical») ⁶.

És possible personalitzar el funcionament del paquet **pbuilder** en els següents aspectes:

⁴A diferència a com funciona el paquet **pbuilder** (que s'explicarà més endavant), l'entorn **chroot** del paquet **sbuilder** fet servir pel sistema automàtic no té una instal·lació mínima de paquets del sistema i pot deixar molts de paquets instal·lats.

⁵Com que el paquet **pbuilder** està en evolució, comprova la configuració actual consultant la documentació.

⁶Consulta <http://buildd.debian.org/> per a més informació sobre el sistema de construcció automatitzada de paquets Debian.

- permet que el teu usuari tingui permís d'escriptura en el directori `/var/cache/pbuilder/result`.
- genera un directori, p. ex. `/var/cache/pbuilder/hooks`, amb permís d'escriptura per al teu usuari. En aquest directori posaràs els guions «hook».
- Estableix els fitxers `~/.pbuilder` o `/etc/pbuilder` amb el següent contingut:

```
AUTO_DEBSIGN=${AUTO_DEBSIGN:-no}
HOOKDIR=/var/cache/pbuilder/hooks
```

Ara pots inicialitzar el sistema local **pbuilder chroot** per primera vegada executant:

```
$ sudo pbuilder create
```

Si treballes amb un paquet font complet, executa les següents ordres en el directori on tinguis els fitxers `nom_del_paquet.orig.tar.gz`, `nom_del_paquet.debian.tar.gz` i `nom_del_paquet.dsc` per actualitzar el sistema local **pbuilder chroot** i, a continuació, compilar el paquet binari:

```
$ sudo pbuilder --update
$ sudo pbuilder --build nom_del_paquet.dsc
```

En acabar la compilació, el paquet compilat estarà en el directori `/var/cache/pbuilder/result/` i tu en seràs el propietari.

La signatura GPG dels arxius `.dsc` i `.changes` es poden generar de la següent manera:

```
$ cd /var/cache/pbuilder/result/
$ debsign nom_del_paquet_versió_arquitectura.changes
```

Si en lloc d'iniciar la construcció del paquet a partir d'un paquet ja construït, disposes d'un directori amb les fonts originals actualitzades, hauràs d'executar les següents ordres des del directori de les fonts originals (on hi haurà el directori `debian` amb el seu contingut):

```
$ sudo pbuilder --update
$ pdebuild
```

Pots «connectar-te» a l'entorn **chroot** executant l'ordre `pbuilder --login --save-after-login` i configurar-lo per adaptar-lo a les teves necessitats. Aquest entorn pot desar-se sortint del «shell» amb `^D` (Control-D).

La darrera versió de l'ordre **lintian** pot executar-se automàticament en l'entorn **chroot** fent servir el guió «hook» disponible a `/var/cache/pbuilder/hooks/B90lintian` i configurat de la següent manera ⁷:

```
#!/bin/sh
set -e
install_packages() {
    apt-get -y --allow-downgrades install "$@"
}
install_packages lintian
echo "+++ lintian output +++"
su -c "lintian -i -I --show-overrides /tmp/build/*.*changes" - pbuilder
# use this version if you don't want lintian to fail the build
#su -c "lintian -i -I --show-overrides /tmp/build/*.*changes; :" - pbuilder
echo "+++ end of lintian output +++"
```

Has de tenir un entorn **sid** actualitzat per construir correctament paquets per a **sid**. En realitat, la versió **sid** pot tenir errades que no fan recomanable la migració del teu sistema en aquesta versió. El paquet **pbuilder** t'ajuda a fer front en aquesta situació.

You may need to update your **stable** packages after their release for **stable-proposed-updates**, **stable/updates**, etc. ⁸ For such occasions, the fact that you may be running a **sid** system is not a good enough excuse for failing to update them

⁷Se suposa que la variable d'entorn `HOOKDIR=/var/cache/pbuilder/hooks` ja està configurada. Tens exemples a `/usr/share/doc/pbuilder/examples`.

⁸Hi ha restriccions per a aquestes actualitzacions del teu paquet per a la versió **stable**.

promptly. The `pbuilder` package can help you to access environments of almost any Debian derivative distribution of the same architecture.

Consulta <http://www.netfort.gr.jp/~dancer/software/pbuilder.html>, `pdebuild(1)`, `pbuilder(8)`, i `pbuilder(8)`.

6.5 git-buildpackage command and similar

Si l'autor original fa servir un sistema de gestió de versions per al codi (VCS) ⁹ pots considerar fer-ho servir. Així facilites la coordinació i la selecció dels pegats per a les fonts. Debian disposa de diversos paquets de guions especialitzats en cada tipus de VCS:

- `git-buildpackage`: conjunt per a la compilació de paquets en repositoris «Git».
- `svn-buildpackage`: programes d'ajuda per al manteniment de paquets Debian amb «Subversion».
- `cvs-buildpackage`: conjunt de paquets de guions Debian per a estructures de directoris CVS.

L'ús de `git-buildpackage` està esdevenint bastant popular entre els desenvolupadors Debian per la gestió dels paquets Debian amb el servidor «Git» a alioth.debian.org (<http://alioth.debian.org/>). ¹⁰ Aquest paquet ofereix moltes ordres per *automatitzar* el manteniment dels paquets:

- `gbp-import-dsc(1)`: import a previous Debian package to a Git repository.
- `gbp-import-orig(1)`: import a new upstream tar to a Git repository.
- `gbp-dch(1)`: generate the Debian changelog from Git commit messages.
- `git-buildpackage(1)`: construeix els paquets Debian des del repositori «Git».
- `git-pbuilder(1)`: construeix els paquets Debian des del repositori «Git» fent servir **pbuilder/cowbuilder**.

Aquestes ordres fan servir 3 branques per gestionar el procés d'empaquetat:

- `main` pels directoris de fonts del paquet Debian.
- `upstream` pels directoris de les fonts de l'autor.
- `pristine-tar` pel fitxer comprimit «.tar» de l'autor generat per l'opció `--pristine-tar`.¹¹

Pots configurar `git-buildpackage` amb `~/ .gbp.conf`. Consulta `gbp.conf(5)`. ¹²

⁹Consulta [Version control systems](http://www.debian.org/doc/manuals/debian-reference/ch10#_version_control_systems) (http://www.debian.org/doc/manuals/debian-reference/ch10#_version_control_systems).

¹⁰[Debian wiki Alioth](http://wiki.debian.org/Alioth) (<http://wiki.debian.org/Alioth>) documenta com fer servir el servei alioth.debian.org (<http://alioth.debian.org/>).

¹¹The `--pristine-tar` option invokes the **pristine-tar** command, which can regenerate an exact copy of a pristine upstream tarball using only a small binary delta file and the contents of the tarball that are typically kept in an `upstream` branch in the VCS.

¹²Aquí tens alguns recursos web disponibles per al públic expert.

- Construcció de paquets Debian amb `git-buildpackage` (</usr/share/doc/git-buildpackage/manual-html/gbp.html>)
- Paquets Debian amb «git» (https://honk.sigxcpu.org/piki/development/debian_packages_in_git/)
- Fent servir «Git» en la construcció de paquets Debian (<http://www.eyrie.org/~eagle/notes/debian/git.html>)
- «git-dpm»: paquets Debian amb el gestor «Git» (<http://git-dpm.alioth.debian.org/>)
- Fent servir «TopGit» per a generar sèries «quilt» en la construcció de paquets Debian (http://git.debian.org/?p=collab-maint/topgit.git;a=blog_plain;f=debian/-HOWTO-tg2quilt;hb=HEAD)

6.6 Reconstrucció ràpida.

Amb un paquet gran, pot ésser que no vulguis re-compilar des de l'inici cada vegada que fas algun canvi en el fitxer `debian/rules`. Per fer proves, pots construir un arxiu `.deb` sense re-compilar les fonts executant ¹³:

```
$ fakeroot debian/rules binary
```

O simplement pots comprovar si el paquet se compila amb:

```
$ fakeroot debian/rules build
```

Un cop has acabat la posada a punt, caldrà reconstruir el paquet amb el procediment explicat anteriorment. No podràs enviar al repositori els fitxers `.deb` construïts d'aquesta manera.

6.7 Jerarquia d'ordres

A continuació teniu un resum ràpid de la jerarquia de les ordres de construcció de paquets. Hi ha diversos procediments per fer el mateix.

- `debian/rules` = guió del responsable del paquet per a la compilació
- `dpkg-buildpackage` = nucli de l'eina de compilació de paquets
- `debuild` = `dpkg-buildpackage` + `lintian` (compilació en un entorn amb les variables controlades)
- `pbuilder` = nucli de l'entorn «chroot» de Debian
- `pdebuild` = `pbuilder` + `dpkg-buildpackage` (compilació en l'entorn «chroot»)
- `cowbuilder` = execució accelerada de `pbuilder`
- `git-pbuilder` = sintaxis fàcil de fer servir de l'ordre `pdebuild` (la fa servir `gbp buildpackage`)
- `gbp` = gestiona els fitxers Debian en un repositori «git»
- `gbp buildpackage` = `pbuilder` + `dpkg-buildpackage` + `gbp`

Tot i que fer servir ordres d'alt nivell com a `gbp buildpackage` i `pbuilder` assegura un entorn de compilació de paquets perfecte, és important entendre com funcionen les ordres de baix nivell com les executades per `debian/rules` i `dpkg-buildpackage` en aquest entorn.

¹³Environment variables that are normally configured to proper values are not set by this method. Never create real packages to be uploaded using this **quick** method.

Capítol 7

Com comprovar el teu paquet per trobar errors.

Cal que coneguis els procediments per comprovar el paquet i localitzar les errades abans d'enviar-lo als repositoris públics.

Provar el paquet en una màquina distinta de la que has fet servir en la construcció és una magnífica idea. Cal que posis atenció en tots els errors detectats en les proves que s'expliquen a continuació.

7.1 Canvis sospitosos

Si trobes un nou arxiu de pegat auto-generat com `debian-changes-*` en el directori `debian/patches` després de compilar el teu paquet no nadiu Debian en el format 3.0 (`quilt`), el més probable és que has canviat algun arxiu per accident o el guió de compilació ha modificat les fonts originals. Si és un error teu, ho corregeixes. Si l'error l'ha causat el guió, corregeix la causa de l'error amb **dh-autoreconf** com a Secció 4.4.3 o bé fes proves amb els arxius `source/options` com s'ha explicat a Secció 5.24.

7.2 Comprovació de la instal·lació del paquet

You must test your package for whether it installs without problems. The `debi(1)` command helps you to test installing all the generated binary packages.

```
$ sudo debi gentoo_0.9.12-1_i386.changes
```

To prevent installation problems on different systems, you must make sure that there are no filenames conflicting with other existing packages, using the `Contents-i386` file downloaded from the Debian archive. The **apt-file** command may be handy for this task. If there are collisions, please take action to avoid this real problem, whether by renaming the file, moving a common file to a separate package that multiple packages can depend on, using the alternatives mechanism (see `update-alternatives(1)`) in coordination with the maintainers of other affected packages, or declaring a `Conflicts` relationship in the `debian/control` file.

7.3 Comprovar els guions del desenvolupador («maintainer scripts»).

Ja s'ha comentat que els *guions del desenvolupador* (els fixers `preinst`, `prerm`, `postinst` i `postrm`) són complicats, excepte si s'utilitzen els generats pel paquet `debhelper`. No es recomana la seva utilització als desenvolupadors principiants (consulta Secció 5.18).

Si el paquet utilitza *guions del desenvolupador* modificats, has de comprovar el seu funcionament en la instal·lació, desinstal·lació, eliminació i actualització. Alguns dels problemes en aquests *guions del desenvolupador* només es detecten en l'eliminació o desinstal·lació. Fes servir l'ordre **dpkg** per fer la comprovació:

```
$ sudo dpkg -r gentoo
$ sudo dpkg -P gentoo
$ sudo dpkg -i gentoo_versió-revisió_i386.deb
```

Segueix aquesta seqüència per a la comprovació:

- Instal·la la versió anterior del paquet (obligatori).
- Actualitza ara a la versió actual.
- Torna a la versió anterior (opcional).
- Desinstal·la el paquet.
- Instal·la la nova versió del paquet.
- Elimina'l.
- Instal·la'l de nou.
- Desinstal·la el paquet.

Si treballes en la construcció de la primera versió del paquet, construeix versions «fantasma» anteriors (és suficient canviar el número de versió) per realitzar les proves i prevenir problemes.

Recorda que si hi ha versions anteriors del paquet en el repositori Debian, els usuaris actualitzaran el paquet des de la versió anterior disponible (i aquesta versió pot ésser distinta en la versió estable i de proves). Realitza les comprovacions també amb aquestes versions.

Encara que no es garanteix la reinstal·lació d'una versió anterior, és preferible assegurar-se que és possible fer-ho sense generar problemes.

7.4 El paquet **lintian**.

Executa **lintian**(1) amb el teu arxiu de canvis `.changes`. L'ordre **lintian** executa diversos guions de comprovació del paquet per localitzar els errors més freqüents ¹.

```
$ lintian -i -I --show-overrides gentoo_0.9.12-1_i386.changes
```

Per suposat, canvia el nom de l'arxiu pel nom de l'arxiu `.changes` del teu paquet. Els missatges d'error de **lintian** es codifiquen amb una lletra inicial en la línia del missatge:

- **E**: error; per indicar violacions de les normes o un error en el paquet.
- **W**: advertència; per advertir d'una possible violació de les normes o error en el paquet (però pot ésser una falsa alarma).
- **I**: informació; informació sobre algun aspecte del paquet (que tal vegada és millorable).
- **N**: nota o anotació; per a missatges detallats que poden ajudar-te en la depuració del paquet.
- **O**: ignorat; per a missatges ignorats (segons la configuració dels fitxers `lintian-overrides`) però que s'emet degut a l'opció `--show-overrides`.

En el cas dels errors (línies amb «E:» inicial), llegeix l'explicació (línies «N:»), fes els canvis necessaris en el paquet o comprova que són avisos falsos. En aquest cas, fes que **lintian** els accepti com s'ha explicat a Secció 5.14.

Observa que pots construir el paquet amb **dpkg-buildpackage** i executar **lintian** només amb una ordre si fas servir **debuild**(1) o **pdebuild**(1).

¹No és necessari afegir a **lintian** l'opció `-i -I --show-overrides` si l'has inclòs en la configuració a `/etc/devscripts.conf` o `~/devscripts` com s'ha explicat a Secció 6.3.

7.5 L'ordre debc.

Pots veure una llista dels fitxers del paquet binari Debian executant l'ordre `debc(1)` així:

```
$ debc nom_del_paquet.changes
```

7.6 L'ordre debdiff.

Pots comparar el contingut de dos paquets de fonts Debian executant l'ordre `debdiff(1)` així:

```
$ debdiff versió_anterior.dsc nova_versió.dsc
```

Pots comparar la llista de fitxers de dos paquets binaris de Debian amb l'ordre `debdiff(1)` executant l'ordre següent:

```
$ debdiff versió_anterior.changes nova_versió.changes
```

Aquest programa és útil per comprovar que no hi ha fitxers canviats d'ubicació o eliminats per error, i que no s'ha realitzat cap altre canvi no desitjat en l'actualització del paquet.

7.7 L'ordre interdiff.

Pots comparar dos fitxers `diff.gz` amb l'ordre `interdiff(1)`. Això és útil per comprovar que el responsable del paquet no ha fet canvis inadvertits en l'actualització de paquets construïts amb el format 1.0. Executa:

```
$ interdiff -z versió_anterior.diff.gz nova_versió.diff.gz
```

El nou format 3.0 de les fonts conserva els canvis en arxius de pegats com es descriu a Secció 5.25. Pots veure els canvis de cada arxiu `debian/patches/*` fent servir l'ordre **interdiff**.

7.8 L'ordre mc.

Algunes de les operacions de comprovació del paquet explicades poden realitzar-se de manera molt intuïtiva si es fa servir un gestor de fitxers com `mc(1)`, que permet visualitzar el contingut del paquet `*.deb`, i dels fitxers `*.udeb`, `*.debian.tar.gz`, `*.diff.gz` i `*.orig.tar.gz`

Comprova que no hi hagi fitxers innecessaris o de tamany zero, tant en el binari com en el paquet font. De vegades, hi ha coses que no es varen netejar adequadament: caldrà ajustar el fitxer `rules` per arreglar això.

Capítol 8

Actualitzar el paquet.

Després del llançament del paquet, és possible que hakis d'actualitzar-ho aviat.

8.1 Nova revisió Debian del paquet.

Suposem que s'ha enviat un informe d'error del teu paquet amb el número #654321, i que descriu un problema que pots solucionar. Per construir una nova revisió del paquet, necessites:

- Si cal aplicar una nova modificació, executa:
 - `dquilt new nom_modificació.patch` per establir el nom de la modificació.
 - `dquilt add arxiu_a_modificar` per establir el fitxer al qual s'aplicarà la modificació.
 - Corregir el problema en el paquet de fonts degut a un error de l'autor.
 - `dquilt refresh` per desar els canvis realitzats en el fitxer del pegat `nom_modificació.patch`.
 - `dquilt header -e` per afegir la descripció;
- Si cal actualitzar una modificació ja existent, executa:
 - `dquilt pop nom_modificacio.patch` per tornar a cridar l'arxiu `nom_modificacio.patch`;
 - Corregir el problema existent en la versió incorrecta del fitxer del pegat `nom_modificació.patch`.
 - `dquilt refresh` per actualitzar `nom_modificació.patch`.
 - `dquilt header -e` per actualitzar la descripció a la capçalera de l'arxiu del pegat.
 - `while dquilt push; do dquilt refresh; done` per aplicar tots els pegats eliminant *les coses innecessàries*;
- Afegir la informació de la revisió a l'inici del fitxer `changelog` (del directori «Debian»), p. ex. executant `dch -i` o explícitament indicant el número de versió i revisió executant `dch -v versió-revisió`, i a continuació detallar els canvis realitzats amb un editor ¹.
- Incloure la descripció (breu) de l'error i la solució, seguida de la referència de la notificació de l'error amb (Closes: #654321). D'aquesta manera, l'informe d'error es «tancarà» *automàticament* pel sistema de manteniment del repositori de Debian quan el paquet sigui acceptat en el repositori.
- Repeteix els passos anteriors per a cada una de les modificacions realitzades actualitzant l'arxiu Debian `changelog` amb `dch` si és necessari.
- Repeteix el que vas fer a Secció 6.1 i Capítol 7.

¹Per escriure la data i hora en el format correcte, cal fer servir `LANG=C date -R`.

- Una vegada satisfet, canvia el valor de la distribució en el fitxer `change log` de `UNRELEASED` al valor de la distribució objectiu `unstable` (o bé `experimental`).²
- Upload the package as in Capítol 9. The difference is that this time, the original source archive won't be included, as it hasn't been changed and it already exists in the Debian archive.

One tricky case can occur when you make a local package, to experiment with the packaging before uploading the normal version to the official archive, e.g., `1.0.1-1`. For smoother upgrades, it is a good idea to create a `change log` entry with a version string such as `1.0.1-1-rc1`. You may unclutter `change log` by consolidating such local change entries into a single entry for the official package. See Secció 2.6 for the order of version strings.

8.2 Inspecció d'una nova versió de l'autor.

When preparing packages of a new upstream release for the Debian archive, you must check the new upstream release first.

Comença per llegir els fitxers `change log`, `NEWS` i qualsevol altre documentació on l'autor original expliqui els canvis realitzats en la nova versió.

Pots comprovar els canvis entre les fonts originals de la nova versió i de l'anterior per detectar qualsevol canvi sospitós de produir errors executant:

```
$ diff -uN nom_arxiu-versió_anterior nom_arxiu-nova_versió
```

Les modificacions realitzades en els fitxers generats per «Autotools» (`missing`, `aclocal.m4`, `config.guess`, `config.h.in`, `config.sub`, `configure`, `depcomp`, `install-sh`, `ltmain.sh` i `Makefile.in`) pots ignorar-les. Fins i tot pots eliminar-los abans d'executar `diff` en les fonts per inspeccionar-les.

8.3 Nova versió del programa font.

Si el paquet `nom_del_paquet` que examines està correctament empaquetat fent servir els nous formats 3.0 (native) o 3.0 (quilt), per empaquetar una nova versió de l'autor hauria d'ésser suficient copiar el directori `debian` de la versió anterior a la nova, i a continuació realitzar les adaptacions necessàries. Pots copiar el directori `debian` de la versió anterior a la nova versió executant `tar xvzf /ruta/a/nom_del_paquet_versió_anterior.debian.tar.gz` des del directori de les fonts de la nova versió³. Per descomptat, caldrà fer algunes tasques òbvies:

- Comprimir les fonts originals en el fitxer `nom_del_paquet_número_nova_versió.orig.tar.gz`.
- Actualitzar el fitxer `change log` del directori «`debian`» executant `dch -v número_nova_versió-1`.
 - Afegeix una nova línia amb el text `New upstream release` per indicar que es tracta d'una nova versió de les fonts originals.
 - Descriu breuement els canvis realitzats *en les fonts originals per l'autor* que solucionen errors informats i tanca els informes d'aquests errors afegint `Closes: #número_del_informe_error`.
 - Descriu breuement els canvis realitzats *a les fonts originals* pel desenvolupador (tu mateix en aquest cas) per solucionar els errors informats i tanca els informes afegint `Closes: #número_del_informe_error`.
- Executa `while dquilt push; do dquilt refresh; done` per aplicar els pegats eliminat *les coses innecessàries*.

Si les modificacions no s'executen correctament, inspecciona la situació (mira la informació dels fitxers `.rej`) com segueix:

- Si un dels pegats aplicats està integrat en les fonts originals:

²Si fas servir l'ordre `dch -r` per fer efectiu aquest darrer canvi, assegura't que deses el fitxer `change log` des del editor.

³Si la versió anterior del paquet està empaquetada amb l'antic format 1.0, pots fer el mateix executant `zcat /ruta/a/nom_del_paquet_versió_anterior.diff.gz | patch -p1` des del directori de les fonts de la nova versió.

- executa `dquilt delete` per eliminar-lo.
- Si un dels pegats entra en conflicte amb els canvis realitzats per l'autor en les fonts originals:
 - executa `dquilt push -f` per aplicar els pegats de la versió anterior per forçar els rebuigs (tindrà la informació dels rebuigs en els fitxers `truc.rej`).
 - Edita els fitxers `truc.rej` manualment per saber l'efecte que es pretén amb `truc.rej`.
 - Executa `dquilt refresh` per actualitzar el pegat.
- Continua fins a l'execució de `while dquilt push; do dquilt refresh; done`.

Pots automatitzar aquest procés fent servir l'ordre `uupdate(1)` com segueix:

```
$ apt-get source nom_del_paquet
...
dpkg-source: info: extracting nom_del_paquet in nom_del_paquet-número_versió_anterior
dpkg-source: info: unpacking nom_del_paquet-número_versió_anterior.orig.tar.gz
dpkg-source: info: applying nom_del_paquet-número_versió_anterior-1.debian.tar.gz
$ ls -F
nom_del_paquet-número_versió_anterior/
nom_del_paquet-número_versió_anterior-1.debian.tar.gz
nom_del_paquet-número_versió_anterior-1.dsc
nom_del_paquet-número_versió_anterior.orig.tar.gz
$ wget http://example.org/nom_del_paquet/nom_del_paquet-número_versió_actual.tar.gz
$ cd nom_del_paquet-número_versió_anterior
$ uupdate -v número_versió_actual ../nom_del_paquet-número_versió_actual.tar.gz
$ cd ../nom_del_paquet-número_versió_actual
$ while dquilt push; do dquilt refresh; done
$ dch
... documenta les modificacions realitzades
```

Si has configurat el fitxer `debian/watch` com s'ha explicat a Secció 5.21, no és necessari que executis l'ordre **wget**. Simplement, executa `uscan(1)` en el directori `nom_del_paquet-número_versió_anterior` en lloc de l'ordre **uupdate**. Així, es buscarà automàticament el fitxer de les fonts, es descarregarà en el teu ordinador i s'executarà l'ordre **uupdate** ⁴.

Pots fer el llançament de l'actualització del paquet repetint el que s'ha explicat en Secció 6.1, Capítol 7 Capítol 9.

8.4 Actualitzar el format del paquet.

Per actualitzar un paquet no és necessari actualitzar el format de construcció del paquet. Encara així, pots aprofitar tota la funcionalitat de `debhelper` i del format 3.0 fent el següent ⁵:

- Si necessites novament algun dels fitxers de plantilla eliminats, pots regenerar-los executant **dh_make** amb l'opció `--addmissing` des del directori de les fonts. A continuació, fes les modificacions ja explicades.
- Si el paquet no està actualitzat per fer servir la nova sintaxi **dh** de la versió 7+ de `debhelper` en el fitxer `debian/rules`, fes-ne l'actualització per fer servir **dh**. També hauràs d'actualitzar `debian/control`.
- Si actualitzes el fitxer `rules` construït pel mecanisme d'inclusió `Makefile` del sistema de compilació Debian (`cdb`s) a la nova sintaxi **dh**, llegeix els següents documents per entendre les variables de configuració `DEB_*`.
 - còpia local de `/usr/share/doc/cdb/cdb-doc.pdf.gz`
 - The Common Debian Build System (CDBS), FOSDEM 2009 (http://meetings-archive.debian.net/pub/debian-meetings/2009/fosdem/slides/The_Common_Debian_Build_System_CDBS/)

⁴Si l'ordre **uscan** descarrega les fonts però no executa l'ordre **uupdate**, cal modificar l'arxiu `debian/watch` afegint `debian uupdate` al final de l'URL del fitxer.

⁵Si la persona que patrocina el teu paquet o altres desenvolupadors fan objeccions a l'actualització del format del paquet, no val la pena entossudir-se a argumentar a favor. Hi ha altres coses més importants que atendre.

- Si treballes amb un paquet construït amb el format 1.0 sense l'arxiu `nom_del_paquet.diff.gz`, pots actualitzar-lo a la nova versió 3.0 (native) afegint el fitxer `debian/source/format` amb la línia 3.0 (native). Copia els altres fitxers del directori `debian/*`.
- Si treballes amb un paquet construït amb el format 1.0 amb el fitxer `nom_del_paquet.diff.gz`, pots actualitzar-lo al nou format 3.0 (native) afegint el fitxer `debian/source/format` amb la línia 3.0 (native). Copia els altres fitxers del directori `debian/*`. Importa el fitxer `nom_del_paquet.diff` generat amb l'ordre `filterdiff -z -x '*/debian/*' nom_del_paquet.diff.gz > nom_del_paquet.diff` al sistema **quilt** ⁶.
- Si el paquet s'ha construït amb un altre sistema de pegats com `dpatch`, `dbs` o `cdb`s, fent servir les opcions `-p0`, `-p1` o `-p2`, pots convertir-ho al format `quilt` fent servir el guió `deb3` at <http://bugs.debian.org/581186>.
- Si el paquet s'ha construït executant l'ordre `dh` amb l'opció `--with quilt` o amb `dh_quilt_patch` i `dh_quilt_unpatch`, elimina tot això i fes servir el nou format font 3.0 (quilt).

Has de consultar **DEP - Debian Enhancement Proposals** (<http://dep.debian.net/>) i adoptar les propostes ACCEPTADES.

Repassa la secció Secció 8.3 per si has de repetir alguns dels passos indicats en aquesta secció

8.5 Conversió a UTF-8

Si el documents originals fan servir una codificació antiga, actualitzar-los a la codificació **UTF-8** és una bona idea.

- Fes servir `iconv(1)` per convertir codificacions de fitxers de text sense format.

```
iconv -f latin1 -t utf8 fitxer_original.txt > fitxer_convertit.txt
```

- Fes servir `w3m(1)` per convertir fitxers HTML a fitxers sense format en codificació UTF-8. Quan ho facis, assegura't que la configuració local fa servir UTF-8.

```
LC_ALL=en_US.UTF-8 w3m -o display_charset=UTF-8 \
    -cols 70 -dump -no-graph -T text/html \
    < fitxer_original.html > fitxer_convertit.txt
```

8.6 Recordatori per actualitzar paquets.

Here are a few reminders for updating packages:

- Conserva les entrades anteriors del fitxer `change log` (sona a obviat, però s'han donat casos d'executar `dch` en lloc de `dch -i`).
- Reconsidera els canvis en la construcció del paquet Debian: elimina les modificacions anteriors (sigui el que sigui) i recorda't d'afegir tot el necessari, sempre que no hi hagi una bona raó per no fer-ho.
- Si s'ha realitzat alguna modificació en la compilació (ho veuràs quan inspeccionis els canvis en les fonts originals) pot ser necessari actualitzar el fitxer `debian/rules` i les dependències de compilació en el fitxer `debian/control`.
- Comprova si hi ha alguna comunicació de pegats del paquet en el sistema de gestió d'errors (algun usuari pot haver construït i enviat un pegat que puguis fer servir) en **Debian Bug Tracking System (BTS)** (<http://www.debian.org/Bugs/>).
- Comprova el contingut del fitxer `.changes` per assegurar-te que envies el paquet a la distribució correcta, que els informes d'errors que es tanquen amb la nova versió estan llistats en el camp `Closes`, que el contingut dels camps `Maintainer` i `Changed-By` són correctes, que has signat el fitxer amb la teva clau GPG, etc.

⁶Pots fragmentar el fitxer `nom_del_paquet.diff` en diversos fitxers de pegats fent servir l'ordre `splitdiff`.

Capítol 9

Enviar el paquet.

Ara que has comprovat el teu paquet en profunditat, pots publicar-ho a un arxiu públic per compartir-ho.

9.1 Enviar el paquet al repositori de Debian.

Quan siguis desenvolupador oficial Debian ¹, podràs carregar el paquet al repositori de Debian ². Pots fer això manualment, però és més fàcil fer-ho amb les eines automàtiques ja disponibles com `dupload(1)` o `dput(1)`. A continuació descriurem com fer-ho amb **dupload** ³.

Per començar, caldrà ajustar la configuració de **dupload**. Pots fer-ho editant el fitxer general `/etc/dupload.conf`, o construint el teu arxiu personalitzat `~/dupload.conf` amb els paràmetres que vulguis canviar.

Consulta el manual de `dupload.conf(5)` per conèixer el significat de cada opció.

L'opció `$default_host` determina quina de les cues de càrrega es farà servir per defecte. `anonymous-ftp-master` és la primera però és possible que vulguis fer servir una altra més ràpida ⁴.

Si tens connexió a Internet, pots pujar el paquet executant:

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

dupload comprova que les sumes MD5/SHA1/SHA256 dels fitxers transferits coincideixen amb els llistats en el fitxer `.changes`, en cas contrari t'avisarà i hauràs de reconstruir el paquet com s'explica a Secció 6.1 per poder enviar-ho correctament.

Si trobes algun problema amb la pujada del paquet a <http://ftp.upload.debian.org/pub/UploadQueue/>, pots solucionar-ho pujant «manualment» (fent servir **ftp**) un arxiu `*.commands` signat amb GPG ⁵. Per exemple, aquest `hello.commands`:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
Uploader: Foo Bar <Foo.Bar@example.org>
Commands:
  rm hello_1.0-1_i386.deb
  mv hello_1.0-1.dsx hello_1.0-1.dsc
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.10 (GNU/Linux)
```

¹Consulta Secció 1.1.

²Hi ha arxius d'accés públic, com ara <http://mentors.debian.net/> que funcionen gairebé de la mateixa manera com arxiu de Debian i proporcionen un àrea de càrrega a les persones que no són DD. Pots construir un arxiu equivalent tu mateix fent servir les eines que figuren a <http://wiki.debian.org/HowToSetupADebianRepository>. Així, aquesta secció també és útil per a les persones que no són DD.

³Consulta Secció 1.1.

⁴See Debian Developer's Reference 5.6, "Uploading a package" (<http://www.debian.org/doc/manuals/developers-reference/pkgs.html#upload>).

⁵Consulta <http://ftp.upload.debian.org/pub/UploadQueue/README>. També pots fer servir l'ordre **dcut** del paquet `dput`.

```
[...]  
-----END PGP SIGNATURE-----
```

9.2 Incloure el fitxer `orig.tar.gz` per a la transferència del paquet al repositori.

A l'hora d'enviar per primera vegada el paquet al repositori, cal incloure l'arxiu `orig.tar.gz` amb les fonts originals. Si el número de revisió Debian del paquet no és `1` ni `0`, hauràs d'executar l'ordre **`dpkg-buildpackage`** amb l'opció `-sa` per forçar la inclusió de l'arxiu `orig.tar.gz`. Contrariament, amb l'opció `-sd` s'exclou l'arxiu `orig.tar.gz` de les fonts.

Per a l'ordre **`dpkg-buildpackage`**:

```
$ dpkg-buildpackage -sa
```

Per a l'ordre **`debuild`**:

```
$ debuild -sa
```

Per a l'ordre **`pdebuild`**:

```
$ pdebuild --debbuildopts -sa
```

Per altra banda, amb l'opció `-sd` s'evita l'inclusió de l'arxiu `orig.tar.gz` de les fonts originals.

9.3 Enviaments discontinuats.

If you created multiple entries in `debian/changelog` by skipping uploads, you must create a proper `*_changes` file that includes all changes since the last upload. This can be done by specifying the **`dpkg-buildpackage`** option `-v` with the version, e.g., `1.2`.

Per a l'ordre **`dpkg-buildpackage`**:

```
$ dpkg-buildpackage -v1.2
```

Per a l'ordre **`debuild`**:

```
$ debuild -v1.2
```

Per a l'ordre **`pdebuild`**:

```
$ pdebuild --debbuildopts "-v1.2"
```

Apèndix A

Tècniques avançades

Here are some hints and pointers for advanced packaging topics that you are most likely to deal with. You are strongly advised to read all the references suggested here.

Pot ésser necessari editar manualment els fitxers de plantilla per a la construcció del paquet generats per l'ordre **dh_make** per seguir els detalls tractats en aquest capítol. La nova ordre **debmake** gestiona millor aquestes etapes de la construcció de paquets.

A.1 Biblioteques compartides.

Abans de construir paquets de **biblioteques** compartides, has de llegir les següents referències bàsiques en detall:

- [Debian Policy Manual, 8 "Shared libraries"](http://www.debian.org/doc/debian-policy/ch-sharedlibs.html) (<http://www.debian.org/doc/debian-policy/ch-sharedlibs.html>)
- [Debian Policy Manual, 9.1.1 "File System Structure"](http://www.debian.org/doc/debian-policy/ch-opersys.html#s-fhs) (<http://www.debian.org/doc/debian-policy/ch-opersys.html#s-fhs>)
- [Debian Policy Manual, 10.2 L-Libraries](http://www.debian.org/doc/debian-policy/ch-files.html#s-libraries) (<http://www.debian.org/doc/debian-policy/ch-files.html#s-libraries>)

Heus aquí alguns consells simplistes per a que pugis començar.

- Les biblioteques compartides son fitxers objecte en format **ELF** que contenen codi compilat.
- Les biblioteques compartides es distribueixen com a fitxers ***.so** (ni fitxers ***.a** ni ***.la**).
- Les biblioteques compartides s'utilitzen principalment per compartir codi comú entre diversos executables fent servir l'ordre **ld**.
- Les biblioteques compartides, a vegades es fan servir per proporcionar complements («plugins») a un executable mitjançant el procediment **dlopen**.
- Shared libraries export **symbols**, which represent compiled objects such as variables, functions, and classes; and enable access to them from the linked executables.
- El **SONAME** (el nom lògic) de la biblioteca compartida `libnom_biblioteca.so.1`: `objdump -p libnom_biblioteca.so.1 | grep SONAME`¹
- El «SONAME» (nom lògic) d'una biblioteca compartida en general coincideix amb el nom del fitxer de la biblioteca (però no sempre).
- El «SONAME» (nom lògic) de les biblioteques compartides enllaçades a `/usr/bin/foo`: `objdump -p /usr/bin/foo | grep NEEDED`²

¹Alternativament: `readelf -d libnom_biblioteca.so.1 | grep SONAME`

²Alternativament: `readelf -d libnom_biblioteca.so.1 | grep NEEDED`

- `libnom_biblioteca1`: el paquet de biblioteca de la biblioteca compartida `libnom_biblioteca.so.1` amb la versió ABI del nom lògic («SONAME») 1.³
- Els guions del desenvolupador del paquet de la biblioteca han d'executar `ldconfig` en les circumstàncies específiques per generar els enllaços simbòlics necessaris per a «SONAME» (nom lògic).⁴
- `libfoo1-dbg`: the debugging symbols package that contains the debugging symbols for the shared library package `libfoo1`.
- `libfoo-dev`: the development package that contains the header files etc. for the shared library `libfoo.so.1`.⁵
- Debian packages should not contain *.la Libtool archive files in general.⁶
- Debian packages should not use RPATH in general.⁷
- Encara que és una mica antiquat i és només una referència secundària, [Debian Library Packaging Guide](http://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html) (<http://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html>) encara pot ésser útil.

A.2 Gestionant `debian/nom_de_l_paquet.symbols`

When you package a shared library, you should create a `debian/package.symbols` file to manage the minimal version associated with each symbol for backward-compatible ABI changes under the same SONAME of the library for the same shared library package name.⁸ You should read the following primary references in detail:

- [Debian Policy Manual, 8.6.3 "The symbols system"](http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-symbols) (<http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-symbols>)⁹
- `dh_makeshlibs(1)`
- `dpkg-gensymbols(1)`
- `dpkg-shlibdeps(1)`
- `deb-symbols(5)`

Here is a rough example of how to create the `libfoo1` package from the upstream version 1.3 with the proper `debian/libfoo1.symbols` file:

- Prepara l'estructura de directoris Debian de les fonts fent servir el fitxer original de l'autor `libnom_biblioteca-1.3.tar.gz`
 - Si és la primera vegada que es construeix un paquet de `libnom_biblioteca1`, genera el fitxer `debian/libnom_biblioteca1.symbols` amb el contingut buit.
 - Si la versió 1.2 anterior de l'autor es va empaquetar en el paquet `libnom_biblioteca1` amb el fitxer `debian/libnom_biblioteca1.symbols` apropiat en el seu paquet font, fes-lo servir una altra vegada.
 - If the previous upstream version 1.2 was not packaged with `debian/libfoo1.symbols`, create it as the `symbols` file from all available binary packages of the same shared library package name containing the same SONAME of the library, for example, versions 1.1-1 and 1.2-1.¹⁰

³Consulta [Debian Policy Manual, 8.1 "Run-time shared libraries"](http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-runtime) (<http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-runtime>).

⁴Consulta [Debian Policy Manual, 8.1.1 "ldconfig"](http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-ldconfig) (<http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-ldconfig>).

⁵Consulta [Debian Policy Manual, 8.3 "Static libraries"](http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-static) (<http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-static>) i [Debian Policy Manual, 8.4 "Development files"](http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-dev) (<http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-dev>).

⁶Consulta [Debian wiki ReleaseGoals/LAFileRemoval](http://wiki.debian.org/ReleaseGoals/LAFileRemoval) (<http://wiki.debian.org/ReleaseGoals/LAFileRemoval>).

⁷Consulta [Debian wiki RpathIssue](http://wiki.debian.org/RpathIssue) (<http://wiki.debian.org/RpathIssue>).

⁸Els canvis ABI incompatibles amb versions anteriors, normalment requereixen l'actualització del «SONAME» (nom lògic) de la biblioteca i del paquet de la biblioteca compartida a d'altres nous.

⁹Per a biblioteques C++ i altres casos pels quals el maneig individual de símbols és difícil, es millor guiar-se per [Debian Policy Manual, 8.6.4 "The shlibs system"](http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-shlibdeps) (<http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-shlibdeps>).

¹⁰Totes les versions anteriors dels paquets Debian estan disponibles a <http://snapshot.debian.org/> (<http://snapshot.debian.org/>). La revisió Debian del paquet segueix a la versió per facilitar el manteniment de versions anteriors («backport») del paquet: 1.1<<1.1-1~bpo70+1<<1.1-1 i 1.2<<1.2-1~bpo70+1<<1.2-1

```
$ dpkg-deb -x libfoo1_1.1-1.deb libfoo1_1.1-1
$ dpkg-deb -x libfoo1_1.2-1.deb libfoo1_1.2-1
$ : > symbols
$ dpkg-gensymbols -v1.1 -plibfoo1 -Plibfoo1_1.1-1 -0symbols
$ dpkg-gensymbols -v1.2 -plibfoo1 -Plibfoo1_1.2-1 -0symbols
```

- Make trial builds of the source tree with tools such as **debuild** and **pdebuild**. (If this fails due to missing symbols etc., there were some backward-incompatible ABI changes that require you to bump the shared library package name to something like **libfoo1a** and you should start over again.)

```
$ cd libfoo-1.3
$ debuild
...
dpkg-gensymbols: warning: some new symbols appeared in the symbols file: ...
see diff output below
--- debian/libfoo1.symbols (libfoo1_1.3-1_amd64)
+++ dpkg-gensymbolsFE5gzx      2012-11-11 02:24:53.609667389 +0900
@@ -127,6 +127,7 @@
foo_get_name@Base 1.1
foo_get_longname@Base 1.2
foo_get_type@Base 1.1
+ foo_get_longtype@Base 1.3-1
foo_get_symbol@Base 1.1
foo_get_rank@Base 1.1
foo_new@Base 1.1
...
```

- If you see the diff printed by the **dpkg-gensymbols** as above, extract the proper updated **symbols** file from the generated binary package of the shared library. ¹¹

```
$ cd ..
$ dpkg-deb -R libfoo1_1.3_amd64.deb libfoo1-tmp
$ sed -e 's/1\..3-1/1\..3/' libfoo1-tmp/DEBIAN/symbols \
>libfoo-1.3/debian/libfoo1.symbols
```

- Construir paquets per distribuir amb eines com **debuild** i **pdebuild**.

```
$ cd libfoo-1.3
$ debuild -- clean
$ debuild
...
```

A més a més dels exemples anteriors, cal comprovar la compatibilitat ABI amb més atenció i actualitzar manualment les versions dels símbols (si és necessari). ¹²

Encara que només és una referència secundària, [Debian wiki UsingSymbolsFiles](http://wiki.debian.org/UsingSymbolsFiles) (<http://wiki.debian.org/UsingSymbolsFiles>) i els seus enllaços a altres pàgines web pot ésser d'utilitat.

A.3 Multi-arquitectura

La nova funció multi-arquitectura introduïda a la versió «wheezy» de Debian integra el suport per a la instal·lació en més d'una arquitectura dels paquets binaris (particularment a **i386** i **amd64**, però també amb altres combinacions) en **dpkg** i **apt**. És convenient que llegeixis les següents referències detalladament:

¹¹La revisió Debian es deriva de la versió per facilitar el manteniment de versions anteriors («backport») del paquet: **1.3 << 1.3-1-bpo70+1 << 1.3-1**

¹²Consulta [Debian Policy Manual](http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-updates), 8.6.2 "Shared library ABI changes" (<http://www.debian.org/doc/debian-policy/ch-sharedlibs.html#s-sharedlibs-updates>).

- [Ubuntu wiki MultiarchSpec](https://wiki.ubuntu.com/MultiarchSpec) (<https://wiki.ubuntu.com/MultiarchSpec>) (original)
- [Debian wiki Multiarch/Implementation](http://wiki.debian.org/Multiarch/Implementation) (<http://wiki.debian.org/Multiarch/Implementation>) (estat a Debian)

S'utilitzen triplets com `i386-linux-gnu` i `x86_64-linux-gnu` per als directoris d'instal·lació de les biblioteques compartides. El triplet actual s'estableix de forma dinàmica al valor `$(DEB_HOST_MULTIARCH)` per `dpkg-architecture(1)` a cada compilació. Per exemple, el directori d'instal·lació de les biblioteques multi-arquitectura es pot canviar de la següent manera.¹³

Directorí antic	directorí multi-arquitectura i386	directorí multi-arquitectura amd64
<code>/lib/</code>	<code>/lib/i386-linux-gnu/</code>	<code>/lib/x86_64-linux-gnu/</code>
<code>/usr/lib/</code>	<code>/usr/lib/i386-linux-gnu/</code>	<code>/usr/lib/x86_64-linux-gnu/</code>

Here are some typical multiarch package split scenario examples for the following:

- el codi font de la biblioteca `libnom_biblioteca-1.tar.gz`
- el codi font d'una ordre `bar-1.tar.gz` escrit en un llenguatge compilat
- el codi font d'una ordre `baz-1.tar.gz` escrit en un llenguatge interpretat

Paquet	Arquitectura	Multi-arquitectura:	Contingut del paquet
<code>libnom_del_paquet1</code>	any	same	la biblioteca compartida, és co-instal·lable
<code>libnom_del_paquet1-dbg</code>	any	same	els símbols de depuració de la biblioteca compartida, són co-instal·lables
<code>libnom_del_paquet1-dev</code>	any	same	els fitxers de capçalera i d'altres d'una biblioteca compilada, co-instal·lable
<code>libnom_del_paquet1-tools</code>	any	foreign	els programes de suport en temps d'execució, no són co-instal·lables
<code>libnom_del_paquet1-doc</code>	any	foreign	els fitxers de documentació de la biblioteca compartida
<code>bar</code>	any	foreign	els fitxers del programa compilat, no són co-instal·lables
<code>bar-doc</code>	all	foreign	els fitxers de documentació del programa
<code>baz</code>	all	foreign	els fitxers del programa interpretat

Cal tenir en compte que el paquet de desenvolupament ha de tenir un enllaç simbòlic a la biblioteca compartida associada **sense el número de versió**. P. ex.: `/usr/lib/x86_64-linux-gnu/libfoo.so -> libfoo.so.1`

A.4 Construint un paquet de biblioteca compartit

You can build a Debian library package enabling multiarch support using `dh(1)` as follows:

- Actualitza `debian/control`.
 - Add `Build-Depends: debhelper (>=10)` for the source package section.
 - Afegeix `Pre-Depends: ${misc:Pre-Depends}` per a cada paquet binari de biblioteca compartida.
 - Afegeix el camp `Multi-Arch`: per a cada secció de paquet binari.
- Set `debian/compat` to "10".
- Canvia el directori habitual `/usr/lib/` al directori multi-arquitectura `/usr/lib/$(DEB_HOST_MULTIARCH)/` per a tots els guions de la construcció del paquet.

¹³Old special purpose library paths such as `/lib32/` and `/lib64/` are not used anymore.

- Call `DEB_HOST_MULTIARCH` `DEB_HOST_MULTIARCH` in `debian/rules` to set the `DEB_HOST_MULTIARCH` variable first.
- Canvia `/usr/lib/` per `/usr/lib/$(DEB_HOST_MULTIARCH)/` a `debian/rules`.
- If `./configure` is used in part of the `override_dh_auto_configure` target in `debian/rules`, make sure to replace it with `dh_auto_configure -- .`¹⁴
- Canvia cada repetició de `/usr/lib/` per `/usr/lib/*/` als fitxers `debian/nom_del_paquet.install`
- Generate files like `debian/foo.links` from `debian/foo.links.in` dynamically by adding a script to the `override_dh_` target in `debian/rules`.

```
override_dh_auto_configure:
    dh_auto_configure
    sed 's/@DEB_HOST_MULTIARCH@/$(DEB_HOST_MULTIARCH)/g' \
        debian/nom_del_paquet.links.in > debian/nom_del_paquet.links
```

Comprova que el paquet de biblioteca compartida conté només els fitxers que s'esperava i que el paquet «-dev» continua funcionant correctament.

All files installed simultaneously as the multiarch package to the same file path should have exactly the same file content. You must be careful of differences generated by the data byte order and by the compression algorithm.

A.5 Paquet nadiu Debian

Si el paquet es construeix només per a Debian o fins i tot només per fer-ho servir en local, les seves fonts poden incloure tots els fitxers de `debian/*`. Hi ha dos mètodes per construir el paquet.

You can make the upstream tarball by excluding the `debian/*` files and package it as a non-native Debian package as in Secció 2.1. This is the normal way, which some people encourage using.

L'alternativa és seguir el model de treball pels paquets nadius Debian.

- Genera un paquet nadiu de fonts Debian en el format 3.0 (native) fent servir un únic arxiu comprimit en format «tar» afegint tots els arxius.

- `nom_del_paquet_versió.tar.gz`
- `nom_del_paquet_versió.dsc`

- Construcció de paquets binaris Debian des de paquets de fonts Debian nadius.

- `nom_del_paquet_versió_arquitectura.deb`

For example, if you have source files in `~/mypackage-1.0` without the `debian/*` files, you can create a native Debian package by issuing the **dh_make** command as follows:

```
$ cd ~/el_meu_paquet-1.0
$ dh_make --native
```

Then the `debian` directory and its contents are created just like in Secció 2.8. This does not create a tarball, since this is a native Debian package. But that is the only difference. The rest of the packaging activities are practically the same.

Després d'executar l'ordre **dpkg-buildpackage**, veuràs els següents fitxers en el directori superior:

¹⁴Alternativament, pots afegir els arguments `--libdir=\${prefix}/lib/$(DEB_HOST_MULTIARCH)` and `--libexecdir=\${prefix}/lib/$(DEB_HOST_MULTIARCH)` a `./configure`. Fitxa't que `--libexecdir` especifica el directori predeterminat per instal·lar programes executables que són engegats per altres programes més que no pas pels usuaris. El valor predeterminat per «Autotools» és `/usr/libexec/` però a Debian és `/usr/lib/`.

- `el_meu_paquet_1.0.tar.gz`

Aquest és el «tarball» del codi font generat a partir del directori `el_meu_paquet - 1.0` per l'ordre **`dpkg-source`** (el seu sufix no és `orig.tar.gz`).

- `el_meu_paquet_1.0.dsc`

This is a summary of the contents of the source code, as in the non-native Debian package. (There is no Debian revision.)

- `el_meu_paquet_1.0_i386.deb`

This is your completed binary package, as in the non-native Debian package. (There is no Debian revision.)

- `el_meu_paquet_1.0_i386.changes`

Aquest arxiu descriu els canvis realitzats en la versió actual del paquet de la mateixa forma que en els paquets Debian no nadius (no hi ha el codi de revisió Debian).