NAME

debmake − program to make a Debian source package

SYNOPSIS

debmake [**−h**] [**−c** | **−k**] [**−n** | **−a** *package−version***.orig.tar.gz** | **−d** | **−t**] [**−p** *package*] [**−u** *version*] [**−r** *revision*] [**−z** *extension*] [**−b** "*binarypackage, ...*]" [**−e** *foo@example.org*] [**−f** "*firstname lastname*"] [**−i** "*buildtool*" | **−j**] [**−l** *license_file*] [**−m**] [**−o** *file*] [**−q**] [**−s**] [**−v**] [**−w** "*addon, ...*"] [**−x** [01234]] [**−y**] [**−L**] [**−P**] [**−T**]

DESCRIPTION

debmake helps to build a Debian package from the upstream source. Normally, this is done as follows:

- The upstream tarball is downloaded as the *package−version***.tar.gz** file.
- It is untarred to create many files under the *package−version/* directory.
- debmake is invoked in the *package−version/* directory, possibly without any arguments.
- Files in the *package−version*/**debian/** directory are manually adjusted.
- **dpkg−buildpackage** (usually from its wrapper **debuild** or **pdebuild**) is invoked in the *package−version/* directory to make Debian packages.

Make sure to protect the arguments of the **−b**, **−f**, **−l**, and **−w** options from shell interference by quoting them properly.

optional arguments:

−h, **−−help**
show this help message and exit.

−c, **−−copyright**
scan source for copyright+license text and exit.

- **−c**: simple output style
- **−cc**: normal output style (similar to the **debian/copyright** file)
- **−ccc**: debug output style

−k, **−−kludge**
compare the **debian/copyright** file with the source and exit.

The **debian/copyright** file must be organized to list the generic file patterns before the specific exceptions.

- **−k**: basic output style
- **−kk**: verbose output style

−n, **−−native**
make a native Debian source package without **.orig.tar.gz**. This makes a "**3.0 (native)**" format package.

If you are thinking of packaging a Debian-specific source tree with **debian/*** in it into a native Debian package, please think otherwise. You can use the "**debmake −d −i debuild**" or "**debmake −t −i debuild**" commands to make a "**3.0 (quilt)**" format non−native Debian package. The only difference is that the **debian/changelog** file must use the non−native version scheme: *version−revision*. The non−native package is more friendly to downstream distributions.

−a *package−version***.tar.gz**, **−−archive** *package−version***.tar.gz**
use the upstream source tarball directly. (**−p**, **−u**, **−z**: overridden)

The upstream tarball may be specified as *package_version***.orig.tar.gz** and **tar.gz**, for all cases may be **tar.bz2**, or **tar.xz**.

If the specified upstream tarball name contains uppercase letters, the Debian package name is generated by converting them to lowercase letters.

If the specified argument is the URL (http://, https://, or ftp://) to the upstream tarball, the upstream tarball is downloaded from the URL using **wget** or **curl**.

−d, **−−dist**
run the “make dist” command equivalents first to generate the upstream tarball and use it.

The “**debmake −d**” command is designed to run in the *package/* directory hosting the upstream VCS with the build system supporting the “**make dist**” command equivalents. (automake/autoconf, Python distutils, …)

−t, **−−tar**
run the “**tar**” command to generate the upstream tarball and use it.

The “**debmake −t**” command is designed to run in the *package/* directory hosting the upstream VCS. Unless you provide the upstream version with the **−u** option or with the **debian/changelog** file, a snapshot upstream version is generated in the **0~%y%m%d%H%M** format, e.g., *0~1403012359*, from the UTC date and time. The generated tarball excludes the **debian/** directory found in the upstream VCS. (It also excludes typical VCS directories: **.git/ .hg/ .svn/ .CVS/**.)

−p *package*, **−−package** *package*
set the Debian package name.

−u *version*, **−−upstreamversion** *version*
set the upstream package version.

−r *revision*, **−−revision** *revision*
set the Debian package revision.

−z *extension*, **−−targz** *extension*
set the tarball type, *extension*=(**tar.gz**|**tar.bz2**|**tar.xz**). (alias: **z**, **b**, **x**).

−b "*binarypackage[:type],…*", **−−binaryspec** "*binarypackage[:type],…*"
set the binary package specs by the comma separated list of *binarypackage:type* pairs, e.g., in the full form “**foo:bin,foo−doc:doc,libfoo1:lib,libfoo1−dbg:dbg,libfoo−dev:dev**” or in the short form “**,−doc,libfoo1,libfoo1−dbg, libfoo−dev**”.

Here, *binarypackage* is the binary package name; and the optional *type* is chosen from the following *type* values:

- **bin**: C/C++ compiled ELF binary code package (any, foreign) (default, alias: **""**, i.e., *null−string*)
- **data**: Data (fonts, graphics, …) package (all, foreign) (alias: **da**)
- **dbg**: Debug symbol package (any, same) (alias: **db**) (deprecated for stretch and after since the −dbgsym package is automatically generated)
- **dev**: Library development package (any, same) (alias: **de**)
- **doc**: Documentation package (all, foreign) (alias: **do**)
- **lib**: Library package (any, same) (alias: **l**)
- **perl**: Perl script package (all, foreign) (alias: **pl**)
- **python**: Python script package (all, foreign) (alias: **py**)
- **python3**: Python3 script package (all, foreign) (alias: **py3**)
- **ruby**: Ruby script package (all, foreign) (alias: **rb**)

- **script**: Shell script package (all, foreign) (alias: **sh**)

The pair values in the parentheses, such as (any, foreign), are the **Architecture** and **Multi−Arch** stanza values set in the **debian/control** file.

In many cases, the **debmake** command makes good guesses for *type* from *binarypackage*. If *type* is not obvious, *type* is set to **bin**. For example, **libfoo** sets *type* to **lib**, and **font−bar** sets *type* to **data**, . . .

If the source tree contents do not match settings for *type*, the **debmake** command warns you.

−e *foo@example.org*, **−−email** *foo@example.org*
set e−mail address.

The default is taken from the value of the environment variable **$DEBEMAIL**.

−f "*firstname lastname*", **−−full name** "*firstname lastname*"
set the fullname.

The default is taken from the value of the environment variable **$DEBFULLNAME**.

−i "*buildtool*", **−−invoke** "*buildtool*"
invoke "*buildtool*" at the end of execution. *buildtool* may be "**dpkg−buildpackage**", "**debuild**", "**pdebuild**", "**pdebuild −−pbuilder cowbuilder**", etc.

The default is not to execute any program.

Setting this option automatically sets the **−−local** option.

−j, **−−judge**
run **dpkg−depcheck** to judge build dependencies and identify file paths. Log files are in the parent directory.

- *package***.build−dep.log**: Log file for **dpkg−depcheck**.
- *package***.install.log**: Log file recording files in the **debian/tmp** directory.

−l "*license_file,. . .*", **−−license** "*license_file,. . .*"
add formatted license text to the end of the **debian/copyright** file holding license scan results.

The default is to add **COPYING** and **LICENSE**, and *license_file* needs to list only the additional file names all separated by "**,**".

−m, **−−monoarch**
force packages to be non−multiarch.

−o *file*, **−−option** *file*
read optional parameters from *file*. (This is not for everyday use.)

The *file* parameter is sourced as the Python3 code at the end of **para.py**. For example, the package description can be specified by the following file.

```
para['desc'] = 'program short description'
para['desc_long'] = '''\
 program long description which you wish to include.
 .
 Empty line is space + .
 You keep going on . . .
'''
```

−q, **−−quitearly**

quit early before creating files in the **debian/** directory.

−s, **−−spec**
use upstream spec (setup.py for Python, etc.) for the package description.

−v, **−−version**
show version information.

−w "*addon,...*", **−−with** "*addon,...*"
add extra arguments to the **−−with** option of the **dh**(1) command as *addon* in **debian/rules**.

The *addon* values are listed all separated by "**,**", e.g., "**−w "python2,autoreconf""**.

For Autotools based packages, setting **autoreconf** as *addon* forces running "**autoreconf −i −v −f**" for every package build. Otherwise, **autotools−dev** as *addon* is used as the default.

For Autotools based packages, if they install Python programs, **python2** as *addon* is needed for packages with "**compat < 9**" since this is non−obvious. But for **setup.py** based packages, **python2** as *addon* is not needed since this is obvious and it is automatically set for the **dh**(1) command by the **debmake** command when it is required.

−x *n*, **−−extra** *n*
generate extra configuration files as templates.

The number *n* determines which configuration templates are generated.

- **−x0**: bare minimum configuration files. (default if these files exist already)
- **−x1**: all **-x0** files + desirable configuration files. (default for new packages)
- **−x2**: all **-x1** files + interesting configuration files. (recommended for experts, multi binary aware)
- **−x3**: all **-x2** files + unusual configuration template files with an extra **.ex** suffix to ease their removal. (recommended for new users) To use these as configuration files, rename their file names to ones without the **.ex** suffix.
- **−x4**: all **-x3** files + copyright file examples.

−y, **−−yes**
"force yes" for all prompts. (without option: "ask [Y/n]"; doubled option: "force no")

−L, **−−local**
generate configuration files for the local package to fool **lintian**(1) checks.

−P, **−−pedantic**
pedantically check auto−generated files.

−T, **−−tutorial**
output tutorial comment lines in template files.

EXAMPLES

For a well behaving source, you can build a good−for−local−use installable single Debian binary package easily with one command. Test install of such a package generated in this way offers a good alternative to the traditional "**make install**" command into the **/usr/local** directory since the Debian package can be removed cleanly by the "**dpkg −P** ..." command. Here are some examples of how to build such test packages. (These should work in most cases. If the **−d** option does not work, try the **−t** option instead.)

For a typical C program source tree packaged with autoconf/automake:

- **debmake −d −i debuild**

For a typical Python module source tree:

- **debmake −s −d −b":python" −i debuild**

For a typical Python module in the *package−version***.tar.gz** archive:

- **debmake −s −a** *package−version***.tar.gz −b":python" −i debuild**

For a typical Perl module in the *Package−version***.tar.gz** archive:

- **debmake −a** *Package−version***.tar.gz −b":perl" −i debuild**

HELPER PACKAGES

Packaging may require installation of some additional specialty helper packages.

- Python3 programs may require the **dh−python** package.
- The Autotools (Autoconf + Automake) build system may require **autotools−dev** or **dh−autoreconf** package.
- Ruby programs may require the **gem2deb** package.
- Java programs may require the **javahelper** package.
- Gnome programs may require the **gobject−introspection** package.
- etc.

CAVEAT

debmake is meant to provide template files for the package maintainer to work on. Comment lines started by **#** contain the tutorial text. You should remove or edit such comment lines before uploading to the Debian archive.

The license extraction and assignment process involves a lot of heuristics; it may fail in some cases. It is highly recommended to use other tools such as **licensecheck** from the **devscripts** package in conjunction with **debmake**.

There are some limitations for what characters may be used as a part of the Debian package. The most notable limitation is the prohibition of uppercase letters in the package name. Here is a summary as a set of regular expressions:

- Upstream package name (**−p**): [−+.a−z0−9]{2,}
- Binary package name (**−b**): [−+.a−z0−9]{2,}
- Upstream version (**−u**): [0−9][−+.:˜a−z0−9A−Z]*
- Debian revision (**−r**): [0−9][+.˜a−z0−9A−Z]*

See the exact definition in Chapter 5 − Control files and their fields in the “Debian Policy Manual”.

debmake assumes relatively simple packaging cases. So all programs related to the interpreter are assumed to be "**Architecture: all**". This is not always true.

DEBUG

Please report bugs to the **debmake** package using the **reportbug** command.

The character set in the environment variable **$DEBUG** determines the logging output level.

- **i**: print information
- **p**: list all global parameters
- **d**: list parsed parameters for all binary packages
- **f**: input filename for the copyright scan

- **y**: year/name split of copyright line
- **s**: line scanner for format_state
- **b**: content_state scan loop: begin−loop
- **m**: content_state scan loop: after regex match
- **e**: content_state scan loop: end−loop
- **c**: print copyright section text
- **l**: print license section text
- **a**: print author/translator section text
- **k**: sort key for debian/copyright stanza
- **n**: scan result of debian/copyright (“**debmake −k**”)

Use this as:

```
$ DEBUG=pdfbmeclak debmake . . .
```

See README.developer in the source for more.

AUTHOR

LICENSE

Expat License

SEE ALSO

The **debmake−doc** package provides the “Guide for Debian Maintainers” in plain text, HTML and PDF formats under the **/usr/share/doc/debmake−doc/** directory.

Also, please read the original Debian New Maintainers’ Guide provided by the the **maint−guide** package.

See also **dpkg−source**(1), **deb−control**(5), **debhelper**(7), **dh**(1), **dpkg−buildpackage**(1), **debuild**(1), **quilt**(1), **dpkg−depcheck**(1), **pdebuild**(1), **pbuilder**(8), **cowbuilder**(8), **gbp−buildpackage**(1), **gbp−pq**(1), and **git−pbuilder**(1) man pages.