How Software Producers Can Distribute Their Products Directly in DEB Format

Aaron Isotton

`<aaron@isotton.com>`

Table of Contents

Introduction .. 1
Copyright and License .. 1
Credits / Contributors .. 2
Feedback .. 2
The Path to Take .. 2
Packaging Your Product .. 2
Doing it Yourself .. 2
Have Somebody Else Do it for You .. 3
Distributing Packages Yourself .. 3
Creating a Package Repository .. 3
Adding Your Packages to Debian .. 3
What You Get .. 3
The DFSG .. 4
Filing an ITP .. 4
Getting a Sponsor and Uploading .. 5
A. Special Cases .. 5
Special Notes for Packages Outside Debian .. 5
The Three Distributions of Debian .. 5
Versioned Dependencies .. 5
Linking Statically .. 6
Depending on Packages Which Are Not in Stable .. 6
Creating Debian Packages on Other Operating Systems .. 6
Converting a package from another format to DEB .. 6
Making it Easy to Create both RPM and DEB Packages .. 7

Abstract

This document explains how software producers can integrate their products open or closed source with Debian.

Introduction

This document is intended as a starting point to explain how software producers can integrate their products with Debian, what different situations can arise depending on the license of the products and the choices of the producers, and what possibilities there are. It does not explain how to create packages, but it links to documents which do exactly that. You should read this if you are not familiar with the big picture of creating and distributing Debian packages, and optionally with adding them to the Debian distribution.

This document's master location is http://www.isotton.com/debian/docs/distribute-deb/.

Copyright and License

Credits / Contributors

In this document, I have the pleasure of acknowledging (in alphabetic order):

- *Andrew Shugg* for aspelling the whole thing, and for telling me not to make fun of punch cards :)
- *Stuart Young* for the idea how to depend on packages which are not in stable, and for pointing out the whole RPM issue.
- and many others.

Feedback

Feedback is most certainly welcome for this document. Send your additions, comments and criticisms to the following email address: `<aaron@isotton.com>`.

The Path to Take

To integrate a software product with Debian, first you need to package it. You can either do that yourself, or you can ask a Debian developer to do that for you. In either case, you'll end up with a package in DEB format; you can then distribute it yourself or add it to Debian, or both.

If you decide to add your packages to Debian and I strongly recommend you do that because it is better both for everybody you have to find out whether they are DFSG-compatible, to post an ITP and to ask a Debian developer to check and upload them.

If you don't add your packages to Debian, and they are many or frequently updated, you might want to set up a package repository to make installing and upgrading easier for your users.

Packaging Your Product

You can either package your product yourself, or, if your your product seems interesting enough to some Debian developer, get him to package it for you. Of course, a Debian developer will most probably not be interested in doing the work for you if you don't want to add the package to Debian in the end; it is also easier to find a packager for a DFSG-compatible product than for one which isn't.

Doing it Yourself

The packaging process is explained in detail in the following documents: the Debian Policy [http://www.debian.org/doc/debian-policy], the Debian New Maintainer's Guide [http://www.debian.org/doc/maint-guide] and the Debian Developer's Reference [http://www.debian.org/doc/developers-reference].

It is easier to build your packages under Debian, but it is possible to do that under other operating systems as well; if you really want to do that, read the section called "Creating Debian Packages on Other Operating Systems".

If you need help while packaging, you can subscribe to the appropriate mailing lists [http://www.debian.org/MailingLists/]; there are many competent and very helpful people there. There are very many different lists discussing different topics; the most interesting ones for you are probably `debian-devel` and `debian-mentors`.

Have Somebody Else Do it for You

If you don't want to do the packaging yourself, you can try convincing a Debian developer to do that for you. As most of them are very busy, you will only find one if he is personally interested in your package, and if you are prepared to add your package to Debian; in any case, you can try asking on `debian-devel`.

Distributing Packages Yourself

Distributing packages yourself can be very simple; basically, you only have to distribute the files which are generated during the package process. As a reminder, typically you get five types of files when generating a package, only three of which are important here. Your users need the .deb file to install the binary package, and the .orig.tar.gz and .diff.gz files to install the source of the package.

Whether you distribute your files on CD-ROM, online or on punch cards is your problem, but if your package is open source, do provide the source code, i.e. the `.orig.tar.gz` and `.diff.gz` files.

If you decide not to add your packages to Debian, and especially if it is closed-source, please read the section called "Special Notes for Packages Outside Debian", as there are some special precautions to take.

Creating a Package Repository

An alternative to this simple method is creating a package repository. A package repository is a directory with the DEBs of your packages and a few additional special files in it; any Debian user can add your package repository to his `sources.list` file, and then your packages will be listed to him together with all the others available in Debian. This is a good idea if you have many packages with complicated dependencies, or often updated packages. Of course you can also create a package repository if none of the above is the case, but you want to provide this additional service to your users.

How to create a package repository is documented in detail in the Debian Repository HOWTO [http://www.isotton.com/debian/docs/repository-howto/].

Adding Your Packages to Debian

What You Get

Once you have packaged your product, you can distribute it yourself, add it to Debian, or both. If you want to distribute your packages only yourself, you can either distribute the .deb files you got while packaging, or create a package repository. If you want to add your package to Debian, there are a few more steps to take.

Instead of simply uploading the .debs on your server (of course you can still do that, if you want), you upload your packages to the Debian archive. To be allowed to do that, you have to respect the Debian Policy, which tries to make sure that all the packages are installed and uninstalled in standardized ways, and that there are no conflicts among them (and many other things).

Every package which is in Debian is automatically added to the package list, and all users can easily install and uninstall it using the Debian package tools. Moreover, it will automatically be compiled for all different Debian platforms (IA32, IA64, Sparc and many others) if that is possible, and it will be added to the Debian Bug Tracking System (BTS), which makes it easy for everybody to report bugs or suggest improvements.

The package will be mirrored worldwide on way more independent servers than most companies can call their own, and, if it is in the *stable* distribution, it will also be added to the official CD images. There are also other Linux distributions which are based on Debian, and it is possible that your packages will be added to those as well.

This means that for thousands of users worldwide installing your package will be as simple as typing `apt-get install` *`package-name`*; that average users, novices and professionals will be equally able to report bugs and give suggestions; and that thousands people who had never heard of your product and company will be suddenly aware of their existence.

The DFSG

An important part of Debian is the DFSG, short for Debian Free Software Guidelines. It is a set of guidelines about the license of a package, and a package goes into different sections of Debian depending on whether its license is DFSG-compatible or not. If it is, it goes into the Debian *main* section; if it isn't, it goes into the *non-free* section; if it is DFSG-compatible but depends on other packages in the *non-free* section, it goes into *contrib*.

A package is basically DFSG-compatible if its source code is available, and if everybody is allowed to modify and distribute it; examples of compatible licenses are the GNU General Public License [http://www.gnu.org/copyleft/gpl.html], the BSD License [http://www.debian.org/misc/bsd.license] and the Artistic License [http://www.perl.com/pub/a/language/misc/Artistic.html]. Whether your license is among the ones listed or not, you should read the Debian Social Contract [http://www.debian.org/social_contract.html] to find out more about it.

All this might seem only a detail, but it isn't. Strictly speaking, the *non-free* and the *contrib* section are not parts of Debian; they are just a service from Debian to the users who need such packages. In fact, Debian users (especially the ones involved with the Debian Project) tend to prefer packages from *main* to the others; if they can choose among two similar packages from *main* and *non-free* and have no special motivation to prefer the latter, they will almost always install the former. It also quite often happens that if there is a package in *non-free* with a large user base and no one in *main* which could replace it, that the Debian developers start to actively look for or even to develop an alternative. If you are interested in a large user base or in the best support from the Debian developers, it is thus in your own interest to adapt your license if that is possible.

To some of you, the DFSG may seem some kind of half-religious institution with the only purpose of generating flame wars on the mailing lists; whether this is true or not, and independently of your opinion, respecting it will get you much more support than adding another cool feature to your software.

Filing an ITP

Before you start packaging, you should file an ITP (Intent To Package). That is a special bug report saying that you want to package some product. So the other developers know that you're working on it and will not start working on it, too.

You can easily file an ITP using the **reportbug** tool and specifying “wnpp” as the package you want to report a bug against.

Getting a Sponsor and Uploading

As long as you aren't an accepted Debian Developer, you cannot upload to the Debian archive directly. You have to package your product and to ask for a *sponsor* on the `debian-mentors` mailing list. A sponsor is a (generally) seasoned Debian developer who will check your packages and give you hints and support until he thinks they are ready to be accepted into Debian. Then he will upload them and they will be added to Debian. Once you've proved that you are able to create proper new packages and that you are willing to maintain them, you can also become a Debian developer, and maybe one day sponsor as well. However, this out of the scope of this document; if you are interested, you can read more about this subject in the Debian Developer's Reference [http://www.debian.org/doc/developers-reference/].

A. Special Cases

Special Notes for Packages Outside Debian

If your package is not in Debian, and especially if it is closed-source or its license doesn't allow the redistribution of modified sources, you should pay special care to a few subtleties of packaging to make it as flexible as possible.

The Three Distributions of Debian

Debian is divided into three distributions: stable, testing and unstable. Whenever a new package is added, or an existing package is updated, it goes into unstable. Once it has been in unstable for ten days without revealing serious bugs, it automatically moves into testing. When the release manager decides it's time for a new release, he declares the testing distribution as frozen. This means that no new packages can be added, and no existing ones may be updated. Only outstanding bugs may be fixed. Once he thinks that testing is ready to be released, it becomes stable and a new testing distribution is added. Notice that the distributions are also referred to as branches.

This means that the three distributions (but especially unstable and stable) are sometimes very different, and that they may (and generally will) contain a different set of packages and different versions of the same packages. Thus a single package may not be installable on all distributions, or might not work on all of them. This is no problem when a package is in Debian, as it will automatically move from unstable into testing and finally together with the other packages it depends on, but it certainly is a problem for packages outside Debian.

As unstable and testing change very frequently (daily), it is very difficult to keep up with them, especially since not all users do a daily upgrade of all packages, but may upgrade only some the packages. It is thus best to target stable, but keeping a few things in mind to ease the installation on the other distributions.

Versioned Dependencies

The single most important thing is to get the versioned dependencies right. If, for example, your package needs `libfoo` version 1.01 or greater, and the version in the current stable is 1.07, you should not depend on version 1.07, but on version 1.01 or greater.

This is very important because even if 1.07 is the last released version of `libfoo`, sooner or later a greater version will follow, and make it into unstable and testing. If your package depends on `libfoo` 1.07 (and no other version), it would suddenly become uninstallable on unstable and testing. It's much better to take the (relatively small) risk of creating a package which will install but not work properly on testing and unstable than being 100% sure that sooner or later it will just become uninstallable.

Still supposing that your package needs version 1.01 or greater, and that the current version in stable is 1.07, depending on 1.07 or greater would also be wrong. Sure, it would work in the current stable, but if you depend on the smallest possible version it might happen that your package can be installed and work on older versions of Debian too!

Linking Statically

If your package depends on a rarely-used library which is not packaged in Debian (for example a library of your company), or on a fast changing one, linking statically to it might be a good idea. This does not mean that you should link all possible libraries which might ever change in the next twenty years statically, just the ones you think might cause problems if linked dynamically. Just use common sense here.

Warning

Never link statically to libc.

Depending on Packages Which Are Not in Stable

If your package depends on another package which is available in unstable and/or testing, you can backport that package to stable (or create a new one from scratch), supply it together with your main package and depend on both packages the one currently in testing/unstable, and the one you backported using an or ('|') dependency. This way your package will use the original package in testing/unstable and your custom version in stable.

Creating Debian Packages on Other Operating Systems

It is possible to create Debian packages also on other (Unix-compatible) operating systems, but it is much more difficult and not a very good idea.

It's probably easier to set up a system with Debian as native OS than trying to emulate its functionality on another, but anyway, here's how to do it.

First, you need to obtain **debootstrap** for your OS, install it, read its documentation and execute it. It will create a whole Debian directory tree in some subdirectory of your choice. Once you've done that, you only have to **chroot** into that directory, and you can run a Debian system (or something very similar to a real Debian system, at least) under your other OS of choice.

Notice that this will only work for systems with proper support for Unix filesystems (with permissions, and everything), and with a **chroot** command. In other words, this will *not* work on Windows.

Converting a package from another format to DEB

If you want to package a product you've already packaged in another format, that might ease your task, especially if the package is simple.

There's a program called *alien* which can convert packages between different formats; as of this writing it supports Redhat RPM, Debian DEB, Stampede SLP, Slackware TGZ and Solaris PKG. You can use it to convert your package to DEB, but probably you'll have to do some editing by hand until it works correctly.

Of course, using alien on a well-working package in another format is no excuse not to test the Debian package you generated and not to read the appropriate documentation; alien is and will always be experimental software.

Alien was originally written for Debian and is available as Debian package [http://packages.debian.org/unstable/admin/alien.html], but it was ported to other platforms too.

Making it Easy to Create both RPM and DEB Packages

If you think of the potential packaging problems already when you are programming, you can make it much easier to create RPM, DEB and any other kind of packages.

One of the main problems are the installation and the uninstallation scripts. You should always separate them clearly from the program itself, and if possible write them in a modular way. If you do so, it'll be easy to adapt them to different platforms and packaging systems.

You should peruse the documentation for the different platforms and package systems to learn about the differences and the potential problems.