Debian Binary Package Building HOWTO

Chr. Clemens Lee

<clemens@kclee.de>

2002−11−30, $Date: 2005/08/09 20:49:46 $

Revision History

Revision 4.0	2005−08−09	Revised by: ccl
updated email address and added link to Turkish translation by Oguz Yarimtepe		
Revision 3.0	2003−12−19	Revised by: ccl
fixed two typos reported by Claudio Cattazzo		
Revision 2.0	2003−12−13	Revised by: ccl
applying Frank Lichtenheld's feedback		
Revision 1.0	2003−11−08	Revised by: ccl
first version		

This mini−HOWTO shows how to build a minimal Debian .deb package.

Table of Contents

1. Introduction

The intended use of such a newly created archive is to install it only on your own box, not to get them into the official Debian distribution. To follow the 'official' process, please study the Debian New Maintainers' Guide.

Normal Debian packages get a proper source package, including a debian/rules file which automates the steps involved in creating the binary package. Here we just show how to package a simple shell script or binary executable into a small binary package.

BTW, I assume you know how to use 'tar', 'man', and what a '.tar.gz' file and Debian is (and how to use an editor ;–), but I assume you have never touched programs like 'ar' or 'dpkg'.

1.1. Resources on the Web

The Debian Reference gives an excellent overview as well as detailed information for everything Debian specific.

The official document for creating your own Debian packages is the Debian New Maintainers' Guide.

2. Getting Started

From the Debian Reference 2.2.2 2002−11−30: "The internals of this Debian binary package format are described in the deb(5) manual page. Because this internal format is subject to change (between major releases of Debian), always use dpkg−deb(8) for manipulating .deb files."

From the dpkg−deb man page: "dpkg−deb packs, unpacks and provides information about Debian archives. .deb files can also be manipulated with `ar` and `tar` alone if necessary. Use dpkg to install and remove packages from your system."

You might find lots of example .deb files in directory '`/var/cache/apt/archives/`'. With '`dpkg−deb −I somepackage.deb`' you might get a general overview of what this package offers in particular. '`dpkg−deb −c somepackage.deb`' lists all files which will be installed.

List content of the .deb file with '`ar tv somepackage.deb`'. Use the 'x' option to extract the files.

3. Package Structure

Let's examine one example package a little bit closer. E.g. file 'parted_1.4.24-4_i386.deb' contains these three files:

```
$ ar tv parted_1.4.24-4_i386.deb
rw-r--r-- 0/0      4 Mar 28 13:46 2002 debian-binary
rw-r--r-- 0/0   1386 Mar 28 13:46 2002 control.tar.gz
rw-r--r-- 0/0  39772 Mar 28 13:46 2002 data.tar.gz
```

Now we can start to extract all files including the content of the tar files.

3.1. debian–binary

The content of this file is "2.0\n". This states the version of the deb file format. For 2.0 all other lines get ignored.

3.2. data.tar.gz

The 'data.tar.gz' file contains all the files that will be installed with their destination paths:

```
drwxr-xr-x root/root         0 2002-03-28 13:44:57 ./
drwxr-xr-x root/root         0 2002-03-28 13:44:49 ./sbin/
-rwxr-xr-x root/root     31656 2002-03-28 13:44:49 ./sbin/parted
drwxr-xr-x root/root         0 2002-03-28 13:44:38 ./usr/
drwxr-xr-x root/root         0 2002-03-28 13:44:41 ./usr/share/
drwxr-xr-x root/root         0 2002-03-28 13:44:38 ./usr/share/man/
drwxr-xr-x root/root         0 2002-03-28 13:44:52 ./usr/share/man/man8/
-rw-r--r-- root/root      1608 2002-03-28 13:44:37 ./usr/share/man/man8/parted.8.gz
drwxr-xr-x root/root         0 2002-03-28 13:44:41 ./usr/share/doc/
drwxr-xr-x root/root         0 2002-03-28 13:44:52 ./usr/share/doc/parted/
-rw-r--r-- root/root      1880 2002-03-07 14:20:08 ./usr/share/doc/parted/README.Debian
-rw-r--r-- root/root      1347 2002-02-27 01:40:50 ./usr/share/doc/parted/copyright
-rw-r--r-- root/root      6444 2002-03-28 13:37:33 ./usr/share/doc/parted/changelog.Debian.gz
-rw-r--r-- root/root     15523 2002-03-28 02:36:43 ./usr/share/doc/parted/changelog.gz
```

It must be the last file in the deb archive.

3.3. control.tar.gz

In our example this file has the following content:

```
-rw-r--r--    1 root     root         1336 Mar 28  2002 control
-rw-r--r--    1 root     root          388 Mar 28  2002 md5sums
-rwxr-xr-x    1 root     root          253 Mar 28  2002 postinst
-rwxr-xr-x    1 root     root          194 Mar 28  2002 prerm
```

'md5sums' contains for each file in data.tar.gz the md5sum. In our example the content looks like this:

```
1d15dcfb6bb23751f76a2b7b844d3c57  sbin/parted
4eb9cc2e192f1b997cf13ff0b921af74  usr/share/man/man8/parted.8.gz
2f356768104a09092e26a6abb012c95e  usr/share/doc/parted/README.Debian
a6259bd193f8f150c171c88df2158e3e  usr/share/doc/parted/copyright
7f8078127a689d647586420184fc3953  usr/share/doc/parted/changelog.Debian.gz
98f217a3bf8a7407d66fd6ac8c5589b7  usr/share/doc/parted/changelog.gz
```

Don't worry, the 'md5sum' file as well as the 'postinst' and 'prerm' files are not mandatory for your first package. But please take a note of their existence, every proper official Debian package has them for good reasons.

'prerm' and 'postinst' seem to take care of removing old documentation files and adding a link from doc to share/doc.

```
$ cat postinst
#!/bin/sh
set -e
# Automatically added by dh_installdocs
if [ "$1" = "configure" ]; then
  if [ -d /usr/doc -a ! -e /usr/doc/parted -a -d /usr/share/doc/parted ]; then
    ln -sf ../share/doc/parted /usr/doc/parted
  fi
fi
# End automatically added section

$ cat prerm
#!/bin/sh
set -e
# Automatically added by dh_installdocs
if [ \( "$1" = "upgrade" -o "$1" = "remove" \) -a -L /usr/doc/parted ]; then
  rm -f /usr/doc/parted
fi
# End automatically added section
```

And finally the most interesting file:

```
$ cat control
Package: parted
Version: 1.4.24-4
Section: admin
Priority: optional
Architecture: i386
Depends: e2fsprogs (>= 1.27-2), libc6 (>= 2.2.4-4), libncurses5 (>= \
5.2.20020112a-1), libparted1.4 (>= 1.4.13+14pre1), libreadline4 (>= \
4.2a-4), libuuid1
Suggests: parted-doc
Conflicts: fsresize
Replaces: fsresize
Installed-Size: 76
Maintainer: Timshel Knoll <timshel@debian.org>
Description: The GNU Parted disk partition resizing program
 GNU Parted is a program that allows you to create, destroy,
 resize, move and copy hard disk partitions. This is useful
 for creating space for new operating systems, reorganizing
 disk usage, and copying data to new hard disks.
 .
 This package contains the Parted binary and manual page.
 .
 Parted currently supports DOS, Mac, Sun, BSD, GPT and PC98
 disklabels/partition tables, as well as a 'loop' (raw disk)
 type which allows use on RAID/LVM. Filesystems supported are
 ext2, ext3, FAT (FAT16 and FAT32) and linux-swap. Parted can
 also detect HFS (Mac OS), JFS, NTFS, ReiserFS, UFS and XFS
 filesystems, but cannot create/remove/resize/check these
 filesystems yet.
 .
 The nature of this software means that any bugs could cause
 massive data loss. While there are no known bugs at the moment,
```

```
they could exist, so please back up all important files before
running it, and do so at your own risk.
```

Further information about the control file can be obtained via '`man 5 deb-control`'.

4. Hands On

Now it is time to get practical ourselves. I have a simple shell script named 'linuxstatus' which I want to install as '/usr/bin/linuxstatus'. So first let's create a directory named 'debian' next to the file 'linuxstatus'.

```
$ mkdir -p ./debian/usr/bin
$ cp linuxstatus ./debian/usr/bin
```

4.1. control

Let's start with the control file. The version number must have a dash with an additional Debian package version number, e.g. '1.1–1'. If your program consists e.g. only of portable shell scripts, use 'all' as its 'Architecture'.

For 'Depends' you might need to find out to which package a certain file or program your new package relies onto belongs to. You can use 'dpkg –S <file>' for this to find this out, e.g.:

```
$ dkpg -S /bin/cat
coreutils: /bin/cat
```

Then to find out more about package 'coreutils' you can use the command 'apt–cache showpkg coreutils', which will tell you among other things the current version number that is installed on the system.

As a side note, there are two more ways to find the same information. There is a web page where you can search for Debian files: http://www.debian.org/distrib/packages. Go to the bottom of that page to fill out the web form.

Last not least there is a nice GUI application named 'kpackage', which provides convenient package browsing options and also allows to search after packages given individual file names.

'Suggests', 'Conflicts', and 'Replaces' etc. can be left out if not needed.

So here is the result of our first 'control' file:

```
Package: linuxstatus
Version: 1.1-1
Section: base
Priority: optional
Architecture: all
Depends: bash (>= 2.05a-11), textutils (>= 2.0-12), awk, procps (>= \
1:2.0.7-8), sed (>= 3.02-8), grep (>= 2.4.2-3), coreutils (>= 5.0-5)
Maintainer: Chr. Clemens Lee <clemens@kclee.de>
Description: Linux system information
 This script provides a broad overview of different
 system aspects.
```

The 'control' file gets copied into a directory called 'DEBIAN' inside the other 'debian' directory.

```
$ mkdir -p debian/DEBIAN
$ find ./debian -type d | xargs chmod 755   # this is necessary on Debian Woody, don't ask me why
$ cp control debian/DEBIAN
```

If you expect your package to have a bigger audience in the future it might help to read this Writing Debian package descriptions article.

4.2. dpkg−deb

Now it is almost done. Just type:

```
$ dpkg−deb −−build debian
dpkg−deb: building package `linuxstatus' in `debian.deb'.
$ mv debian.deb linuxstatus_1.1−1_all.deb
```

Uh, that was all easier than expected. Now we just have to install this package on our box and we are done:

```
root# dpkg −i ./linuxstatus_1.1−1_all.deb
```

Type '`linuxstatus`' or '`ls −l /usr/bin/linuxstatus`' to see if it worked. If you don't like your package any more, just type '`dpkg −r linuxstatus`' and check again that the package is deinstalled. If you install a newer version you don't have to remove the old one first, thought.

If you are curious about the version numbering scheme and naming conventions for a Debian package, have a read at this section in The Debian Reference.

5. Double Check

Now that you have gotten a first impression and build your own binary package, it is time to get a little bit more serious and have a look at the quality of the package that we have produced.

5.1. lintian

Luckily for us the Debian project provides a 'lint' like tool for checking Debian packages. This tool is named 'lintian'. If you have not installed it yet on your system, this is a good moment (`apt-get install lintian`).

Now we use this little treasure tool on our new package file:

```
$ lintian linuxstatus_1.1-1_all.deb
E: linuxstatus: binary-without-manpage linuxstatus
E: linuxstatus: no-copyright-file
W: linuxstatus: prerm-does-not-remove-usr-doc-link
W: linuxstatus: postinst-does-not-set-usr-doc-link
```

Uh, doesn't look so perfect. We miss a man page, copyright file, and also those 'prerm' and 'postinst' scripts.

5.2. Minimal Documentation

This is not the place to say much about writing and creating man pages, there are many books that have one or another chapter related to this topic and there is also The Linux MAN−PAGE−HOWTO online. So lets do a little time warp and assume you have now a perfect man page for your script at location `./man/man1/linuxstatus.1`.

The same for a 'copyright' file. You can find enough examples under the `/usr/share/doc` directory with this command: `find /usr/share/doc -name "copyright"`

So here is our own example of a 'copyright' file:

```
linuxstatus

Copyright: Chr. Clemens Lee <clemens@kclee.de>

2002-12-07

The home page of linuxstatus is at:
http://www.kclee.de/clemens/unix/index.html#linuxstatus

The entire code base may be distributed under the terms of the GNU General
Public License (GPL), which appears immediately below.  Alternatively, all
of the source code as any code derived from that code may instead be
distributed under the GNU Lesser General Public License (LGPL), at the
choice of the distributor. The complete text of the LGPL appears at the
bottom of this file.

See /usr/share/common-licenses/(GPL|LGPL)
```

For the 'prerm' and 'postinst' scripts we copy one to one the examples from the 'parted' package above into files with the same name in our own project directory. These files should work for us just as well.

Now we create the debian package again. In the 'control' file we first increase the version number from 1.1−1 to 1.2−1 (since we have written a new man page we increase our internal release number). We also need to copy the new files to their appropriate places:

```
$ mkdir -p ./debian/usr/share/man/man1
$ mkdir -p ./debian/usr/share/doc/linuxstatus
$ find ./debian -type d | xargs chmod 755
$ cp ./man/man1/linuxstatus.1 ./debian/usr/share/man/man1
$ cp ./copyright ./debian/usr/share/doc/linuxstatus
$ cp ./prerm ./postinst ./debian/DEBIAN
$ gzip --best ./debian/usr/share/man/man1/linuxstatus.1
$
$ dpkg-deb --build debian
dpkg-deb: building package `linuxstatus' in `debian.deb'.
$ mv debian.deb linuxstatus_1.2-1_all.deb
```

Gzip is necessary because lintian expects man page files to be compressed as small as possible.

5.3. fakeroot

Now lets see if our package has become a better Debian citizen:

```
$ lintian linuxstatus_1.2-1_all.deb
E: linuxstatus: control-file-has-bad-owner prerm clemens/clemens != root/root
E: linuxstatus: control-file-has-bad-owner postinst clemens/clemens != root/root
E: linuxstatus: bad-owner-for-doc-file usr/share/doc/linuxstatus/ clemens/clemens != root/root
E: linuxstatus: bad-owner-for-doc-file usr/share/doc/linuxstatus/copyright clemens/clemens != root/r
E: linuxstatus: debian-changelog-file-missing
```

Ups, new complains. OK, we will not give up. Actually most errors seem to be the same problem. Our files are all packaged for user and group 'clemens', while I assume most people would prefer having them installed as 'root/root'. But this is easily fixed using the tool '*fakeroot*'. So lets fix and check this quickly (while ignoring the changelog issue):

```
$ fakeroot dpkg-deb --build debian
dpkg-deb: building package `linuxstatus' in `debian.deb'.
$ mv debian.deb linuxstatus_1.2-1_all.deb
$ lintian linuxstatus_1.2-1_all.deb
E: linuxstatus: debian-changelog-file-missing
```

Fine, but we have yet another file to add to the package.

5.4. More Documentation

Let me tell you already that next to a 'changelog' file in the 'doc/linuxstatus' directory a 'changelog.Debian' file is also required. Both should be gzipped as well.

Here are two example files, 'changelog':

```
linuxstatus (1.2-1)

  * Made Debian package lintian clean.

 -- Chr. Clemens Lee <clemens@kclee.de>  2002-12-13
```

and 'changelog.Debian':

```
linuxstatus Debian maintainer and upstream author are identical.
Therefore see also normal changelog file for Debian changes.
```

The Debian Policy file has more details regarding the format of the changelog file.

Now hopefully our last step will be:

```
$ cp ./changelog ./changelog.Debian ./debian/usr/share/doc/linuxstatus
$ gzip --best ./debian/usr/share/doc/linuxstatus/changelog
$ gzip --best ./debian/usr/share/doc/linuxstatus/changelog.Debian
$ fakeroot dpkg-deb --build ./debian
dpkg-deb: building package `linuxstatus' in `debian.deb'.
$ mv debian.deb linuxstatus_1.2-1_all.deb
$ lintian linuxstatus_1.2-1_all.deb
```

Ah, we get no more complains :-). As root you can install now this package over the old one, again with the standard '`dpkg -i`' command.

```
root# dpkg -i ./linuxstatus_1.2-1_all.deb
(Reading database ... 97124 files and directories currently installed.)
Preparing to replace linuxstatus 1.1-1 (using linuxstatus_1.2-1_all.deb) ...
Unpacking replacement linuxstatus ...
Setting up linuxstatus (1.2-1) ...
```

6. Summary

Not to get confused, let us recapture all steps we have taken to build our binary Debian package.

Prerequisite files:

1. one or more binary executable or shell script files
2. a man page for each executable file
3. a 'control' file
4. a 'copyright' file
5. a 'changelog' and 'changelog.Debian' file

Setup temporary 'debian' directories:

1. create '`debian/usr/bin`' directory (or wherever you plan to place your executable files)
2. create '`debian/usr/share/man/man1`' (or whatever section your man page belongs into)
3. create '`debian/DEBIAN`' directory
4. create '`debian/usr/share/doc/<package_name>`'
5. make sure all sub directories of '`debian`' have file permission 0755

Copy files into temporary 'debian' tree:

1. copy executable file into '`debian/usr/bin`' directory (or wherever you plan to place your executable files)
2. copy man page file into '`debian/usr/share/man/man1`' directory
3. copy 'control' file into '`debian/DEBIAN`' directory
4. copy 'copyright', 'changelog', and 'changelog.Debian' files into '`debian/usr/share/doc/<package_name>`'
5. gzip man page, 'copyright', 'changelog', and 'changelog.Debian' files with option '−−best' inside the temporary '`debian`' tree

Build and check binary Debian package:

1. invoke '`dpkg-deb --build`' using '`fakeroot`' on the '`debian`' directory
2. rename resulting '`debian.deb`' file to its final package name including version and architecture information
3. check resulting .deb package file for Debian policy compliance using '`lintian`'

---

7. What Else

There are many details which have not been covered here, like how to distribute Unix daemons, configuration files and much more.

But most important, I want to emphasize again that for Debian maintainers, packages are source packages, not binary packages. They never interact directly with the internals binary packages. In fact only '`dpkg-deb`' and '`dpkg`' developers need to know what they are. In fact it is not recommended to do so.

If a developer were to explain someone how to build a Debian package, he will certainly explain how to make a source package and how to build it.

On the other hand, not every developer wants to submit his software to Debian (yet), but still wants to profit from the advantages a packaging system like 'dpkg' offers without releasing package source code. Personally I will release my freeware projects still as tar.gz files with source code etc. for all kind of platforms, while I plan to offer more and more '.deb' packages for the convenience of Debian users who just want to install and use my software.

If someone wants to do the next step to submit a software package to Debian, you have to move on to study the Debian New Maintainers' Guide as well as the Debian Policy Manual first. On your undertaking to create a Debian source package, also have a look at the debian−mentors mailing list to see experienced and beginning Debian developers interacting with each other and tackling similar problem you might encounter.

---

8. Credits

Thanks to

- Colin Watson for contributing a sentence in the abstract and giving feedback regarding structure, focus and title of this HowTo,
- Bill Allombert for contributing a sentence in the 'What Else' section and giving feedback reinforcing what Colin had said as well,
- Santiago Vila for pointing out that 'md' is a local alias and the (Unix) world has standardized on 'mkdir',
- Tabatha Marshall from the TLDP project for general support,
- Joey Hess, Carlo Perassi, and Joe Riel for minor corrections,
- Claudio Cattazzo for conversion of my private xml document to standard docbook format together which resulted also in a number of minor corrections and improvements [2003−10−04],
- Frank Lichtenheld for finding errors introduced when the HTML pages get generated and for suggesting to use 'dpkg −S <file>' when looking for a package containing a certain file [2003−12−13],
- Hugh Hartmann for translating this HOWTO to Italian and Claudio Cattazzo for revising the translation [2003−12−13], and
- Oguz Yarimtepe for translating this HOWTO to Turkish [2005−08−09].

9. Links

Debian New Maintainers' Guide:	http://www.debian.org/doc/maint−guide/
Debian:	http://www.debian.org/
The Debian Reference:	http://www.debian.org/doc/manuals/reference/reference.en.html
Debian Packages:	http://www.debian.org/distrib/packages
Writing Debian package descriptions:	http://people.debian.org/~walters/descriptions.html
The Linux MAN−PAGE−HOWTO:	http://www.tldp.org/HOWTO/mini/Man−Page.html
Debian Policy Manual:	http://www.debian.org/doc/debian−policy/
Debian Mentors:	http://lists.debian.org/debian−mentors/
The Linux Documentation Project:	http://www.tldp.org/
Plug: my own little Unix page:	http://www.kclee.de/clemens/unix/