

**Umeå University**  
Department of Applied Physics and Electronics

**Linux as Development Environment 7.5 ECTS**  
**5EL142 HT-16**

**Assignment 13 - Debian Package Handling**

Submitted 2017-08-19  
Author: Lorenz Gerber, 20161202-2033 (lorenzottogerber@gmail.com)  
Instructor: Sven Rönnbäck, John Berge, Björne A Lindberg

## 1 Introduction

This lab was about creating Debian packages from the application written for lab 6 and lab 11 respectively.

Litterature study on various web resources showed that there are basically two different approaches: Starting with a tar.gz source archieve of the so called ‘upstream application’ to be packaged for debian, or create directly a binary package. The former will ultimately also result in a binary package. It’s the more generic way where the source is first compiled for the respective platform. As the lab instructions and forum entries suggested that both is viable, it was decided to produce directly binary packages.

## 2 Method

### 2.1 General Description

First a suitable binary has to be obtained or created. For normal programs, the requirements are not very specific. Here it was also aimed to build a shared library package. For this, the binary has to be rebuild according to more strict specifications (e.g. using -soname flag)[2].

Then, the directory tree has to be set up and the binary file(s) are copied into it. Further, the required configuration files have to be created and copied into the directory tree.

Then the debian package is ready to be packaged/created.

Finally, the result should be checked using the debian package linter tool ‘lintian’. Usually, the file modes have to be adjusted. It was found most suitable to iterate the last two steps, ‘building’ and ‘linting’ until a satisfactory result is obtained.

### 2.2 Tools

For producing a binary deb package, almost no tools besides a text editor were needed:

1. gcc build system to first produce the binaries
2. dpkg-deb –build: to create the actual package
3. fakeroot: used with dpkg-deb, to create the package as root user
4. lintian: linter tool to check quality of the deb package

For building source packages, a number of other tools that help to set up the initial directory tree and template files would be available.

## 3 Detailed Description

### 3.1 libelectro1

As it was decided to package the libraries as a real shared library package, the proposed name ‘electrolib’ was changed to ‘libelectro’. First the the libraries where rebuild using the -soname flag [3]:

```
gcc -fPIC -c -Wall libresistance.c
gcc -fPIC -c -Wall libpower.c
gcc -fPIC -c -Wall libcomponent.c
```

2(6)

```
gcc -shared -Wl,-soname,libelectro.so.1 \  
-o libelectro.so.1.0.1 libresistance.o libpower.o libcomponent.o -lc
```

This resulted in a shared library: libelectro.so-1.0.1. A symbolic link libelectro.so.1 was also created:

```
ln -s libelectro.so-1.0.1 libelectro.so.1
```

Then a new directory tree was build:

```
mkdir -p ./debian/usr/lib  
mkdir -p ./debian/DEBIAN
```

The library and the symbolic link where copied into ./debian/usr/lib. Then in ./debian/DEBIAN three new files where created: control, triggers and shlibs.

The control file was edited as follows [1]:

```
Package: libelectrol  
Version: 1.0-1  
Section: libs  
Priority: optional  
Architecture: amd64  
Depends: libc6 (>= 2.2.1)  
Maintainer: L. Gerber <lorenz.gerber@provement.se>  
Description: library with functions to calculate  
e12 replacement resistance values.
```

The triggers file contains only one line to trigger ldconfig to update the system with the newly installed shared library [2]:

```
activate-noawait ldconfig
```

The shlibs file contains the following line [2, 8.6.4.2.]:

```
libelectro 1 libelectrol
```

Then the binary package was created:

```
fakeroot dpkg-deb --build libelectro-1.0/  
mv libelectro-1.0.deb libelectro.so-1.0-1_amd64.deb
```

Finally the quality of the produced package was checked using:

```
lintian libelectro.so-1.0-1_amd64.deb
```

This resulted in the follwing output:

```
E: libelectrol: unstripped-binary-or-object usr/lib/libelectro.so.1.0.1  
E: libelectrol: debian-changelog-file-missing  
E: libelectrol: no-copyright-file
```

The first error relates to the debugging symbols left in the binary as the library was compiled with the 'g' flag. In some documentation, it is advised to leave debugging symbols. For real Debian packages and productive code, there would be the possibility for two versions of the same library, a stripped one and a development library with the debugging symbols left. Here, for testing, the symbols where stripped using:

```
objcopy --strip-debug --strip-unneeded libelectro.so.1.0.1
```

The changelog file had to be gzipped with the parameters `-n -9`. Lintian is very picky about the format of the changelog file. The ‘ITM close bug’ issue still shown in ‘lintian’ can not be mended as it is a mechanism used when a package is developed to be included in the debian distribution: Lintian recognizes that the changelog only contains one entry, hence it assumes a new package. New packages have to be announced by filing a bug-report in the ITM system. Then the bug number has to be mentioned in the first entry to automatically close it. The following is the example text for the changelog file to be situated in `usr/share/doc/electrolib1/`:

```
libelectrol (1.0-1) UNRELEASED; urgency=low
```

```
* Initial release.
```

```
Not intended to be included in Debian, hence closes no initial bug.
```

```
-- Lorenz Gerber <lorenz.gerber@provement.se> Fri, 18 Aug 2017 8:08:00 +0000
```

**The following copyright file was added in libelectro-1.0/usr/share/doc/libelectrol/:**

```
Format: https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
```

```
Upstream-Name: electrotest
```

```
Source: https://github.com/lorenzgerber/electrotest
```

```
Files: *
```

```
Copyright: 2017 Lorenz Gerber <lorenz.gerber@provement.se>
```

```
License: GPL-2+
```

```
This program is free software; you can redistribute it
and/or modify it under the terms of the GNU General Public
License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later
version.
```

```
.
```

```
This program is distributed in the hope that it will be
useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the GNU General Public License for more
details.
```

```
.
```

```
You should have received a copy of the GNU General Public
License along with this package; if not, write to the Free
Software Foundation, Inc., 51 Franklin St, Fifth Floor,
Boston, MA 02110-1301 USA
```

```
.
```

```
On Debian systems, the full text of the GNU General Public
License version 2 can be found in the file
`/usr/share/common-licenses/GPL-2'.
```

Afterwards ‘lintian’ indicated that the package was clean. It was then installed using:

```
sudo dpkg -i libelectro.so-1.0-1_amd64.deb
```

and uninstalled using:

```
sudo dpkg -r libelectro1
```

### 3.2 electrotest

First, it was tested whether electrotest could be build linking against the shared library provided in the debian package libelectro1. Unfortunately, this was not the case. It was established that there was another symbolic link in the library package needed: from 'lib-electro.so' to 'libelectro.so.1.0.1'. Then electrotest compiled and could be run:

```
gcc -Wall -std=c11 -pendantic -c electrotest.c
gcc -Wall -o electrotest electrotest.o -lelectro -lm -L/usr/lib
```

The same steps as for the library were needed to build a Debian package for electrotest. First, the directory tree was created. Here the binary was copied into 'electrotest-1.0/usr/bin'. 'electrotest-1.0/DEBIAN' only contains the 'control' file which was modified accordingly:

```
Package: electrotest1
Version: 1.0-1
Section: electronics
Priority: optional
Architecture: amd64
Depends: libc6 (>= 2.2.1), libelectro1 (>= 1.0-1)
Maintainer: L. Gerber <lorenz.gerber@provement.se>
Description: shell application to calculate
    e12 replacement resistance values.
```

Further, 'electrotest-1.0/usr/share/doc/electrotest1/' contains the files 'copyright' and 'changelog.Debian.gz' which were also adapted. Further, 'lintian' hinted the need for man pages. They were created in 'electrotest-1.0/usr/share/man/man1/':

```
.\" Manpage for electrotest1.
.\" Contact lorenzottogerberls@gmail.com to correct errors or typos.
.TH ELECTROTEST 1 "18 August 2017"
.SH NAME
electrotest \- shell program to calculate e12 replacement resistances
.SH SYNOPSIS
.B electrotest
.SH DESCRIPTION
Interactively queries the parameters to determines e12 replacement resistances.
.SH RETURN VALUE
exit status
.SH CONFORMING TO
C11
.SH BUGS
currently no bugs known
.SH SEE ALSO
.BR libelectro(1)
.SH AUTHOR
Lorenz Gerber (lorenzottogerber@gmail.com)
```

### 3.3 electrotestgtk

To package ‘electrotestgtk’ basically the same procedure as for ‘electrotest’ was followed except that there are a whole range of dependencies to include for GTK+-2.0. First it was attempted to use `ldd electrotestgtk` but this resulted in a list of about 50 libraries and felt unreasonable to write into the package control file. It was assumed that most of the libraries should get included as downstream libraries of a few top GTK+-2.0 related libraries. Hence `pkg-config gtk-2.0 --libs` gave a shortlist of libraries: `libgtk-x11-2.0`, `libgdk-x11-2.0`, `libpangocairo-1.0`, `libatk-1.0`, `libcairo`, `libgdk_pixbuf-2.0`, `libgio-2.0`, `libpangoft2-1.0`, `libpango-1.0`, `libgobject-2.0`, `libglib-2.0`, `libfontconfig` and `libfreetype`.

After including these in the control file, and updating the respective files (copyright, changelog.Debian.gz, man-pages, binary), the package could be built in the same way as ‘electrotest’. However, it turned out that these were not the libraries to be included: On package install using ‘`sudo dpkg -i`’, the error message of missing libraries was returned. It got clear, that it was rather the debian package name that was required instead of all the single libraries. Hence, it was enough to require ‘`libgtk2.0-0`’ in the control file which is probably available anyway on most desktop (debian/ubuntu) linux systems. Finally the all packages could be installed and run. It was also tested that installation of either ‘electrotest’ or ‘electrotestgtk’ was not possible when ‘electrolib’ was not installed. Further, it was tested that ‘electrolib’ could not be uninstalled as long as either ‘electrotest’ or ‘electrotestgtk’ were still installed.

## 4 Conclusion

### 4.1 Lab 13

This was a lab that I first expected to be rather quick and easy. It turned out that things were a bit more complicated than expected. But it was definitively a very useful lab as one could rehearse and deepen the knowledge about shared. The most annoying experience during the exercise was to get the changelog format right. I find it strange that the requirements are so strict instead of building the tools more advanced that finally have to handle the files. But of course, changelog would be less of a hassle when one builds first the source package and makes use of the automated tools that yield an initial changelog file and further allow automated entries into the changelog file.

### 4.2 Course

All in all, it took me almost a year to finish this course. That was mostly because I’m also 100% CS student at umu and work 50% to make a living. I used Linux systems since a long time. But during this course, I felt more and more ashamed that I never before tried to dig deeper into it. I think the course is really good and focuses mostly on the right spots. Because I used Linux already a long time before, I was reluctant to read the book in the beginning (I have access to it over Safari Online from work). The more advanced the course got, the more I realized how good actually the book is. Finally I ended up reading most of the related chapters. I’m very happy that you allowed to study even at a slower speed. I’m aware that this can be annoying for the teacher when you get submissions from students that started long time ago. Hence I would like to thank the teacher/instructors for doing a great job. I will certainly soon sign up for another distant learning course at Applied Physics and Electronics dep.

## References

- [1] Debian Policy Manual, control files and their fields. <https://www.debian.org/doc/debian-policy/ch-controlfields.html>, 2017. accessed: 2017-08-17.
- [2] Debian Policy Manual, shared libraries. <https://www.debian.org/doc/debian-policy/ch-sharedlibs.html>, 2017. accessed: 2017-08-17.
- [3] The Linux Documentation Project, shared libraries. <https://tldp.org/HOWTO/Program-Library-HOWTO/shared-libraries.html>, 2017. accessed: 2017-08-17.