# Debian Packaging

von Frank Habermann <lordlamer@lordlamer.de>
Korrekturen von Robert Walter <leprovokateur@gmx.de>
Version: 1.1

## Index



## 1. Einleitung

In dieser Dokumentation möchte ich zeigen, dass es nicht schwer ist, ein Debian-Paket zu erstellen. Die Dokumentation richtet sich vor allem an Anfänger, die ihre ersten Schritte machen wollen, um ein Debian-Paket zu erstellen. Ich werde das Erstellen des Paketes anhand von phpsysinfo zeigen.

## 2. Dank

Ohne Hilfe hätte ich diese Dokumentation nicht schreiben können und hätte auch noch keine Debian-Pakete erstellen können. Also ein herzliches Dankeschön an alle, die mir geholfen haben. Allen voran Daniel Baumann <daniel@debian.org>, den ich mit meinen Fragen gequält habe ;) Danke, dass du trotzdem immer eine Antwort parat hattest.

## 3. Debian-Versionen und die Philosophie dahinter

Bevor wir mit dem Pakete-bauen anfangen können, müssen wir erst einmal ein Grundverständnis für die verschiedenen Distributionszweige in Debian mitbringen. Dieses Grundverständnis benötigt ihr, damit ihr wisst was/wie/wo euer Paket macht.

Es gibt Debian grundsätzlich in 3 Versionen: stable, testing und unstable. stable enthält immer die gleichen Programmversionen einer Software. In stable werden nur Security-Pakete eingespielt. Es werden also keine neuen Funktionen zu Programmen hinzugefügt oder entfernt. Noch als kleiner

Hinweis, wenn in einer neueren Programmversion ein sicherheitsrelevanter Bug gefunden wird und er auch in stable auftritt, muss dieser Bug auch in stable separat gefixt werden, aber ohne das Funktionen hinzugefügt oder entfernt werden.

Kommen wir nun zu unstable. unstable enthält immer die neuesten Pakete der Paketbetreuer. Das heißt, unstable ist sehr aktuell was die Programmversionen angeht, kann aber, wie es der Name schon sagt, manchmal sehr instabil sein, da die Pakete nicht immer getestet sind. Aber was ihr euch merken müsst ist, dass, wenn ihr Pakete baut und diese in Debian einbringt, landen diese immer zuerst in unstable!

In unstable bleiben die Pakete in der Regel für 10 Tage, sofern keine kritischen Fehler bemerkt wurden oder in der Zwischenzeit nicht bereits eine neuere Version eingespielt wurde. Nach diesen 10 Tagen rutschen sie dann in testing. In testing liegen also die etwas “ausgereiften” Pakete, welche trotzdem noch recht aktuell sind.

So, nun kennen wir die 3 Softwarezweige in Debian, aber was passiert, wenn Debian jetzt eine neue Version veröffentlichen will? Das geschieht folgendermaßen: Es wird ein Paketfreeze in testing geben. Dann werden in testing nur noch die releasekritischen Bugs entfernt. Wenn alle entfernt sind, wird testing der neue Stable Zweig.

Jeder Softwarezweig in Debian hat auch einen Namen. Ihr habt sicher schon von Potato,Woody,Sarge und Etch gehört. Auch SID ist euch bestimmt bekannt. Jedes Debian Release erhält einen Namen. unstable nimmt eine kleine Sonderrolle ein. unstable heißt auch SID und wird auch immer SID heißen da unstable ja nie released wird.

Wenn es nun ein neues stabiles Release gibt, wird das alte Release noch mindestens ein Jahr nach Erscheinen der nächsten stable-Version mit Securityfixes versorgt und irgendwann danach dann eingestellt und ins Archiv verschoben.

Euch als Paketbetreuer muss klar sein, dass, wenn ihr Pakete in Debian habt, ihr euch um die Sicherheit eurer Software kümmern müsst! Das heißt, auch wenn es ältere Programmversionen von euch in stable gibt, müsst ihr diese mit Securityfixes sicher halten.

So, ich denke, das sollte für das Grundverständnis reichen. Jetzt können wir loslegen, uns unsere Arbeitsumgebung anzulegen.

## 4. Erstellen einer Arbeitsumgebung

Wie wir erfahren haben, kommen neue Pakete immer als erstes in unstable. Das heißt nun für euch, ihr braucht eine unstable-Umgebung, in der ihr eure Pakete baut. Dies geht auf zweierlei Wegen. Entweder ihr benutzt unstable direkt als System oder ihr erstellt euch eine chroot-Umgebung für unstable. Ich

tendiere eher zum Zweiten und werde euch zeigen, wie ihr diese einrichtet.

### 4.1 Erstellen einer chroot-Umgebung

Eine kurze Erklärung vorweg für alle Leute, die nicht wissen, was eine chroot-Umgebung ist. Eine chroot-Umgebung ist quasi ein Gefängnis oder ein Raum, in dem ich Prozesse laufen lassen kann, ohne dass diese Prozesse aus diesem Gefängnis ausbrechen können. Kleines Beispiel: Ihr lasst euren Apache in einer chroot-Umgebung laufen. Wenn nun dieser Apache zum Beispiel "gehackt" wird, kann der "Hacker" nur Dateien in dieser chroot-Umgebung ändern oder ausspähen. (Hinweis: Dies ist nur theoretisch – Es ist auch praktisch möglich noch aus dieser Umgebung auszubrechen. Soll an dieser Stelle aber nicht weiter erläutert werden.)

So, nun können wir anfangen, unsere Umgebung zu erstellen. Dafür benötigen wir jetzt debootstrap. debootstrap ist ein Programm, dass eine Minimalinstallation eines Debianzweiges in ein Verzeichnis machen kann. Wir haben also vor, mit debootstrap uns eine Minimalinstallation von unstable in ein Verzeichnis unserer Wahl zu machen.

Zuerst einmal brauchen wir debootstrap oder cdebootstrap, welches die C Umsetzung von debootstrap ist. Wenn ihr Debian stable verwendet, müsst ihr euch debootstrap bzw. cdebootstrap von Backports (www.backports.org) holen; die Version in stable ist zu alt, um aktuelle Systeme zu erstellen.

Das heißt, ihr installiert nun debootstrap mit "dpkg -i paketname", wenn ihr die Datei selbst heruntergeladen habt oder ihr fügt Backports als Repository zu eurer Sourcelist hinzu und installiert dann debootstrap mit "apt-get install debootstrap". Wie ihr das macht, müsst ihr selber entscheiden, Hauptsache, ihr installiert debootstrap ;)

Wenn ihr debootstrap jetzt installiert habt, könnt ihr auch nur als root debootstrap verwenden. Nun müsst ihr nur noch wissen, in welchem Verzeichnis ihr die Umgebung erstellen wollt. Entweder direkt bei einem Benutzer oder in einem separatem Bereich, Eure Wahl. Wechselt in dieses Verzeichnis und erstellt dann mit folgendem Befehl die unstable-Umgebung:

```
debootstrap --variant=buildd sid unstable-chroot ftp://ftp.de.debian.org/debian/
```

Die Installation wird je nach Internetanbindung eine Weile dauern. Es werden jetzt die benötigten Pakete heruntergeladen und in dieses Verzeichnis installiert. Für nähere Informationen zu debootstrap empfiehlt sich die Manpage.

Nachdem debootstrap fertig ist, habt ihr in diesem Fall ein Verzeichnis "unstable-chroot" oder halt so wie ihr den Namen des Verzeichnis an debootstrap übergeben habt. Jetzt können wir die Umgebung mit chroot betreten. Dies geht ganz einfach mit einem "`chroot unstable-chroot`" oder "`chroot`

`/pfad/zu/unstable-chroot`".

Jetzt müssen wir noch ein paar Sachen konfigurieren. Als erstes müsst ihr noch in /etc/apt/source.list die Repositories von unstable hinzufügen.

```
deb http://ftp.de.debian.org/debian/ unstable main non-free contrib
deb-src http://ftp.de.debian.org/debian/ unstable main
```

Als nächstes müsst ihr die /etc/resolv.conf anlegen und mit

```
nameserver 192.168.1.1
```

einen Nameserver angeben zur Namensauflösung. Danach ein

```
apt-get update && apt-get upgrade
```

Nun noch locales und vim installieren mit

```
apt-get install locales vim
```

und dann „`dpkg-reconfigure locales`“ machen. Jetzt fehlen noch ein paar benötigte Programme

```
apt-get install build-essential fakeroot debhelper gnugpg dh-make lintian
devscripts wdiff
```

Jetzt haben wir es fast geschafft. Jetzt müssen wir uns einen Nutzer anlegen mit

```
adduser fhabermann
```

Nach dem Anlegen des Benutzers wechseln wir zu diesem mit

```
su - fhabermann
```

Mit dem Benutzer müssen wir jetzt nur noch einmal unsere gnupg-Daten anlegen oder vorhandene hierher kopieren.

## 5. Debian Paket erstellen

Jetzt können wir anfangen, uns um unser Paket zu kümmern. In diesem Fall ist unser Paket ganz einfach phpsysinfo. Dieses lade ich mir mit wget von Sourceforge in meine chroot-Umgebung herunter.

Danach entpacke ich es, um dann eine Paket-Orig zu erstellen, die Debian braucht. Die Orig-Datei ist

die eigentliche Quelldatei des Paketes. Diese wird dann unsere Ausgangsbasis sein. Dafür entpacke ich das tar.gz Archiv mit tar -xzf phpsysinfo-x.y.z.tar.gz. Jetzt hab ich ein Verzeichnis phpsysinfo, das ich in phpsysinfo-x.y.z umbenenne und benenne phpsysinfo-x.y.z.tar.gz in phpsysinfo_x.y.z.orig.tar.gz um.

So, weiter geht es. Jetzt wechseln wir wieder in das Verzeichnis phpsysinfo-x.y.z. Jetzt müssen wir das "debian"-Verzeichnis erstellen, in dem alle relevanten Daten liegen, um das Paket zu erstellen und zu installieren.

Dies machen wir ganz einfach mit

```
dh_make –e user@mydomain.org –c gpl –f ../phpsysinfo_x.y.z.orig.tar.gz
```

Als Parameter geben wir gleich unsere Email und die Lizenz mit an. Für weitere Parameter einfach „`–help`" benutzen oder die Manpage lesen.

Jetzt haben wir ein Verzeichnis "debian", in dem sich einige Dateien befinden. Es gibt einige relevante Dateien und einige Beispieldateien. Diese Beispieldateien sind mit .ex oder .EX gekennzeichnet. Da wir diese Beispieldateien nicht benötigen, können wir diese ganz problemlos löschen.

Kommen wir zu den relevanten Dateien, welche da wären: changelog, compat, control, copyright, README.Debian, rules. Alle diese Dateien müssen wir an unsere Bedürfnisse anpassen.

Fangen wir an mit changelog. In dieser Datei werden alle Änderungen an unserem Paket festgehalten. Hier müssen wir am Anfang nur die Zeile mit dem Initial Release kürzen, damit dort nur Initial Release steht. Und was wir nicht vergessen sollten, ist den Namen und die Zeitzone bzw. auch die Zeit anzupassen.

Kommen wir zur Datei compat. In dieser Datei wird die debhelper Major Version eingetragen. Wir brauchen hier nur eine 5 eintragen.

Nun kommt die control. Hier tragen wir bei Section "web" ein und bei Priority "optional". Dann passen wir auch fein unseren Namen an und setzen hier auch wieder die debhelper Version auf 5. Bei Architecture setzen wir "all" ein. Das machen wir, weil unser Paket keine plattformabhängigen Binärdaten enthält. Nun müssen wir bei Depends unsere Paketabhängigkeiten angeben. Also, welche Pakete benötigt werden, wenn unser Paket installiert ist. Also, für unser Webpaket brauchen wir einen Webserver und PHP. Also schaut unsere Zeile wie folgt aus:

Depends: apache2 | apache | httpd, php5 | php4

Die Paketabhängigkeiten werden durch Komma getrennt aufgelistet. Die Pipe dient hier als

“Entweder/Oder”. Das heißt, wir brauchen für unser Paket zum Beispiel PHP5 oder PHP4.

Nun müssen wir in der control eine Description angeben. Diese darf nur maximal 60 Zeichen lang sein und sollte aussagekräftig sein. Unterhalb von Description geben wir eine ausführliche Beschreibung für unser Paket an. Auch hier darauf achten, dass am Anfang ein Leerzeichen steht, damit der Text eingerückt ist, und dass wir nicht mehr als 70 Zeichen in einer Zeile stehen. Unsere control müsste dann so aussehen:

```
Source: phpsysinfo
Section: web
Priority: optional
Maintainer: Frank Habermann <lordlamer@lordlamer.de>
Build-Depends: debhelper (>= 5)
Standards-Version: 3.7.2

Package: phpsysinfo
Architecture: all
Depends: apache2 | apache | httpd, php5 | php4
Description: display information about host being accessed
 Phpsysinfo will display things like Uptime, CPU, Memory, SCSI, IDE,
 PCI, Ethernet, Floppy and Video Information.
```

Damit sind wir auch schon fertig mit der control und kommen zu der copyright. Meine persönlich “liebste” Datei. In dieser Datei geben wir alle Lizenzen an, die in unseren Paket enthalten sind. Wir geben die Autoren und die Lizenzinhaber an und geben auch an, auf welcher Datei welche Lizenz liegt. dh_make, welches wir ja ausgeführt haben, hat in unsere copyright schon die GPL eingetragen. Jetzt müssen wir unsere Datei nur noch anpassen und müssen uns die Lizenzinformationen zu phpsysinfo angucken. Unsere copyright müsste dann etwa wie folgt aussehen:

```
This package was debianized by Hereward Cooper (Hereward Matthew Lawrence
Cooper) <zadok@phreaker.net> on Tue,  1 Jan 2002 22:46:32 +0000.
It is maintained by Frederik Schüler <fs@debian.org> since September 2003.

It was downloaded from http://phpsysinfo.sourceforge.net/

Authors:
                Uriah Welcome <precision@devrandom.net>
                Matthew Snelham <infinite@valinux.com>
                Jacob Moorman <moorman@users.sourceforge.net>
                Webbie <webbie@users.sourceforge.net>

Copyright: 1999-2006 Uriah Welcome, Matthew Snelham, Jacob Moorman, Webbie

   This package 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; version 2 dated June, 1991.

   This package 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 complete text of the GNU General
Public License can be found in `/usr/share/common-licenses/GPL'.


includes/class.Template.inc.php is Copyright 1999-2000 NetUSE GmbH
Kristian Koehntopp, licensed under the GNU Lesser General Public License
which, on Debian systems, can be found in `/usr/share/common-licenses/LGPL'.


includes/XPath.class.php is copyright by

                Nigel Swinson <nigelswinson@users.sourceforge.net>
                Sam Blum <bs_php@infeer.com>
                Daniel Allen <bigredlinux@yahoo.com>
                Michael P. Mehl <mpm@phpxml.org>

and is tri-licensed under the terms of the MPL, GPL and LGPL.
```

Nachdem wir die copyright angepasst haben, nehmen wir uns die README.Debian vor. In diese Datei schreiben wir letzte relevante Informationen für die Benutzer für nach der Installation. In unserem Fall halten wir hier einfach fest, dass man die Konfiguration an seine Bedürfnisse anpassen soll und das diese Datei in /etc/phpsysinfo/ liegt. Wie die Konfiguration dahin, kommt sag ich euch gleich.

Bevor wir uns nun um die rules kümmern, möchte ich noch eine Datei erstellen. Und zwar die apache.conf, die wir für die spezielle Konfiguration des Apache benutzen, um unsere Websoftware anzuzeigen. In diese Datei tragen wir folgendes ein:

```
Alias /phpsysinfo /usr/share/phpsysinfo/

<DirectoryMatch /usr/share/phpsysinfo/>
Options +FollowSymLinks
AllowOverride None
order deny,allow
deny from all
allow from 127.0.0.0/255.0.0.0
</DirectoryMatch>
```

Damit sagen wir dem Apache, dass, wenn an der URL ein /phpsysinfo hängt, unsere Applikation in /usr/share/phpsysinfo aufgerufen werden soll. Für weitere Details zur Apache-Konfiguration solltet ihr die Manpage lesen.

Kommen wir nun zu der Datei rules. In dieser Datei wird festgehalten, was bei der Installation gemacht werden soll. Also, wo unsere Daten abgelegt werden sollen und Weiteres. Es ist nichts weiter als ein simples Makefile, das Befehle ausführt. Beim Kopieren bzw. Installieren gibt es einiges zu beachten. Hier mal ein Beispiel für einen Kopieraufruf:

```
mkdir -p debian/phpsysinfo/usr/share/phpsysinfo
cp -r templates debian/phpsysinfo/usr/share/phpsysinfo/
```

Mit diesen beiden Befehlen werden im System einmal /usr/share/phpsysinfo angelegt und dort der Ordner templates hineinkopiert. Ja, aber wieso schreibt man dann debian/phpsysinfo/...? Weil wir unsere Daten nach debian/phpsysinfo/ installieren müssen. Das dient als eigentliches Root-Verzeichnis. Wir passen dann unsere rules so an, dass sie wie folgt aussieht:

```
#!/usr/bin/make -f

# Uncomment this to turn on verbose mode.
#export DH_VERBOSE=1

build:

clean:
        dh_testdir
        dh_testroot

        dh_clean

install: build
        dh_testdir
        dh_testroot
        dh_clean -k
        dh_installdirs

        mkdir -p debian/phpsysinfo/usr/share/phpsysinfo
        cp -r distros.ini images includes index.php phpsysinfo.dtd templates
debian/phpsysinfo/usr/share/phpsysinfo/
        find debian/phpsysinfo/usr/share/phpsysinfo -type f | xargs chmod 0644
        install -D -m 0644 config.php.new
debian/phpsysinfo/etc/phpsysinfo/config.php

binary-arch: build install

# Build architecture-dependent files here.
binary-indep: build install
        dh_testdir
        dh_testroot
        dh_installchangelogs ChangeLog
        dh_installdocs
        dh_install
        dh_link
```

```
        dh_compress
        dh_fixperms
        dh_installdeb
        dh_gencontrol
        dh_md5sums
        dh_builddeb

binary: binary-indep binary-arch
.PHONY: build clean binary-indep binary-arch binary install
```

Nun haben wir alle relevanten Dateien angepasst und können das Paket endlich erstellen. Dafür wechseln wir nochmal mit "`cd ..`" in das eigentliche phpsysinfo-Verzeichnis und führen dann folgendes aus:

```
dpkg-buildpackage -rfakeroot
```

Hier rattert unser Rechner jetzt durch und fragt uns auch 2mal nach unseren gnupg-Passwort, welches wir angeben müssen. Danach sollten wir in einem Verzeichnis darüber einige neue Dateien erstellt bekommen haben. Das sind

```
phpsysinfo_2.5.2-1_all.deb
phpsysinfo_2.5.2-1.diff.gz
phpsysinfo_2.5.2-1.dsc
phpsysinfo_2.5.2-1_i386.changes
```

Bisher sieht alles gut aus. Jetzt sollten wir aber auch nochmal prüfen, ob wirklich alles passt. Das machen wir mit lintian:

```
lintian -i phpsysinfo_2.5.2-1_i386.changes
```

Wenn wir keine Ausgabe bekommen, ist alles gut. Wenn wir Ausgaben bekommen, kriegen wir auch gleich eine Fehlermeldung, was wir noch berichtigen müssen. Dies können zum Beispiel falsche Syntax in der changelog oder in der copyright sein.

Wenn nun aber alles gut ist, haben wir es geschafft. Wir haben unser Debian-Paket mit Erfolg erstellt!

## 6. Paket in Debian integrieren

Damit unser Paket in die öffentliche Distribution kommt, müssen wir uns einen richtigen Debian-Entwickler suchen. Denn nur die richtigen Debian-Entwickler können Pakete nach Debian hochladen. Die Entwickler schauen vor allem bei eurem ersten Paket etwas genauer hin. Wenn alles OK ist, wird es von ihnen signiert und dann auf die FTP-Server von Debian gespielt. Von nun an läuft alles relativ automatisch. Aber dazu gleich mehr.

Ihr müsst also die relevanten Dateien

```
phpsysinfo_2.5.2-1.diff.gz
phpsysinfo_2.5.2-1.dsc
phpsysinfo_2.5.2.orig.tar.gz
```

an euren Debian-Entwickler übergeben. Nicht aber euer erstelltes .deb Paket selber. Das .deb Paket wird nochmal neu generiert durch die Informationen in diesen Dateien. Dieser Vorgang kann dann bis zu 48 Stunden dauern. Ihr bekommt dann jedenfalls eine Email an die in den Scripten eingetragene Emailadresse.

Wie der aktuelle Status eures Paketes ist und ob es gemeldete Bugs gibt und weitere Informationen gibt es auf http://packages.qa.debian.org. Dort sucht ihr einfach nach eurem Paketnamen.

## 7. Hilfe zur Selbsthilfe

Tja, nachdem ihr jetzt mal gesehen habt, wie man ein Debian Paket erstellt, wollt ihr vielleicht auch mal eines erstellen und stoßt vielleicht mal hier und da auf so manches Problem. Falls ihr dann keinen Debian-Entwickler kennt, der eure “nervigen” Fragen (Gruß nochmal an Daniel) beantwortet, würde ich euch folgendes empfehlen.

Schaut mal auf: http://www.debian.org/devel/ - Dort gibt es jede Menge Informationen für euch.

Weiterhin kann ich euch die Mailingliste: http://lists.debian.org/debian-mentors/ empfehlen, wo ihr auch eure Fragen stellen könnt.

Falls ihr nun keine eigenen Softwarepakete habt, um daraus Debian-Pakete zu erstellen, so gibt es auf http://www.debian.org/devel/wnpp eine Liste mit Paketen, die von euch gepflegt werden können oder die Hilfe brauchen.

Ich hoffe, mit diesen Informationen hab ich euch schon ein wenig weitergeholfen.

## 8. Schluss

Danke, dass du bis hier gelesen hast und vielleicht auch hoffentlich ein Paket erstellen konntest. Ich hoffe, die Dokumentation hat dir geholfen und vielleicht bringst auch du deine Pakte mit in Debian ein.

Euer Frank Habermann <lordlamer@lordlamer.de>

**Changelog**

1.0 - Initiale Fassung
1.1 - Korrekturen von Robert Walter