RPM im Eigenbau
Aus der Schule unseres RPM-Teams

von Oliver Burger, aka obgr_seneca

Immer wieder treten Fragen von Usern auf, wie man denn aus seinem selbstcompilierten Programm ein RPM baut. Um hier etwas Licht ins Dunkel zu bringen und vielleicht den einen oder anderen dazu anzuregen, sich mit dem Thema genauer zu befassen, hatte ich mir überlegt mal eine Anleitung anhand von Beispielen zu schreiben.

Man muss sich zuerst darüber im Klaren sein, dass es hier drei mehr oder weniger verschiedene Aufgabenstellungen gibt:

- der Rebuild eines vorhandenen src.rpm-Paketes (zum Beispiel aus dem Cooker)
- der Bau basierend auf einem vorhandenen src-rpm, das jedoch angepasst werden muss
- der Bau direkt aus den Sourcen

Vorarbeiten

Bevor man anfängt Pakete zu bauen, muss man noch ein paar Vorarbeiten leisten. So muss man dem System unter Anderem mitteilen, in welchen Verzeichnisse die einzelnen „Produkte“ des Paketbaus landen sollen. Dies geschieht in der Datei „~/.rpmmacros“ und sieht zum Beispiel so aus:

```
# Use the home directory for building.
# Needed dirs (or symlinks): BUILD RPMS/i686 SOURCES SPECS SRPMS

%_topdir            /home/oli/rpm-build/rpm/2008
%_tmppath             /home/oli/rpm-build/rpm/2008/tmp

%_signature           gpg
%_gpg_name             RPM-Team MandrivaUser.de <rpm@mandrivauser.de>
%_gpg_path            ~/.gnupg

%packager             Oliver Burger
%distribution         Mandriva Linux
%vendor              RPM-Team Mandrivauser.de <rpm@mandrivauser.de>

%distsuffix          mud
%distversion          2008.0

%_enable_debug_packages %{nil}
%debug_package          %{nil}
```

Eine kleine Erklärung hierzu: Zuerst muss man dem System angeben, in welches Verzeichnis gebaut werden soll, dies geschieht mit der Angabe

```
%_topdir            /home/oli/rpm-build/rpm/2008
```

Es wird also grundsätzlich ins HOME-Verzeichnis des Users gebaut, darin habe ich mir ein Verzeichnis „rpm-build“ angelegt, in dem alles landet, was irgendwie mit dem Bau von Paketen zusammenhängt. Darin, das ist meine eigene Konstruktion, das eigentliche Bauverzeichnis „rpm“ und (da ich in das gleiche Verzeichnis auch meine 2007.1er-Pakete baue) noch „2008“. Diese Ordnung muss man natürlich nicht übernehmen, sie hat sich bei mir mehr zufällig so entwickelt. Unterhalb des „%_topdir“ muss nun der Verzeichnisbaum für den RPM-Bau existieren, dies sind die Verzeichnisse BUILD, RPMS, RPMS/i586 bzw. RPMS/x86_64, SOURCES, SPECS und SRPMS.

Des Weiteren muss man noch ein (natürlich existierendes) temporäres Verzeichnis angeben, bei mir „tmp“ unterhalb des Bauverzeichnises.

Die weiteren Angaben sind nun mud-spezifisch. Zuerst die Angabe des Signatur-Verfahrens und des zu verwendenden Schlüssels, da unsere Pakete ja eine Signatur erhalten. Dann der Name des Paketbauers, in diesem Fall meiner, die Distribution, für die die Pakete gebaut werden, sowie die Angabe des Paketanbieters, außerdem noch die Version der Distribution und ein Suffix, bei Mandriva das „mdv“, bei uns eben „mud“.

```
%_enable_debug_packages %{nil}
%debug_package          %{nil}
```

Diese Angaben sorgen dafür, dass kein „debug“-Paket gebaut wird, dies wäre eher für Tests gedacht.

Als Weiteres gibt es noch die Datei „~/.rpmrc“, die Angaben über die CPU und die zu benutzenden Compiler-Flags enthält. Hier habe ich wegen breiterer Nutzungsmöglichkeit nur sehr begrenzte Einstellungen vorgenommen. Die von mir genutzte Datei stammt ursprünglich von Mandriva:

```
buildarchtranslate: i386: i586
buildarchtranslate: i486: i586
buildarchtranslate: i586: i586
buildarchtranslate: i686: i586

##original Mandriva Flag
##optflags: i686 -O2 -g -march=i686

##  Flags taken from
##  http://www.freehackers.org/gentoo/gccflags/flag_gcc3.html

## i686 / Pentium Pro
## optflags: i686 -march=i686 -O3 -pipe -fomit-frame-pointer

## Pentium III / Celeron2
##optflags: i686       -march=pentium3 -O3 -pipe -fomit-frame-pointer

## Pentium-M
##optflags: i686       -march=pentium-m -O3 -pipe -fomit-frame-pointer

## Pentium IV
##optflags: i686 -march=pentium4 -O3 -pipe -mfpmath=sse -fomit-frame-pointer
##-fforce-addr -falign-functions=4 -fprefetch-loop-arrays
##optflags: i686 -march=pentium4 -O3 -pipe -mfpmath=sse -fomit-frame-pointer
##-ftracer -fforce-addr -fprefetch-loop-arrays
##optflags: i686 -march=pentium4 -O3 -pipe -fomit-frame-pointer

## Athlon (AMD)
optflags: i686       -march=athlon -O3 -pipe -fomit-frame-pointer
##optflags: i686       -march=athlon-tbird -O3 -pipe -fomit-frame-pointer
```

Cooker-Rebuild

Der Cooker-Rebuild ist hierbei die einfachste Methode. Wenn man das für sich privat macht, reicht es, ein „rpm –rebuild Paketname.src.rpm“ einzugeben. Das System fängt dann an, vor sich hin zu arbeiten und wird irgendwann, wenn alle Abhängigkeiten erfüllt sind und keine Konflikte auftraten, eine Erfolgsmeldung abgeben. Die Pakete landen in dem in .rpmmacros angegebenen Verzeichnis, in meinem Fall „/home/oli/rpm-build/rpm/2008/RPMS/i586“ und können von dort aus mit „urpmi ./Paketname.rpm“ installiert werden.

Im folgenden Beispiel werden einige weitere Angaben gemacht, wie z.B. eine Kurzbeschreibung sowie eine ausführlichere Beschreibung des Paketes unter „Summary“ bzw. „%description“. Es werden vorher definierte Angaben Name, Version und Release angegeben. Dann folgt die Angabe des Source-Files, in diesem Fall kooldock-0.4.6.tar.gz, dieses muss in SOURCES liegen. Dann die Lizenz des Programmes und die Gruppe, unter der das Paket im Paketmanager geführt wird. Eine Liste der von Mandriva genutzten Gruppen findet sich unter http://wiki.mandriva.com/en/Development/Packaging/Groups auf der Mandriva-Seite.

Außerdem wird der URL der Projektseite der Applikation angegeben.

Die nächste Angabe ist wichtig, kann jedoch grundsätzlich abgeschrieben werden, die Buildroot ist das Wurzelverzeichnis, in dem der Paketbau dann tatsächlich stattfindet.

Nun müssen nur noch die Anforderungen für den Paketbau und die spätere Paketinstallation sowie eventuelle Konflikte mit anderen Paketen angegeben werden. Diese Anforderungen kann man oft den jeweiligen Projektseiten entnehmen oder man achtet auf die Ausgabe des „configure“-Durchlaufs. Manchmal stößt man jedoch auch erst auf eine fehlende Angabe, wenn „make“ mit einer Fehlermeldung abbricht.

Die Angabe hinter „Obsoletes:“ sorgt dafür, dass Vorgängerversionen automatisch deinstalliert werden und die Installation nicht mit einer Fehlermeldung abbricht.

Doch weiter im Text:

```
%prep
%setup -q

%build
make -f Makefile.dist
./configure --prefix=%{_prefix} --libdir=%{_libdir} --mandir=%_mandir

%make

%install
rm -rf %{buildroot}
%{makeinstall_std}

%clean
rm -rf %{buildroot}
```

Hier läuft nun der eigentliche Paketbau ab, die Makros „%prep“ und „%setup -q“ sorgen für die vorbereitenden Schritte: Erstellung des BUILD-Verzeichnises, Entpacken der Sourcen usw.

Unter „%build“ kommt dann der eigentlich Kompilierprozess, im Beispiel muss mit „make -f Makefile.dist“ erst mal das configure-Skript erstellt werden, dann folgt der aus dem Dreisatz bekannte „./configure“-Schritt mit ein paar Angaben wie einem prefix, dem Bibliotheksverzeichnis und dem Verzeichnis für die man-pages, von alleine würde alles in „/usr/local“ landen.

„%make“ ist dann ein einfaches RPM-Makro, das dem normalen „make“ aus dem Dreisatz entspricht.

Es folgt die eigentlich Installationsphase, installiert wird jedoch nur in die oben angegebene Buildroot. Eingeleitet wird diese von „%install“ gefolgt von „rm -rf %{buildroot}“, wodurch eventuelle vorherige Bauversuche aufgeräumt werden. „%{makeinstall_std} sorgt schließlich für das Äquivalent des „make install“ aus dem Dreisatz.

Es folgt noch die „%clean“-Phase, die nach dem Paketbau nochmals aufräumt.

Nach der erfolgreichen Kompilation und Installation der Sourcen fehlt nun noch die Angabe der im Paket enthaltenen Dateien, so dass rpm/urpmi dann auch wissen, welche Dateien zu welchem Paket gehören, sowie das Changelog:

```
%files
%defattr(-,root,root,0755)
%{_bindir}/kooldock
%{_datadir}/apps/kooldock/backgrounds/border-black/*.png
%{_datadir}/apps/kooldock/backgrounds/border-white/*.png
%{_datadir}/apps/kooldock/backgrounds/crystal/*.png
%{_datadir}/apps/kooldock/backgrounds/default/*.png
%{_datadir}/apps/kooldock/backgrounds/fadeout/*.png
%{_datadir}/apps/kooldock/backgrounds/fadeout2/*.png
%{_datadir}/apps/kooldock/backgrounds/fadeout3/*.png
%{_datadir}/apps/kooldock/backgrounds/osx/*.png
%{_datadir}/apps/kooldock/icons/crystalsvg/*/apps/kooldock.png
%{_datadir}/icons/crystalsvg/*/apps/kooldock.png
%{_datadir}/locale/*/LC_MESSAGES/kooldock.mo

%changelog
* Tue Oct 09 2007 Oliver Burger <rpm@mandrivauser.de> 0.4.6-1mud2008.0
- First Build for 2008.0

* Tue Aug 29 2007 Oliver Burger <rpm@mandrivauser.de> 0.4.6-1mud2007.1
- First Build for Mandriva 2007.1
```

Die Einträge der „%files“-Sektion erhält man am Einfachsten, wenn man sie primär leer lässt, der Bau des Paketes bricht dann mit der Meldung der zwar installierten aber nicht gepackten Dateien ab. Diese kann man nun in die „%files“-Sektion hineinkopieren, zu beachten sind gewisse Abkürzungen, so steht „%{_bindir}“ für „/usr/bin“, „%{_datadir}“ für „/usr/share“ und so weiter. Diese kann man aber auch aus dem Mandriva-RPM-Howto unter http://wiki.mandriva.com/en/Development/Howto/RPM entnehmen.

Wie man nun noch im Changelog sieht, entstand das Paket ursprünglich für die 2007.1 und wurde dann erneut (und ohne Änderungen) für die 2008.0 gebaut. Und jetzt:

Viel Spaß beim Paketbau!