RPM – eine kleine Einführung

von Andreas Radke (aka AndyRTR)

Allgemeines

Software wird in Form eines so genannten Quellcode geschrieben. Dieser ist das für Menschen leicht lesbare Format des Programmcodes. Wie der Quelltext aussieht, hängt ganz von der verwendeten Programmiersprache ab(C#, C++, Perl, Java, Assembler u.v.m.).

Es ist Aufgabe des Compilers, aus diesem Quellcode den für den Prozessor lesbaren ausführbaren Binärcode aus Nullen und Einsen zu erzeugen. Diese Aufgabe übernimmt in der Linux-Welt fast immer der GNU C Compiler(gcc). Dieser beherrscht zwischenzeitlich neben C# auch eine Vielzahl anderer Programmiersprachen und ist der Standardcompiler in der OpenSource Bewegung. Nachdem nun ein so genanntes Binärfile (weil aus Nullen und Einsen bestehend) vorliegt, stellt sich die Frage der optimalen Installation

Die Nutzer aus der Windows-Welt wissen, dass Windows-Programme alle eine Setup.exe mitliefern. Diese enthält alle nötigen Informationen, um das Programm am rechten Ort zu installieren.

Unter Linux gibt es nicht nur ein einziges gültiges Dateisystem oder eine einheitliche Distribution. Aber es gibt Empfehlungen für einen einheitlichen Dateisystemstandard[2], den Filesystem Hierachy Standard(FHS).

Die einfachste generische Anweisung gibt der Softwareautor vor und diese liegt dem dritten Schritt des magischen Dreisatzes(configure / make/make install), dem "make install" zu Grunde. In diesen Vorgaben wird bestimmt, an welche Stellen im Dateisystem die zuvor kompilierten Binärdateien kopiert werden.

Der RedHat Package Manager

Wozu also ist also solch ein Paketmanger da?
Der Paketmanager RPM ist ein mächtiges Werkzeug aus der Schmiede von RedHat[3]. Ein Programmpaket=RPM ist eine Sammlung verschiedener Dateien mit einer Reihe von Steueranweisungen. Der Paketmanager hat eine Vielzahl von Aufgaben.

Mit ihm können Programmpakete installiert und auch wieder leicht rückstandsfrei deinstalliert werden. Auch eine Option für Updates ist vorgesehen. Eine weitere wichtige Aufgabe des Paketmanagers ist es, so genannte Abhängigkeiten zu beachten. Es liefert dem Anwender und anderen Anwendungen wichtige Informationen über das Softwarepaket. Es hat auch eine Reihe von Schutz- und Sicherheitsfunktionen eingebaut.

Was macht eigentlich rpm?

Er übernimmt ganz einfach den letzten Schritt des "make install", liefert eine Reihe zusätzlicher Funktionen und versucht, die Softwareverwaltung einheitlich und bequem zu organisieren. Der gerade beschriebene Weg der Installation per Dreisatz ist sehr mühsam und oft nur von erfahrenen Linux-Usern schnell zu beschreiten.

Abhängigkeiten

Sehr viele Programme erfordern so genannte Abhängigkeiten. Das können zum einen Programme oder Programmbibliotheken (="libs") sein, die für den Compilierprozess erforderlich sind oder aber später für die Benutzung nötig sind. So braucht zum Beispiel das Brennprogramm K3B eine lange Liste von Programmen als Voraussetzung für seinen erfolgreichen Betrieb: die KDE-Oberfläche, die wiederum den X-Server benötigt und die CD-Verwaltungs- und Brenn-Backends (= die dahinter stehenden Kommandozeilenprogramme) zum Brennen.

Um diese Abhängigkeiten kümmert sich der Paketmanager rpm. Er sorgt dafür, dass sich das Paket erst dann installieren lässt, wenn alle anderen benötigten Programme installiert sind. Es stellt somit die richtige Funktion des Paketes sicher.

Der ursprüngliche RedHat Package Manager steigt mit der Installation immer genau dann aus, wenn eines der als Abhängigkeit vorgegebenen Pakete nicht auf dem System installiert ist. An der Stelle knüpfen Tools an, die versuchen, alle benötigten Pakete zu berechnen und dann auch noch automatisch zu installieren von CD oder aus dem Internet. Vertreter dieser Tools für rpm sind die Mandriva-Eigenentwicklung "urpmi"[4], "smart"[5](mit Connectiva eingekauft) und "apt4rpm"[6]. Dabei handelt es sich nicht um eigenständige Programme sondern um aufsetzende Scripte - so genannte Wrapper -, die nach umfangreichen Berechnungsmethoden alle Features von rpm ausnutzen und dadurch Zusatzfunktionen anbieten können.

Diese Tools berechnen anhand der in den Paketen hinterlegten Abhängigkeiten eine Liste aller zur Installation oder für ein Update benötigten weiteren Pakete. Meist sind sie so konfiguriert, dass man ihnen bestimmte Softwarequellen mitteilt, in denen sie nach den noch benötigten Pakten selbständig suchen können. Das sind in der Regel die Installations-CDs/DVD und Quellen im Internet. Diese Quellen nennt man auch Repository. Die Tools stellen fest, ob alle erforderlichen Pakete zur Installation bereit stehen und laden diese ggf. aus dem Internet herunter.

Was noch viel wichtiger ist: rpm und die aufsetzenden Tools verhindern, dass man beim Pakete deinstallieren Fehler macht. So-

lange ein Paket noch für ein anderes installiertes Paket benötigt wird, verweigert rpm beharrlich die Deinstallation.

Beispiele

Mit einem dieser Tools kombiniert, wird die Softwareverwaltung unter Linux zum Kinderspiel. So installiert ein "urpmi k3b" das grafische Brennprogramm K3B und achtet auch gleich darauf, evtl. noch fehlende Pakete, zu denen so genannte Abhängigkeiten bestehen, automatisch mit zu installieren.

So ist es problemlos möglich, zunächst nur eine Minimalinstallation der unbedingt nötigen Pakete vorzunehmen. Das ist dann kaum mehr als der Kernel und rpm selbst. Hat man noch urpmi dazu, reicht ein "urpmi k3b" und die Paketverwaltung sollte es schaffen, die ganze grafische Oberfläche nachzuinstallieren und alles, was man noch zum brennen braucht.

Die Datenbank dahinter

RPM legt Informationen über installierte Pakete in einer gemeinsamen Datenbank im Verzeichnisbaum unter /var/lib/rpm/Packages ab. Darin enthalten sind sämtliche von Programmpaketen installierte Dateien mit ihrem Installationspfad. Das ermöglicht später auch die rückstandsfreie Deinstallation. Zusätzlich sind Angaben zum Paketnamen, der Versionsnummer, Ersteller des Paketes mit eMail-Adresse, CPU-Architektur des Zielbetriebssystems (i386-i686, x64, alpha, sparc, powerpc usw.), eine Kurzbeschreibung des Paketes und anderes mehr gespeichert. Die Größe der Datenbank ist abhängig von der Anzahl der installierten Pakete. Bei mir beläuft sich die Größe dieser Datenbank zwischenzeitlich auf mehr als 70MB - durch meine Paketebautätigkeit bedingt.

Updates

Neben der kontrollierten Installation und der rückstandsfreien Deinstallation kann rpm auch zum Erneuern von Paketen verwendet werden. Anhand eines Vergleichs der Versionsnummern gleichnamiger Pakete kann rpm ein Update erkennen und durch Deinstallation des alten Paketes und Installation des neuen Paketes ermöglichen. Wenn die Paketebauer gute Arbeit geleistet haben, bleiben eingerichtete Konfigurationsdateien erhalten oder werden zumindest gesichert.

In Zusammenarbeit mit den Tools wie urpmi, smart oder apt ist es so möglich, eine rpm-basierende Linux-Distribution lange Zeit ohne Neuinstallation zu aktualisieren. Ein ganz extremes Beispiel hierfür bildet der Entwicklerzweig Cooker [7] von Mandriva. Dieses System bekommt fast immer täglich neue Pakete zum Test und will ständig aktualisiert werden.

Sicherheit wird groß geschrieben

RPM ermöglicht Sicherheitsfunktionen wie eine elektronische Signatur (gnupg) der Pakete und die Bildung von Prüfsummen(md5sum). Beides soll gewährleisten, dass die Echtheit der Pakete überprüft werden kann und man kein "faules Ei" bekommt.

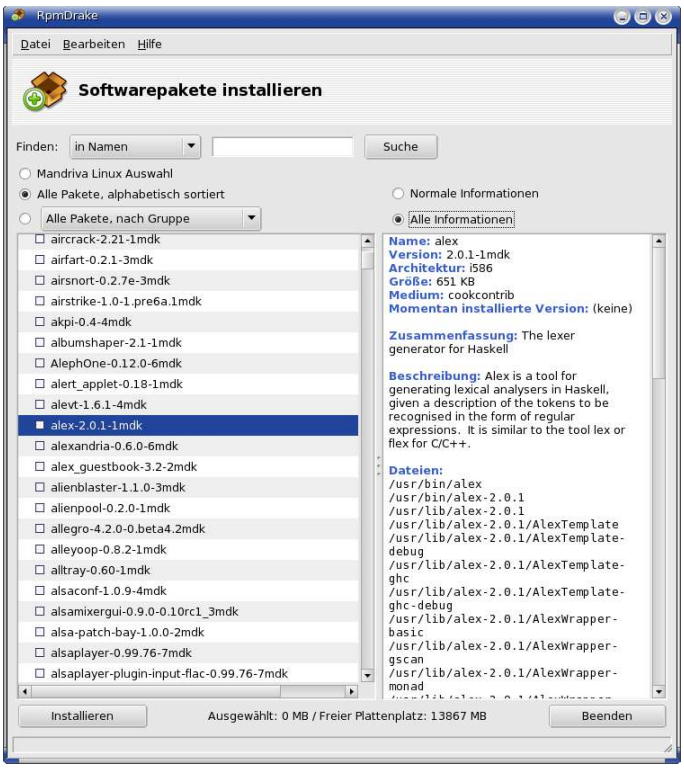
Paketebau

RPM ist auch gleichzeitig das Programm zum Erstellen der Pakete. Dazu folgt in einer weiteren Ausgabe ein gesonderter Artikel.

Die GUIs

Bis jetzt war das alles recht trockene Materie. Damit auch die Mausschubser Spaß an Linux haben und auch die fortgeschrittenen User nicht alle Befehle ständig im Kopf haben müssen, wurden ansprechende, leicht zu bedienende grafische Oberflächen (**G**rafical**U**ser-**I**nterface) entwickelt. Obwohl sie sehr leicht mit der Maus zu bedienen sind, machen sie nichts anderes, als an Hand eines bestimmten Mausklicks einen Befehl an die dahinter stehenden Konsolenprogramme rpm bzw. urpmi, smart oder apt zu schicken.

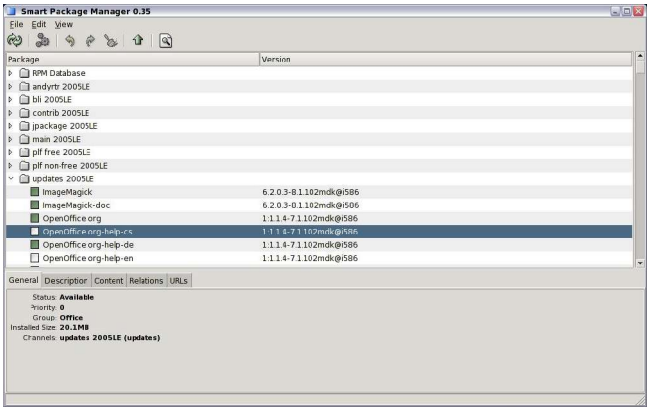
Mandriva nutzt derzeit die Eigententwicklung urpmi und hat dieses sehr gut in das Mandriva Control Center (MCC) integriert.



Dieser Bestandteil nennt sich rpmdrake.

Die Alternativen

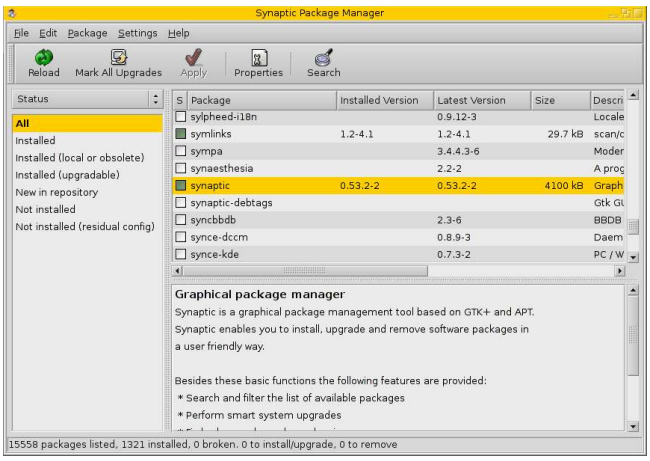
smart und apt4rpm sind beides auch Kommandozeilenprogramme. Beide sind in der Funktion mit urpmi vergleichbar. Eine grafische Oberfläche bringt smart von Haus aus mit:



Die smart GUI

Smart hat den Anspruch, DER Paketmanager der freien Betriebssysteme zu werden. Neben rpm beherrscht es alle wichtigen Paketformate unter Linux: Debians apt-deb, Slackwares Paketformat, urpmi-Quellen und natürlich rpms. Alle jeweils in loser Form oder in eingebundenen Repositories. Die Chancen stehen gut, dass sich hier bald ein neues Standardprogramm herausbildet.

Für apt gibt es verschiedene grafische Oberflächen. Am bekanntesten ist hier synaptic und sein KDE-Pendant kynaptic. Beide sind in der Bedienung der smart-GUI sehr ähnlich. Sie waren zuerst da und bildeten die Vorlage für smart.



Synaptic

Ein paar Tips zum Abschluss:

Kurz seien noch wichtige Befehle bei der Arbeit mit rpm und urpmi genannt:

Paket installieren: `rpm -i Paketname` oder `urpmi Paketname`
Paket deinstallieren: `rpm -e Paketname` oder `urpme Paketname`
Paket updaten: `rpm -U Paketname` oder `urpmi Paketname`

Hilfreich sind auch: "`urpmf Dateiname`" - zeigt an, in welchem Paket sich die gesuchte Datei befindet.

"`rpm -qa | grep Paketname`" - durchsucht alle installierten Pakete nach dem angegeben Paketnamen, ob es installiert ist und zeigt bei Fund die Versionsnummer an.

`rpm -qi Paketname` oder `urpmq -i Paketname` zeigt alle wichtigen Informationen zu einem installierten Paket (Versionsnummer, Paketebauer, Beschreibung usw.) außer der Liste der enthaltenen Dateien.

Noch ein Hinweis: Neben rpm existieren noch als Konkurrenzsysteme zur Softwareverwaltung in der Linuxwelt: das System von Debian aus deb-Paketen und apt als Pendent zu urpmi. Außerdem bieten Gentoo-Linux und die BSD-Betriebssysteme vergleichbare Systeme, die jedoch primär nicht mit Binärpaketen, sondern mit Quellcodepaketen arbeiten. Auch wenn immer wieder behauptet wird, dass rpm von diesen System die schlechteste Abhängigkeitsverwaltung hätte, konnte es bislang niemand technisch begründen. Warum auch? Wenn ein Paket eine bestimmte Abhängigkeit hat, kann dafür die Paketverwaltung nichts.

In der nächsten Ausgabe geht es dann ins Eingemachte:
eigene Pakete bauen!

Euer RPManiac
AndyRTR

[1] http://www.rpm.org/
[2] http://www.pathname.com/fhs/
[3] http://www.redhat.de/
[4] http://www.urpmi.org/
[5] http://smartpm.org/
[6] https://moin.conectiva.com.br/AptRpm
[7] http://qa.mandriva.com/twiki/bin/view/Main/CookerHowTo