Comparison of binary package formats

Purpose of binary packages

The general purpose of binary packages (no matter what format they use), or to be more precise, package management systems, is to provide a prebuilt software including related metainformation in a from that can be used to enable the users and administrators to easily maintain (i. e. install, upgrade, remove etc.) the systems and applications programs so that the system remains consistent and its resources are used in a efficient way.

To achieve this, several standard methods are in use such as centralized package repositories, dependencies detection mechanisms, automated updates etc.

Binary package formats

There are various package formats which differ in their internal structure, features or available functional overlays for yet advanced management, but probably most widespread are RPM, deb and TGZ formats.

The RPM (the abbreviation stands for Redhat package management according to the original vendor) format is used e. g. in SUSE/openSUSE, Mandriva, CentOS and of course also RHEL and Fedora.

The deb(ian) format can be found especially on Debian-based distributions such as Debian or Ubuntu and TGZ is actually not a standardized package format as there are several (in most cases smaller) distributions (ArchLinux, Slackware, ...) which use tar&gzip to store their binary packages (in wide range of formats).

In this paper, I would like to concentrate on the comparison between deb and RPM format as these formats are most widespread nowadays and provide the richest features. There is also a conversion tool between RPM and deb called alien under development.

RPM

A standard way of packaging a program using RPM is to create a SPEC file containing all the metainformation about the package such as name, version, release (often abbreviated as N-V-R), runtime and buildtime dependencies, various installation scripts (called scriptlets) or triggers (event-driven scripts which can be executed when a specific action of another package occurs – like installation, update, uninstallation, ...) etc.

Further, an SRPM (source RPM) is created from the SPEC file and all source files which can be used to easily build the package or distribute the

package sources. From this SRPM one or more final RPMs are created to be installed on the target systems. There is also useful program for automated checking well-formed SPEC files and RPMS called rpmlint.

deb

The deb package structure consist of three main components: first it is the debian-binary file containing the format version number, second, there is a control.tar.gz package entailing all the metainformation about the package similar to the RPM SPEC file, however separated into many files (such as the control file with general metainformation or installation scripts which are each in a separate file too).

Finally as third, there is a data.tar.gz package where all the binaries and other applications programs files are stored. The counterpart of RPM's rpmlint is lintian.

Comparison

First of all, I should anticipate that there are several aspects which are hard to compare but important to consider. One of them is that each distribution usually defines a more or less precise set of packaging rules which influence the usability of their packaging systems in a big way. Moreover, users usually do not choose their distribution according to the package management system it uses and although it would be technically possible to use any of the package management systems on every distribution, there are hardly more users doing it.

From these (and yet more) reasons I'll just point out some pros and cons of both RPM and deb package formats.

- RPM
  + DeltaRPM which saves users traffic by downloading only deltas (diffs) between package versions
  + GPG signed packages
  + very good support for multilib (e. g. both 32bit and 64bit) systems
  + proper autogeneration of runtime dependencies in most of the cases
  - no way to handle suggested (not necessary) dependencies
  - non-standard format (rpm2cpio is needed for extraction)
  - rich features mean less simple design

- deb
  + suggested dependencies
  + package priorities used to classify packages by their importance

+ standard format (tar)
+ some developers appreciate decentralized design of stored metainformation
- no direct multilib support
- no file dependencies

Conclusion

During my presentation and subsequent discussion, there were several valuable notices from my colleagues that I highly appreciate. Two of them I listed below:

- Although the deb packages can signed using GPG, this option is not used presently in Debian.
- Although the deb package format intended to be desktop-based and therefore it is interactive whereas the RPM package format intended to be server-based and as such it is strictly non-interactive, surprisingly in these days RPM is used rather on desktops than servers.

Reference

- DEB:
http://en.wikipedia.org/wiki/Deb_(file_format)
http://tldp.org/HOWTO/html_single/Debian-Binary-Package-Building-HOWTO/
http://www.linuxdevices.com/articles/AT8047723203.html
http://www.debian.org/doc/FAQ/ch-pkg_basics.en.html
- RPM:
http://www.rpm.org
http://en.wikipedia.org/wiki/RPM_Package_Manager
http://docs.fedoraproject.org/drafts/rpm-guide-en/
http://genetikayos.com/code/repos/rpm-tutorial/trunk/rpm-tutorial.html
http://www.ibm.com/developerworks/library/l-rpm1/
http://www.abclinuxu.cz/clanky/navody/rukovet-balice-rpm-i-uvod
- TGZ:
http://www.abclinuxu.cz/clanky/system/balickovaci-system-arch-linuxu-1-format-balicku
- Others:
plkárna FI: thread Distro showdown - let your voices be heard (21. 9. 2007)
http://kitenet.net/ joey/pkg-comp/
http://linux.die.net/man/1/alien
http://www.howtoforge.com/converting_rpm_to_deb_with_alien