Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 12, Number 1 (2019) pp. 37-44

http://www.ripublication.com

A Comparative Study of various Linux Package-Management Systems

Varun A[1], Rahul M Patil[2], Pratibha Kantanavar[3] and Shobha G[4]
[1,3]Department of Electronics and Communications Engineering
[2,4]Department of Computer Science and Engineering, Rashtreeya Vidyalaya College of Engineering, Bangalore, Karnataka, India

Abstract

A package-management system (package manager) is a software tool, or a set of software tools that is used to automatically manage the installation, upgradation, configuration and removal of software packages, Usually, Linux based package managers are classified into frontend managers like Yellowdog Updater Modified (YUM), Dandified (DNF) YUM, Advanced Packaging Tool (APT), Entropy, Pacman and Portgage, and backend managers like Debian and Redhat Package Manager (RPM). The paper provides an inclusive survey about the different package managers highlighting their features, use cases and limitations.

Keywords: APT, Debian, DNF, Pacman, RPM, YUM, Zypp, Portgage

INTRODUCTION

When Linux was first created, software was packaged in the form of tar balls and sent to different systems for installation. A tar ball is a single archive file that can be created with the help of a single tar command, and it usually comprises of multiple files. So, a software application such as a web-based project or a word processor would gather up all the files needed for that project into a tar ball. This tar ball would then be passed to the user for installation. The user would have to untar the tar ball and then use the application. This technique however, had a lot of disadvantages.

There was no way to manage the software once the tar ball was installed. It was difficult for the user to know the version of the software, as it never came along with the tar ball. Besides, any kind of upgrade of the software version was also impossible once the tar ball was installed, as the files were spread across the entire system and there were no means to group them for carrying out the upgradation process. The second disadvantage was that if the software was dependent on any other software, the dependency must be manually installed by the user. This became extremely cumbersome for the users. This resulted in the advent of package-management systems, commonly known as package managers. Package managers help to resolve the software management and version number problems by packaging meta-data along with the actual software. The meta-

data has a detailed description about the number of files in the package, list of all files, version number, description about the package, information about the packager and any other additional information the developer may want to include. The meta-data can also include the dependencies needed for the software packages to work, and scripts that carry commands to do tasks like creating directories, adding user accounts or turning on a service. Usually, Linux based package managers are classified into frontend managers like YUM, DNF, APT, Entropy, Pacman and Portgage and backend managers like Debian package manager and RedHat package manager.

BACKEND PACKAGE-MANAGEMENT SYSTEMS

Backend package managers are low level package managers that are used for installing, removing and upgrading software packages. The commonly used backend package mangers in Linux environments are Debian package manager and RedHat package manager.

Debian Package Manager

The Debian package manager packages all the files into a package with the .deb extension [1]. These are archives that comprise of three components embedded within them.

- The first component is the tar archive which is called data that comprises of all the source codes and the files to be installed. This component is usually compressed using bz2, xz, gz or lzma. However, the compression is usually optional.
- The second component is an archive called control that comprises of the meta-data of the package that includes name, version number, release number and other optional arguments. This component is also usually compressed using bz2, xz, gz or lzma. However, the compression is usually optional.
- The third component is the binary text file that comprises of the Debian format number which is 2.0 at the latest.

Debian packages are installed, upgraded and removed by DPKG, which is a low-level package manager that manages these packages. DPKG is written using C, C++ and Perl. The major Linux distributions namely Debian, Linux mint and Ubuntu use the Debian package manager.

Redhat Package Manager

The Redhat package manager packages all the files into a package having the. rpm extension. It is a binary package format. These are archive files from which the contents including the source codes can be extracted easily [2]. It is a low-level manager that installs, upgrades, removes and also provides information about the RPM based

packages. Like Debian managers, they also contain two major components, namely the compressed source codes and files and the meta-data that is associated with the package. RPM packaging also has other distinct advantages like straight forward installation and uninstallation and are also available for all distributions of Linux. They also have features like cryptographic verification and inclusion of original source achieves which makes verification easier. They also have automatic build time dependency evaluation, making the packaging operation more sophisticated.

FRONTEND PACKAGE-MANAGEMENT SYSTEMS

Frontend package managers are the high-level managers that operate over the lower level backend managers. These managers are more user friendly and can also resolve the various dependencies automatically.

Advanced Packaging Tool (APT)

The Advanced Packaging Tool commonly abbreviated as APT is a high-level packaging tool. It acts as a front end for the low-level Debian based DPKG packaging tool of Linux operating systems as well as OpenSolaris based systems [3]. It also comprises of aptitude, which gets invoked in either the GTK-based method or the cursor-based method. For apt, repositories are the means in which their software is found, and their dependencies resolved. Repositories are essentially directories that consist of the packages, as well as an index file. The Debian project maintains a central repository, which holds over 25,000 packages that are readily available for download and installation. An advantage of the apt package manager is that an infinite number of further repositories can be added to apt's configuration file, which enables them to be successfully queried by apt. Packages in a particular repository can be installed directly without specifying any source. It also performs checks and keeps the packages up to date.

The apt package manager supports install, update, upgrade and removal of files. The update operation takes place in two steps. The first step involves a simple update after the file has been changed, that is immediately followed by upgradation. The apt package manager is more of a binary package manager, and as it does not use the source codes, it fails to provide a lot of customizability. However, the user can customize the code by downloading the source code, making suitable changes to it, and then binarize the files so that it can be packaged. Although it is user friendly compared to the lower level package managers, the commands are usually quite confusing, and it does not offer the versatility that other package managers provide.

One of the main features that apt provides is the way it invokes the dpkg backend. The list of packages that are to be either removed or installed, are topologically sorted. It then invokes dpkg in the most effective order as interpreted from the new topologically sorted information.

YUM Package Manager

The Yellowdog Updater runs as a frontend system for the lower level backend RPM system. The YUM manager can be made to work using both command line arguments as well as graphical interface [4]. Gyum, yum extender and kyum are used as graphical front ends. However, the command line based YUM packaging is technically faster as compared to the usage of the Graphical User Interface (GUI). The YUM package manager is very easy to use, and any installation or removal would only require the word to be preceded by the yum keyword. YUM mainly checks for the dependencies by loading the various header files along with the package meta-data present in the system as a part of the background operation. After completing the dependency resolution, YUM lists the various additional packages that are required and asks the user for consent to install them. After installation, YUM also has a build in verification tool that ensures that the installation has taken place correctly and makes sure that the package is signed as illustrated in Figure 1.

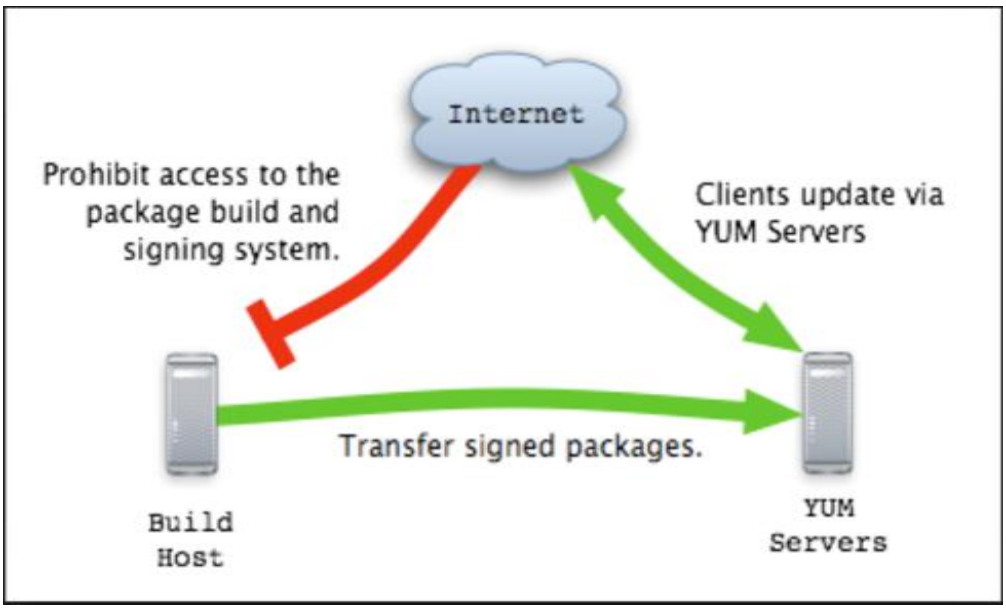

Figure 1: YUM packaging process

However, removal of packagers must be done with utmost care as it would have an unpleasant side effect while removing dependencies. For updating, the update command can be used preceded by the yum command, or else a GUI like pup can be used. The YUM update daemon also has the capacity to update the whole system. YUM also provides its own customized shell where a sequence of operations of can be carried out in a pipeline. An interesting situation is where the cache is deleted, package is installed and then deleted. YUM also has the capability to utilize RPM's roll back function, wherein a new program is installed, but it has to be restored to the original state of the system due to incompatibilities. To leverage this feature, first the system

has to be backed up to the current status. This means that at least one backup must be existing. YUM is more convenient and user friendly as compared to APT package manager. However, it also deals with binaries and not with source codes. So, if the codes have to be edited then they have to first download the required source codes, followed by modification of these source files, after which they are finally converted to the respective binaries.

Dandified YUM Package Manager

The Dandified YUM package manager, or DNF for short, is the advanced and improved version of the YUM package manager. DNF has been the default package manager for Fedora since version number 22. Similar to YUM, DNF also performs the packaging tasks by sitting on top of the RPM backend. DNF was initially written Python, but now efforts are being put into porting it to C [5].

YUM package manager faces a lot of issues with regards to performance and memory usage. DNF has been developed to solve and improve these problems. YUM uses an iterative dependency resolution method, which is an extremely time-consuming process. Instead of the same iterative method, DNF uses a third-party dependency solver known as libsolv. Libsolv has been proven to perform better, because of some primary reasons. These reasons are:

- Utilizes a dictionary approach to store and retrieve dependency and package information.
- Uses a heavily researched concept to resolve package dependencies, known as satisfiability.

Apart from libsolv, other dependencies used by DNF are libdnf, librepo and libcomp. Libdnf is a new library that includes most of the functionality moved from the Python implementation. It is currently being used as the package backend by PackageKit. It is essential a high-level API for DNF and contains most of DNF's features that have been ported to C/C++. Librepo is an API in both C and Python, which is responsible for downloading Linux repository packages. As an alternative to the yum.comps library – responsible for defining how the packages are bundled during installation – DNF uses a library written purely in C, known as libcomps.

To summarize, some of the main features that makes DNF better than YUM are:

- Operations that are extremely memory intensive have been optimized, and hence the memory usage has been improved.
- It is supported by both Python 2 and Python 3, and this is really important as Python 2 is scheduled to reach its EOL soon.
- The dependency resolution is done by using much better and faster technology.
- Has extensive documentation available for the Python APIs.

Being the next-generation version of the YUM package manager, it has a lot more advanced features, and provides a lot of improvements over YUM's existing features. Migration from YUM to DNF has become even more popular today, as YUM is no

longer going to be open sourced. However, DNF like YUM, is a binary package manager and source code level editing requires downloading the source code and then converting it to the respective binaries after editing.

Portgage Package Manager

Portgage is a package manager that was created by Gentoo Linux, although now it is commonly used by other variants of Linux and the Chrome Operating System. The term Portgage comes from the BSD style of package management, which utilizes a collection of ports [6]. It is usually operating system independent due to which Gentoo is also called as a meta distribution. Portgage is usually extremely flexible and is written using Python programming language. It mainly comprises of two parts, namely emake and emerge. Emake is the component that does the actual building and packaging of various target files, while emerge is the component that acts like an interface to emake helping it by resolving all the dependencies, managing the emake repository and other related issues. Emake and emerge are the backend and frnotend of the Portgage package management respectively. A GUI called Porthole is also available for working with Portgage.

Portgage is characterized by an important factor of compilation from source code, followed by packaging providing more customizability to the users in terms of compiler and the target application. It also has various features such as resolving cross platform dependencies, managing a database comprising of installed packages and managing the portgage tree as well as the ebuild repository and parallel packaging. It maintains three types of packaging namely stable, keyword masked which means that the target package has not been sufficiently tested and hard masked which it is unstable or insecure. Although it source code-based packager, Gentoo also supports binary packaging. Usually the packaging process is masked to ensure that it happens without user intervention.

Pacman Package Manager

Pacman is the default package manager for Arch Linux as well as two of its independent entities namely KaOS and Frugalware. It is in accordance to the guidelines of Arch Linux to keep it simple. The package manager is completely written in C language making it the fastest package manager available today. It does not act as frontend for any of the low-level package managers. The components packaged by Packman have an extension pkg.tar.xz. The package is generated using the makepkg command in a specialized directory in which the PKGBUILD bash script is present. The PKGBUILD script is similar to the RPM specfile that contains the instructions that is utilized in packaging the component.

ARCH Linux mainly uses two types of repositories namely Pacman repositories which is used to install Pacman related software and Arch User Repository (AUR) that consists of 30000 different PKGBUILDs for packages that cannot be accessed by Pacman itself. However certain AUR helper softwares are available that mainly take up

required components from both the repositories and link the missing dependencies. One such helper software is the Yaourt helper software.

Pacman however does not have a user friendly or intuitive syntax for beginners. It also does not have officially supported graphical user interface like the other package managers. In terms of customizability, Pacman is a binary package manager and hence it does not provide much of support to the user. However, the Arch Linux also comes with the Arch Build System (ABS) that is similar to Portgage based trees that contains all the PKGBUILDs as well as the supporting software needed for packaging. However, ABS does not come preinstalled with Linux and has to be installed separately by the user. Once installed it is possible for the user to customize the files as required or even add new files. In terms of development, Pacman is more convenient as the user has to only write a PKGBUILD which is far easier that the specfile or emake file used by other package managers. Pacman also provides lesser features compromising the speed of packaging it provides. It does not support auto removal of the files and either does it support the usage of wildcards. Overall, Pacman provides the basic features required as well as high speed packaging and is very convenient for packaging larger file sets.

ZYpp Package Manager

ZYpp also called as libzypp is a frontend package manager of openSUSE based Linux. It has sufficient libraries to utilize both the command line as well as the graphical user interface of openSUSE called as Yast2. It is the frontend package manager for RPM and provides various advanced features like resolving dependencies as well as managing repositories. The pckage manger has been developed using C++ making it quite fast as compared to other managers like YUM and DNF which are written using Python.

Zypp is one of the most beginner friendly package managers owing to its intuitive syntax. Upon combining with Yast2, which is the GUI package manager of Zypp it becomes very powerful. Zypp is a binary package manager and hence does not provide much of support to the user on the customization metric. For customization, the source packages have to be downloaded and then installed further which the user can customize it. Since it is based on RPM it requires a specfile comprising of all the targets or deliverables to be packaged which has to be added to the right location in addition to the source codes. It is a feature rich package manager comprising of all the features provided by DNF except the group install feature. However, for group installs patterns can be used similar to APT tasks. One step installation can also be done using Yast2 graphical user interface, where by upon clicking on a website havin ymp extension assuming that the browser has been set up properly, it should start downloading. Overall, Zypp is highly efficient in terms of speed as well as customizability.

CONCLUSION

The paper focusses on various packaging techniques user for Linux based operating systems. Each of the packaging techniques have their own advantages and

disadvantages. Table 1 provides a summary of various features of all the package managers.

Table 1: Summary of package managers

Package manager	Binary/Source	Install from URL support	GUI enabled	File Extension
APT	Binary	No	Yes	.deb
DNF	Binary	Yes	Yes	.rpm
Pacman	Binary	Yes	No	.pkg.tar.xz
Portgage	Source	No	No	.ebuild
YUM	Binary	Yes	Yes	.rpm
ZYpp	Binary	Yes	Yes	.rpm

It can be summarized that on grounds of user customizability Portgage has an edge over other packaging methods due to the source code-based packaging technique user. In terms of user friendliness DNF and YUM provide better support while Pacman and Zypp utilizes lesser time in the packaging process as it is completely written using C making it the fastest manager. Zypp is also highly feature rich in nature. Based on the conditions, corresponding performance and feature each one as needed can be selected.

REFERENCES

[1] Robles, G., J. Gonzáles-Barahona, and M. Michlmayr, 2005, "Evolution of Volunteer Participation in Libre Software Projects: Evidence from Debian", *Proceedings of the 1st International Conference on Open Software Systems*, Graz, Austria.

[2] Adam Miller, Petr Kovar, Marie Dolezelova, Miroslav Suchy, 2018, "Red Hat Enterprise Linux 7 RPM Packaging Guide", vol 3, pp 15-60.

[3] Pilar Manzano Garcia, Isaakidaas Marios, 2013, "Package Management in Linux", *Fedora Project white papers*.

[4] Ian Shields, 2015,"YUM and RPM package management", *IBM white paper publication*.

[5] Petersen, Richard, 2015, "Installing and Updating Software: DNF, GNOME Software, Packages, DnfDragora, and RPM", *Fedora 28 Edition*.

[6] Remco Bloeman, Chintan Amrit, 2014, "Gentoo Portgage package dependencies", *11th Working Conference on mining software repositories*.