



# LINUX PACKAGE MANAGEMENT

---

AN OBSERVATION ON THE  
TECHNOLOGIES THAT MANAGE  
APPLICATIONS IN LINUX

# LINUX PACKAGE MANAGEMENT

AN OBSERVATION ON THE TECHNOLOGIES THAT MANAGE  
APPLICATIONS IN LINUX

---

## DOCUMENT DETAILS

---

**Last saved:** 19.07.2004

**Total Pages:** 16

---

## AUTHOR DETAILS

---

**Ric de France (rdefrance <at> gmail <dot> com)**

Phone: +61 (0) 412 945554

---

## COPYRIGHT

---

© Ric de France 2004.

This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

---

## CONTENTS

---

1	Synopsis .....	5
2	Introduction .....	6
2.1	Background .....	6
2.2	Objectives of this report .....	6
2.3	Intended Audience .....	6
2.4	Assumptions .....	7
2.5	Limitations of this report .....	7
3	What is a “Package Management system”? .....	8
3.1	Package management equivalent in Microsoft Windows .....	8
3.2	Package management in Linux .....	8
3.2.1	<i>Not utilising a package manager</i> .....	8
3.2.2	<i>Building your own packages</i> .....	8
4	Criteria for Measuring Package Management systems .....	10
4.1	Application ratings .....	10
4.2	Infrastructure ratings .....	11
5	RPM .....	12
5.1	Comments on RPM .....	12
5.2	Rating RPM Package Infrastructre .....	12
5.3	Package management applications using RPM .....	12
5.3.1	<i>rpm</i> .....	12
5.3.2	<i>apt4rpm</i> .....	13
5.3.3	<i>up2date</i> .....	13
5.3.4	<i>urpmi</i> .....	13
5.3.5	<i>yum</i> .....	14
6	DEB .....	15
6.1	Comments on DEB .....	15

6.2	Rating DEB Package Infrastructre .....	15
6.3	Package management applications using DEB.....	15
6.3.1	<i>apt</i> .....	15
7	TGZ .....	16
7.1	Comments on TGZ.....	16
7.2	Rating TGZ Package Infrastructre .....	16
7.3	Package management applications using TGZ.....	16
7.3.1	<i>pkgtool (with installpkg, removepkg, and upgradepkg)</i> .....	16
7.3.2	<i>swaret</i> .....	17
7.3.3	<i>SlackUpdate</i> .....	17
7.3.4	<i>slapt-get</i> .....	18
8	SRC.....	19
8.1	Comments on Source (SRC).....	19
8.2	Package management applications using SRC .....	19
8.2.1	<i>Portage</i> .....	19
8.2.2	<i>sorcery</i> .....	20
9	Summary.....	21
10	Conclusion.....	22

---

## 1 SYNOPSIS

---

GNU/Linux is an Open Source operating system comprising of the Linux kernel and applications. These systems commonly referred to as “Linux distributions”. Distributions come technology to manage the applications, commonly known as a “package manager”. The intention of this report is to observe the differences between the package management systems used by Linux distributions.

*“The greatest aspect of Linux and Open Source is the amount of choice available. One of the worst aspects of Linux and Open Source is making a choice.”*

### 2.1 BACKGROUND

The GNU/Linux operating system has gone from a small kernel project started by Linus Torvalds [1] to fully-fledged operating system that is fully scalable and ready for the Enterprise [2].

The complete operating system of the Linux kernel and additional applications is known as GNU/Linux, and is commonly known as “Linux distributions” (or just “distros”). The nature of Open Source allows any entity to create a distro. Distro are made to:

- Perform specialist tasks (eg. mail server, web server, system / networking utilities, Internet gateway, image / scene rendering);
- Be embedded operating systems in devices (eg. phones, PDAs, digital picture frames);
- Be viable desktop alternatives.

For the remainder of this report “Linux” refers to the operating system and applications, not just the kernel. Linux is developed with a modular design in mind, unlike the monolithic design approach taken by proprietary closed source companies in their operating systems.

A great deal of the software for Linux is released for bug and security fixes, and for feature / performance enhancements (improving on the previous versions). For example, real-world tests have showed a gain in performance from upgrading a Linux kernel from 2.4 to 2.6 [3]. The topics discussed in this report are equally applicable to either a server or a desktop installation of Linux.

A way to manage the installation and management of software on a Linux system is to use technology called “package management systems”.

### 2.2 OBJECTIVES OF THIS REPORT

The objective of this report is to highlight the advantages and disadvantages of the different package management systems used in Linux. This report will rate different package management systems against each other, and conclude on which systems are considered better.

### 2.3 INTENDED AUDIENCE

The intended audience for this report is the IT professional with a basic understanding of Linux systems and the concepts in Open Source software. It does not require a deep understanding of the software used or how to develop software.

## 2.4 ASSUMPTIONS

This report assumes that a Linux system has access to a network repository to install and upgrade its software, and not just from installation media.

## 2.5 LIMITATIONS OF THIS REPORT

This report is not the definitive guide to package management systems. Due to the dynamic nature of Open Source technologies and methodologies, any entity can “fork” the code of an application and come up with their own variation (or even write their own system “from scratch”). There are articles going in far greater detail than this report on the state of package management systems [4].

This report is generic enough to cover a wide range of situations where Linux is used, but it cannot cover all of them. Hence, the information presented here should be used as a guide in deciding which system is best based on each individual situation.

“Package management system” (or “package managers”) is a term to describe the tools behind how software is managed on a Linux system. Package managers are tied to one type of file format. A package management system will record applications installed on a Linux system, and ideally resolve any software dependencies.

A package management system’s main tasks are to install, upgrade, and remove software and underlying dependencies. Dependencies consist of supporting applications and libraries. Package managers should also handle complex tasks like if there is a need for two different versions of the same software to be installed, or if two different versions of software libraries are required on the same system.

At the end of the day, package management is supposed to ease the management of software.

### 3.1 PACKAGE MANAGEMENT EQUIVALENT IN MICROSOFT WINDOWS

In the Microsoft Windows environment package managers are referred to as “installers”. Most installations will consist of the user confirming a few values, and clicking “NEXT, NEXT, NEXT, and then OK”.

While Microsoft (and the third-party companies) attempt to ensure that installation of new software is smooth and simple, sometimes problems are introduced. Occasionally the Window’s registry will get corrupted leaving the system in an unstable state.

### 3.2 PACKAGE MANAGEMENT IN LINUX

In Linux, the main package managers utilise RPM, DEB, TGZ or Source (SRC) formats of files. A distro can be classified by which package management system it uses. For example, Red Hat (Enterprise) Linux is considered RPM-based. A greater list can be found on the Internet [5]. This is not where a distro has been derived from, as that is a different classification [6].

One issue (due to the open nature of Linux) is that the every distro has the choice of organising their file structure the way they see fit (ie. they can put files where they want). This causes a problem when attempting to use a package made for another distro than one that is made for the current installation of Linux.

#### 3.2.1 NOT UTILISING A PACKAGE MANAGER

Users are free to ignore the package management system on their Linux system and to install all their own software. This tends to lead to a system administration nightmare, as all applications have to be manually accounted for. This scenario should be considered the exception, as opposed to the rule.

#### 3.2.2 BUILDING YOUR OWN PACKAGES

Building your own packages is yet another choice users have in Linux. This is not a trivial task. Care is required to ensure that a “home-made” package (if installed on other

systems) does not orphan libraries into areas the system does not know where to find them, or overwrite already installed software and their libraries.

Criteria are grouped into a package manager **application** rating, and an **infrastructure** rating. An overall rating is calculated by adding the application rating to the infrastructure rating. Infrastructure ratings are common to a type of package format, whereas an application rating is specific to a package management tool.

#### 4.1 APPLICATION RATINGS

##### A. INSTALL / UPGRADE NEW SOFTWARE

Does the application install / upgrade software onto the system?

- 1 – Yes
- 0 – No

##### B. REMOVE SOFTWARE

Does the application remove existing software?

- 2 – Yes and removes dependencies
- 1 – Yes
- 0 – No

##### C. AUTOMATED DEPENDENCY RESOLUTION

Are there many interactions required by the user to resolve dependencies after installation / upgrade?

- 4 – No interaction required apart from the initial request for the software
- 3 – Very minor interaction is required
- 2 – User must resolve some dependencies manually
- 1 – User must do the majority of the resolution themselves
- 0 – User must do everything themselves

##### D. SPEED OF PACKAGE INSTALLATION

Does the application take a long time to install software?

- 1 – Fast
- 0 – Slow

##### E. OPTIMISATION

Does the application support optimised packages for each individual system?

- 1 – Yes
- 0 – No

##### F. SYSTEM WIDE UPGRADE

Will the application upgrade the entire system, or must a “fresh” re-install occur to be up with the latest version?

- 2 – Yes (fully upgrades system)
- 1 – Yes (updates system with fixes)
- 0 – No (re-installation must occur)

G. CORPORATE VENDOR SUPPORT

Is there a (paid-for) vendor supporting the package format – via e-mail or a call centre?

- 1 – Yes
- 0 – No

H. COMMUNITY SUPPORT

Is there communities of developers / users supporting the package format – via e-mail, newsgroups or chat-rooms?

- 1 – Yes
- 0 – No

**4.2 INFRASTRUCTURE RATINGS**

I. RELEASE TIME (CRITICAL FIX)

If the developers release a “fix”, how long after will the package maintainers release the corresponding “fix package”?

- 3 – Immediately after fix has been released
- 2 – A few days after fix has been released
- 1 – A week or more after fix has been released
- 0 – No fix released

J. RELEASE TIME (NON-CRITICAL FEATURE)

If the developers release a “features upgrade”, how long after will the package maintainers release the corresponding “upgrade package”?

- 3 – Immediately after upgrade has been released
- 2 – A few days after upgrade has been released
- 1 – A week or more after upgrade has been released
- 0 – No upgrade released until next version of system is released

K. MULTIPLE DISTRO SUPPORT

If the package manager’s format is shared across many distros, can the packages also be shared?

- 1 – Yes
- 0 – No

## 5.1 COMMENTS ON RPM

RPM files acts more like a snapshot. From most reports, eventually the system gets so unstable (from installing and upgrading applications) that the only course to bring stability and to allow for future installation of new applications is to re-install the operating system. Mandrakelinux have even suggested doing “fresh” installs over upgrading [7]. Apart from that admission by a (popular) distribution maker, there have some press over the (poor) state of RPM [8], and even some downright scathing attacks of the dependency issues (known as “RPM Dependency Hell” or just “RPM Hell”) brought about by using RPM [9].

The tools that are usually provided by the commercial RPM distributions only update a system with security / bug fixes. The user’s only option to get the newer applications is to upgrade their systems off new install media (ie. new purchases).

## 5.2 RATING RPM PACKAGE INFRASTRUCTRE

Criteria	Score
Release Time (Critical Fix)	2
Release Time (Non-critical Feature)	0
Multiple Distro Support	0
<b>Total</b>	<b>2</b>

## 5.3 PACKAGE MANAGEMENT APPLICATIONS USING RPM

## 5.3.1 RPM

The application used by RPM packages is called *rpm* (a deliberate distinction has been made in this report but this may not be so obvious in other literature). Red Hat Linux were the initial developers of the *rpm* tool, which forms the base tool of all RPM-based distros.

5.3.1.1 Rating *rpm* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	1
Automated Dependency Resolution	0
Speed of Package installation	1
Optimisation	0
System Wide Upgrade	1
Corporate Vendor Support	1
Community Support	1
Sub Total	6
+ score from package Infrastructure	2
<b>TOTAL</b>	<b>8</b>

### 5.3.2 APT4RPM

The *apt4rpm* application is a port from the package management system used by the DEB-based distros called *apt*. It is considered a community-developed tool, and is not officially supported by the commercial distribution makers that produce RPM-based Linux.

#### 5.3.2.1 Rating *apt4rpm* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	1
Automated Dependency Resolution	3
Speed of Package installation	1
Optimisation	0
System Wide Upgrade	1
Corporate Vendor Support	0
Community Support	1
Sub Total	8
+ score from package Infrastructure	2
<b>TOTAL</b>	<b>10</b>

### 5.3.3 UP2DATE

The *up2date* application from Red Hat (Enterprise) Linux is a wrapper for the *rpm* application. It will check with the official Red Hat repositories for the latest security / bug fixes, and advise the user what Red Hat deem as needed to upgrade the operating system. It does not appear to be a tool that can install new applications that did not previously reside on the system.

#### 5.3.3.1 Rating *up2date* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	1
Automated Dependency Resolution	3
Speed of Package installation	1
Optimisation	0
System Wide Upgrade	1
Corporate Vendor Support	1
Community Support	1
Sub Total	9
+ score from package Infrastructure	2
<b>TOTAL</b>	<b>11</b>

### 5.3.4 URPMI

Mandrakelinux developed the *urpmi* application, which acts as a wrapper around the *rpm* application. It can be made to look for applications and their dependencies from

CDs, local networks, or the Internet. Mandrakelinux wrap the *drakeconf* application as a GUI around *urpmi*.

#### 5.3.4.1 Rating *urpmi* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	1
Automated Dependency Resolution	3
Speed of Package installation	1
Optimisation	0
System Wide Upgrade	1
Corporate Vendor Support	1
Community Support	1
Sub Total	9
+ score from package Infrastructure	2
<b>TOTAL</b>	<b>11</b>

#### 5.3.5 YUM

The *yum* application is a wrapper for *rpm*. It is another community-developed application, but has been included in the recent Fedora Project's Linux (considered to be the beta testing ground for Red Hat (Enterprise) Linux).

#### 5.3.5.1 Rating *yum* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	1
Automated Dependency Resolution	3
Speed of Package installation	1
Optimisation	0
System Wide Upgrade	1
Corporate Vendor Support	0
Community Support	1
Sub Total	8
+ score from package Infrastructure	2
<b>TOTAL</b>	<b>10</b>

### 6.1 COMMENTS ON DEB

DEB is the package format used by the Debian distro, and Debian-based distros. DEB packages come from one of three main branches – “stable”, “testing”, or “unstable”. The official Debian project community (with over 1000 active volunteers maintaining over 13000 packages) maintains these branches [10]. The Debian community prides themselves on being the most stable, secure, and well-tested distro – even surpassing the commercially marketed distros.

Since most of commercial distros are RPM-based, third party vendors sell their proprietary products in RPM packages only. *alien* is used to convert RPM packages to DEB packages.

### 6.2 RATING DEB PACKAGE INFRASTRUCTRE

Criteria	Score
Release Time (Critical Fix)	2
Release Time (Non-critical Feature)	2
Multiple Distro Support	1
<b>Total</b>	<b>5</b>

### 6.3 PACKAGE MANAGEMENT APPLICATIONS USING DEB

#### 6.3.1 APT

In a DEB-based distro, *apt* is a package management system. It is not a tool itself, but is a suite of libraries used by several command line programs – most noticeably being *apt-get*. The methodologies and techniques used by *apt* appear to get ported to many other package management systems like *apt4rpm*, *slapt-get*, and *fink* (a port to Apple’s OS X). *apt* uses *dpkg* to do the actual installation and removal of packages. *apt* can have front-ends added on top of it like *aptitude* and *synaptic*.

#### 6.3.1.1 Rating *apt* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	1
Automated Dependency Resolution	4
Speed of Package installation	1
Optimisation	0
System Wide Upgrade	2
Corporate Vendor Support	0
Community Support	1
Sub Total	10
+ score from package Infrastructure	5
<b>TOTAL</b>	<b>15</b>

### 7.1 COMMENTS ON TGZ

The TGZ format has its basis in being a combination of an archive file (TAR file) that is then compressed (or GZIP'ed). The Slackware distro has used this format since its inception, and now is synonymous with the TGZ package format. TGZ is basically a lack of dependency resolution. The system administrator has to resolve dependencies manually, thus ensuring that only the minimum amount of packages is required for a Linux system (known as “reducing bloat”) [11].

The TGZ format follows the Slackware basic design philosophy of attempting to keep things simple [12]. The base tools for the system is the basic archiving and compression command line tools that can be found on a Linux system. *rpm2tgz* performs RPM to TGZ package conversion.

After installation users should then run the freshly installed application, and determine if it runs correctly, or if further TGZ packages are required to be installed.

### 7.2 RATING TGZ PACKAGE INFRASTRUCTRE

Criteria	Score
Release Time (Critical Fix)	2
Release Time (Non-critical Feature)	2
Multiple Distro Support	1
<b>Total</b>	<b>5</b>

### 7.3 PACKAGE MANAGEMENT APPLICATIONS USING TGZ

#### 7.3.1 PKGTOOL (WITH INSTALLPKG, REMOVEPKG, AND UPGRADEPKG)

One level above *tar* (used to uncompress TGZ packages) is *installpkg*, *removepkg*, and *upgradepkg* and the front end to these command ling tools is *pkgtool*. *pkgtool* is similar to *aptitude*, providing a text based menu driven application. Most users new to TGZ format usually find *pkgtool* easier to manipulate packages for new applications, but some will eventually move down to the command line tools (ie. *installpkg*, *removepkg*, *upgradepkg*, and sometimes *tar*), as these offer greater control and flexibility over what is being done when managing packages.

#### 7.3.1.1 Rating *pkgtool* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	1
Automated Dependency Resolution	0
Speed of Package installation	1
Optimisation	0
System Wide Upgrade	2
Corporate Vendor Support	1
Community Support	1

Criteria	Score
Sub Total	7
+ score from package Infrastructure	5
<b>TOTAL</b>	<b>12</b>

### 7.3.2 SWARET

The *swaret* application is a command line tool that checks upon a repository to see which dependency applications and libraries are required. It can be set up to use a local repository or to use on from the Internet. The makers of Slackware have included it with their distribution (the only other tools being *tar* and *pkgtool*), but have not installed it by default.

#### 7.3.2.1 Rating *swaret* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	1
Automated Dependency Resolution	3
Speed of Package installation	1
Optimisation	0
System Wide Upgrade	2
Corporate Vendor Support	1
Community Support	1
Sub Total	10
+ score from package Infrastructure	5
<b>TOTAL</b>	<b>15</b>

### 7.3.3 SLACKUPDATE

*SlackUpdate* appears to be the equivalent of Red Hat's *up2date*. It appears to synchronise itself with the official Slackware repositories of TGZ applications and find out which packages are required for updating. It does not appear to be able to install packages that did not previously exist on the system, but once a TGZ package is installed, *SlackUpdate* can then be used to obtain the latest up-to-date version of it.

#### 7.3.3.1 Rating *SlackUpdate* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	1
Automated Dependency Resolution	3
Speed of Package installation	1
Optimisation	0
System Wide Upgrade	1
Corporate Vendor Support	1
Community Support	1
Sub Total	9

Criteria	Score
+ score from package Infrastructure	5
<b>TOTAL</b>	<b>14</b>

### 7.3.4 SLAPT-GET

*slapt-get* is another application is used for managing TGZ packages on a Linux system. Unlike the basic package management tools for a TGZ-based distro, it will attempt to resolve dependencies. The *slapt-get* application mimics *apt* by retrieving dependencies, and then passes the rest of the work to *installpkg*, *upgradepkg*, and *removepkg*.

#### 7.3.4.1 Rating *slapt-get* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	1
Automated Dependency Resolution	4
Speed of Package installation	1
Optimisation	0
System Wide Upgrade	2
Corporate Vendor Support	0
Community Support	1
Sub Total	10
+ score from package Infrastructure	5
<b>TOTAL</b>	<b>15</b>

### 8.1 COMMENTS ON SOURCE (SRC)

A Source-based (SRC-based) distro is one of the hardest distros to categorise, as all Linux distros have the potential of being SRC-based. For the purposes of this report, SRC-based distros are one that install mainly install from source code compressed files. Source code has to be compiled to a binary format allowing the Linux system to use it. All other package formats mentioned in this report have been compressed binary files.

With the improvements in modern hardware, a source-based distro offers much more freedom and flexibility than any binary-based system [13]. An example is optimisations during compilation, a SRC-based Linux system is considered faster than binary-based distros [14].

A typical SRC package will contain the compressed source code from the developers. Some SRC packages also contain some minor patches to the developer’s code, allowing for some customising from the distro makers. Benefits can be gained from SRC-based that cannot be matched by binary-based systems. Source code auditing, custom source code patching, binary optimisation, “sandbox” build environments are just some of the features available.

Every SRC-based distro does things its own particular way, hence why there will be an individual infrastructure rating for each SRC package manager.

### 8.2 PACKAGE MANAGEMENT APPLICATIONS USING SRC

#### 8.2.1 PORTAGE

The *Portage* system is from attempting to automate and optimise as much as possible [15]. *Portage* also has the capability of installing binary packages (including RPM, DEB, and TGZ packages after some transformation has occurred to them), although these are limited in number usually to third party closed source applications (eg. nVidia drivers, and Sun Java SDK / JRE).

*Portage* has one command line tool that does most of the work of installing, updating, and removing software. *Porthole* is a front-end GUI application for the *Portage* system.

##### 8.2.1.1 Rating Portage Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	2
Automated Dependency Resolution	4
Speed of Package installation	0
Optimisation	1
System Wide Upgrade	2
Corporate Vendor Support	0
Community Support	1

Criteria	Score
Sub Total for package management Application	11
Release Time (Critical Fix)	3
Release Time (Non-critical Feature)	3
Multiple Distro Support	1
Sub Total for package Infrastructure	7
+ score from package management Application	11
<b>TOTAL</b>	<b>18</b>

## 8.2.2 SORCERY

The *sorcery* system has a separate command line tool for installing, upgrading, and removing applications, and synchronising its application repository. In other ways it is similar to *Portage*.

One advantage *sorcery* does have over *Portage* is that it is designed to update the local source code repository only with what is required. In the long term, it is estimated that a system using *sorcery* downloads less when upgrading software than even a binary system [16].

The only disadvantage of *sorcery* is that it does not have as many users as the *Portage* system has. Therefore, there is a smaller resource pool to draw from if problems occur, or if a package update is required.

### 8.2.2.1 Rating *sorcery* Package Management Application

Criteria	Score
Install / Upgrade new software	1
Remove Software	2
Automated Dependency Resolution	3
Speed of Package installation	0
Optimisation	1
System Wide Upgrade	2
Corporate Vendor Support	0
Community Support	1
Sub Total for package management Application	10
Release Time (Critical Fix)	2
Release Time (Non-critical Feature)	2
Multiple Distro Support	1
Sub Total for package Infrastructure	5
+ score from package management Application	10
<b>TOTAL</b>	<b>15</b>

---

## 9 SUMMARY

---

The following is a summary of the ratings of the applications observed in this report.

<b>Tool</b>	<b>Overall Rating</b>
Portage	18
apt	15
slapt-get	15
sorcery	15
swaret	15
SlackUpdate	14
pkgtool	12
up2date	11
urpmi	11
apt4rpm	10
yum	10
rpm	8

When using any computer system, the main goal of any user is to get their task done, as unobtrusively as possible. A user should not have to worry about having to resolve dependencies if all they want to do is to set up a particular application. Too often than not, package management systems hinder the user from completing their tasks.

This report has made observations on 12 package management systems used in the modern Linux distro. Each of those systems comes with its own capabilities, tools, and front-ends.

When choosing a distro to use, it should be based on the features it provides. Most users will agree that a fast system, that does not require constant re-installation as benefits. Long-term solutions call for an approach that will allow for high uptimes, especially when Linux forms the platform for mission critical applications. Upgrading of a system while it's still performing would be a boon.

There is no single “best choice” for every situation, although the *Portage* tool (from the viewpoint of this report) does offer the largest amount of features. Flexibility, optimisation, availability of source code, and automated dependency resolution are all to be found, with the only main disadvantage being the time taken to establish the base Linux system. The *srcery* system would exceed the *Portage* system in this report if it were not for a smaller pool of developers and users.

Following on from that would be the Debian developed *apt* tool. This tool would be the better choice when it comes to all the binary-based package systems. The tools offered by the TGZ-based systems follow this.

RPM-based systems may have been the starting point for many people using Linux; they appear not to be the place where they stay. With issues in the “almost proprietary like” format of the packages built, and others to do with non-automated dependency resolution, they would be most suited for a system that never needs to be updated.

Lastly, this report should also illustrate that commercially marketed package management systems in the world of Open Source, are not always the sole or even the best solutions available.

---

## APPENDIX A: REFERENCES

---

These web pages listed below correspond to references in this report. At the time of writing, all the links were verified to be valid.

- [1] **Initial announcement of Linux**  
<http://www.shortfamilyonline.com/tech/unix/history-of-linux/reference/25-Aug-1991-what-would-you-like-to-see-most-in-minix.html>
- [2] **Forrester Finds Linux Enterprise Ready Today**  
<http://channelzone.ziffdavis.com/article2/0,1759,1586510,00.asp>
- [3] **Kernel comparison: Web serving on 2.4 and 2.6**  
<http://www-106.ibm.com/developerworks/linux/library/l-web26/>
- [4] **A four part (somewhat technical) suite of articles on the existing approaches to software installation**  
<http://www.linuxworld.com/story/32661.htm>  
<http://www.linuxworld.com/story/32664.htm>  
<http://www.linuxworld.com/story/32667.htm>  
<http://www.linuxworld.com/story/32669.htm>
- [5] **What is your distribution's package management?**  
<http://www.distrowatch.com/stats.php?section=packagemanagement>
- [6] **How independent is your distribution?**  
<http://www.distrowatch.com/stats.php?section=independence>
- [7] **Mandrake Linux (RPM-based) with their justification to do a fresh install over an upgrade**  
<http://www.mandrakeuser.org/docs/install/iupdate.html>
- [8] **An article by the owner of Distrowatch.com as an observation of the state of RPM technologies**  
<http://www.distrowatch.com/dwres.php?resource=article-rpm>
- [9] **A “rant” about RPM hell, but it does have some excellent examples of installations / upgrades that have gone wrong**  
[http://www.germane-software.com/~ser/Files/Essays/RPM\\_Hell.html](http://www.germane-software.com/~ser/Files/Essays/RPM_Hell.html)
- [10] **Debian Project organisation**  
[http://en.wikipedia.org/wiki/Debian#Project\\_organization](http://en.wikipedia.org/wiki/Debian#Project_organization)
- [11] **An observation on Slackware 9.0**  
<http://linux.editme.com/slackware>
- [12] **Introduction to Slackware and simplicity**  
<http://www.slackfiles.net/documentation/en/books/slackware-basics/html/introduction.html#AEN39>

- [13] **The Emergence of Source Based GNU Linux Distributions**  
<http://sorcerer.wox.org/docs/distro/distro1.html>
  
- [14] **Two separate articles on Source-based distro (Gentoo) vs. Binary-based distro (Mandrakelinux)**  
<http://www.unicom.com/chrome/a/000432.html>  
<http://www.gentoo.org/main/en/performance.xml>
  
- [15] **Reason behind *Portage***  
[http://www.gentoo.org/main/en/philosophy.xml#doc\\_chap2](http://www.gentoo.org/main/en/philosophy.xml#doc_chap2)
  
- [16] **Long term source-based distro usage**  
<http://sorcerer.wox.org/docs/distro/distro2.html>