Package Managment

Many users decide to use Linux based Operating Systems because of the freedoms they enjoy, included in the list is the freedom to choose the right tool for the job. The Slackware package management schema is a model for displaying the choices that are available to the users. When maintaining a server or even a personal system a certain amount of control must be given to the administrator, the control and simplicity of Slackware package management is what may in fact make you want to try or continue to use Slackware.

Brief Explanation of Slackware Packages

In terms of security and accountability Slackware is excellent because it brings you software in the condition that the authors intended. What this means is that the Slackware community does not modify the source or add and remove functions from software beyond the author's design, unless a small patch is required to work on Slackware. All packages are built using the original source code in which only official patches are applied, the only modifications that are made for the packages are adding .desktop files (executables for windows managers) and running installed applications/scripts to update databases, font or icon caches to include the newly installed files.

Other distributions such as debian and redhat choose to modify the original source code with their own revisions prior to distribution, this can add additional functionality but in some cases such as the debian OpenSSL packages it can result in exploits that only effect the specific distribution which can remain unseen for some time because the original authors (those who know the applications best) were not responsible for the changes or aware of the changes. This additional liability is something that an administrator does not want hanging over their head when running a production system which requires heightened security and uptime, with Slackware these issues cannot happen as all code that is used has been developed by those who know the functionality of the applications best, namely the original authors.

Slackware packages can be found with any of the following extensions:

- tbz - Slackware package archive compressed using bzip2
- tlz - Slackware package archive compressed using lzip
- tgz - Slackware package archive compressed using gzip
- txz - Slackware package archive compressed using xz

The reason for the various package formats is the evolution of compression tools, as better tools have been developed Slackware has adapted their packages to take advantage of the higher compression tools.

Currently the two most commonly used formats are tgz and txz. txz is the current standard for official packages because of the increased compression rate, this change has allowed the development team to reduce the filesize in installation packages which in turn reduced the space requirements on the installation DVD, leaving additional space for new applications or documentation to be added in the same space as before. The tgz format has a less impressive compression rate but it is sufficient for packages that you build on your system such as slackbuilds, since on most modern systems disk space is abundant.

Dependency Tracking

One of the major complaints by new users is the fact that Slackware does not automatically track dependencies and install dependencies when you install a file. To many this may seem like a negative mark against Slackware, but many users prefer to know exactly what is installed on their systems and what it is for. The lack of dependency tracking allows the system administrators to install only the dependencies that are required for their necessary functionality without introducing unneeded components.

Why restrict Automatic Dependency Resolution?

- Lack of administrative control
- It prompts for installation of packages that may have a bad security history, without manual research you would not be able to verify the security history of an installed package
- It may install an application that conflicts with current installed applications, which can break software components in your system.
- If you may decide to recompile an application and remove certain functions, the removal of said functions may eliminate the need to have specific dependencies installed but with an auto-resolution system it will force those packages even though they are not needed.

The official and many of the third-party package management tools in Slackware try to keep away from automatic dependency resolution so the administrator is in full control of the system.

Installation and Package Management Tools

There are many package management tools for Slackware,both official and from third-parties, both groups are discussed below.

Official Tools

Included in the base installation of Slackware are two packages **pkgtools** and **slackpkg**.

pkgtools contains a set of applications that are to be used for basic package management.

slackpkg is a package manager and update tool for you to use with the official Slackware servers.

pkgtools

pkgtools contains the following executable applications

refer to the man pages on your system for more detailed information including options and syntax.

- **installpkg** - This application is used to install a new package.
- **removepkg** -This application is used to remove a package from your system.
- **upgradepkg** - upgradepkg upgrades a Slackware package from an older version to a newer one. It does this by isntalling the new package onto the system, and then removing any files from the old package that aren't in the new package *(taken from the man pages)*.
- **explodepkg** - This tool uncompresses and untars a Slackware package into your current directory so the contents can be reviewed prior to installation.
- **pkgtool** - pkgtool is a menu-driven package maintenance tool provided with the Slackware Linux distribution. It allows the user to install, remove, or view software packages through an interactive system. Pkgtool can also be used to re-run the menu-driven scripts normally executed at the end of a Slackware installation. This is useful for doing basic reconfiguration (like changing the mouse type). *(taken from the man pages)*.
- **makepkg** - This application is used to create a new Slackware package from the contents of your current directory.

slackpkg

slackpkg is to be used to install official slackware packages from the official Slackware servers and to manage updates and upgrades. This tools is very useful for keeping a system up-to-date and for doing a distribution version upgrade without having to do a complete reinstall or having to download and burn a disk.

When attempting a version upgrade with slackpkg **read the file UPGRADE.TXT** from the installation disk to determine instala order and avoid breaking your system.

Unofficial Tools

The unofficial tools are tools that were made by third parties which are not officially supported by Slackware. These tools can be broken into two groups package retrieval and package building.

The package retrieval tools have a functionality which is similar to YAST, yum, apt-get, urpmi and various tools which are used in other distributions. These package retrieval tools call to specific online package repositories which have been set by the user and allow the user to search the repositories to download and install pre-configured packages.

In contrast the package building tools are somewhat similar to the package retrieval tools but rather than downloading a pre-build package they retrieve a build script *(which can be customized)* and the subsequent source code allowing the user to custom compile an application for their system and their needs. Once the application has been compiled, the scripts pass the proper instructions to utilize the resulting binaries and build a Slackware formatted package that properly interacts with the Slackware package management tools. The best comparison for package building tools is to compare them to the portage system that is used in Gentoo Linux.

Listed below are the Unofficial tools and a brief explanation of their functionality:

- **swaret** - This tool is designed to replicate the functionality of the Debian apt-get system, you identify the repositories you would like to use and it allows you to download and install

packages from the specified location, this tool also attempts dependency resolution.
- **slapt-get** - This tool is designed to replicate the functionality of the Debian apt-get system, you identify the repositories you would like to use and it allows you to download and install packages from the specified location(s).
- **sbopkg** - This tool syncs with the slackbuilds.org build script repository, it is used to pull build scripts, upon choosing the scripts to build it allows you to choose the build order and also customize the scripts. Upon completion of the scripts it will allow you to build a Slackware package or to build and install the package.
- **slpkg** is a powerful software package manager that installs, updates, and removes packages on Slackware based systems. It automatically computes dependencies and figures out what things should occur to install packages.

Installation Methods

With of the quality of various tools, you have options for various functionality when it comes to package management and installations. The main three methods of installation are listed below:

- Install from a pre-built package
- Compile the application yourself from the source code, This is ill advised in it's standard meaning, I will discuss this later.
- Build a custom package from a build/slackbuild script

Install from a Pre-built Package

To install Slackware from a pre-built package you can use **installpkg**, **upgradepkg**, **swaret** or **slapt-get**. This is very simple, in **swaret** or **slapt-get** you only need to choose the package and select to install it. To install a package using the official tools you only need to call the program and use the package name as the argument, the example below is installing the application wine from the current workin directory.

example:

```
#installpkg wine-2.5.6-x86.tgz
```

Compile the Application from the Source Code

Many people have grown accustomed to this method over time, but it is not preferred in Slackware, instead, to document your actions and avoid replication of efforts it is advisable to build a slackbuild script, which will save you time and effort in the future.

Build a Custom Package from a Build/slackbuild Script

This is the true magic in Slackware, every official package that you receive has a slackbuild file, in the source directory on the Slackware disk you will find the slackbuild files and source code for every officially supported package. This availability in the source directory, allows you to view all options/modifications that were done to a package and if you need you can modify the script and

arguments to fit your needs. Once the script has been run and the package has been built you then have a redistributable package for the specified application that fits your specific needs.

The slackbuild.org script repository houses user submitted scripts for building stable applications for Slackware that have been tested on various architectures by knowledgeable members of the Slackware community. Included in the slackbuild archive are the build script, license information, .desktop files (if needed), icons, and a .info file that tell you the version, source download location, md5sum, supported architectures and the author of the script. There is also a mailing list and an irc channel available if you have questions or experience problems.

Review slackbuild_scripts for an example of how to build a package from a slackbuild, and how to build custom scripts.

Package Tracking

When using Slackware, you have the ability to track the installation date, installed files, and the package description for all packages that are installed on your system. This tracking is done within the /var/log/packages directory. This directory contains a single text file for each installed application, the file contains the package description and list of installed files/directories from the current package. You can easily view the list of files within the */var/log/packages* directory and sort by name, creation date or whatever additional criteria you use.

For some examples of the usability of the */var/log/packages* directory, you can review the following:

If you are looking to see what package installed a specific file to your computer, let say the file is */sbin/iptables*, you can run the command

```
grep /sbin/iptables /var/log/packages/*
```

This will display all files within the package directory that contain the exact phrase that was called by grep.

If you want to review the installed files for a package to find all files that were installed in the */bin* directory, let say we are looking at the pkgtools package, you can run the command

```
grep /var/log/package/pkgtools-13.0-noarch-3.txt "/bin/"
```

In a similar fashion Slackware moves the text files from */var/log/packages* to */var/log/removed-packages* when you remove a package from your system, this makes it easier to have adequate tracking of what you have installed and what you removed and when each action was taken, this will help you greatly if you are wanting to replicate the list of installed packages from one system to another, you can easily write a script to compare the output of the folders on both systems and output a resulting file of non-matching packages that you can later use in a script for automatic installation.

More commands and references can be found inthe related articles listed below.

Related Package Managment Articles

Page	Description	Tags
Package Managment	Package Managment Many users decide to use Linux based Operating Systems because of the freedoms they enjoy, included in the list is the freedom to choose the right tool for the job. The Slackware package management schema is a model for displaying the choices that are available to the users. When maintaining a server or even a personal system a certain amount of control must be given to the administrator, the control and simplicity of Slackware package management is what may in fact make you w...	slackware, package management, author mfillpot
Unofficial Package Repositories	Unofficial Package Repositories * slackbuilds.org - This site holds community submitted builds scripts to easily build Slackware packages from the source code of various applications. * slackware.it - This is a repository of pre-built packages compiled in various languages.	slackware, package management, package repos, author mfillpot

Related Package Managment How-Tos

Page	Description	Tags
Building A Package	Building A Package This is a rough outline for building Slackware packages. Some steps may not be neccessary, some steps might be missing. Use the discussion page for side-notes such as using slacktrack (when DESTDIR fails) and other utilities like checkinstall.	howtos, software, makepkg, package management, author slackwiki
Building and Installing Packages with sbopkg	Building and Installing Packages with sbopkg Sbopkg is a command-line and dialog-based tool to synchronize with the SlackBuilds.org (“SBo”) repository, a collection of third-party SlackBuild scripts to build Slackware packages. The program has a curses based interface which lets you pick and assemble the programs which you want to compile from source into packages. It can also be used non-interactively in case you know beforehand what your goal is - in that case it is	howtos, software, sbo, package management, author ldkraemer
Installing Software	Installing Software Overview There are three basic ways to install software in Slackware: install a pre-build binary package, make your own package, or compile the sources and install the resulting binaries manually. If you need to compile a program from source code, making it into your own package is easy and useful. Here's how these methods work.	howtos, software, installing, package management, author peterwwillis
Querying Installed Packages	Querying Installed Packages Sometimes you might want to check whether a particular package is installed or which version of a package is installed on your system. If the package is part of the Slackware installation you could use the slackpkg tool:	howtos, software, package management, package tracking, author sycamorex

Sources

* Original source: * Linux.com

* Originally written by mfillpot for linux.com, reposted with permission from Linux.com * Contrbutions

by mfillpot

slackware, package management, author mfillpot