Getting to know Slackware packaging tools

There are two types of package tools; menu based tools and command line based tools. There is actually only one menu based tool and that's "pkgtool" all the rest are command line tools.

You use pkgtool to get an overview of what's installed on your system. It provides menu options to view installed packages, the content of individual packages and you can also remove currently installed packages by using the menus and you can install new ones. If you use pkgtool to install packages, you can specify a directory containing one or more packages and you will be asked a 'yes' or 'no' question about if you want them installed. If you select to remove packages you will get a list of all currently installed packages and you can then select one or more to remove.

All of the above functions can also be accomplished by using the command line tools. If, for example, you just need to install a single package, it may seem a bit tedious to have to go through a lot of menus. That's why we have the command line tools. Here's a description of each of them and how I mostly use them.

installpkg

Description:

installpkg is used to install a prebuilt slackware package. Basically what it does is to unzip and untar the package in the "/" (root) directory of your filesystem, and subsequently it executes the script "install/doinst.sh" from the package if it's included (try to unpack a package from your slackware CD into a new directory and take a look at it).i

How I use installpkg:

In almost all cases all I do is type a command like "installpkg packagename.tgz" (or "installpkg packagename.tar.gz"). In a few cases I may wish to review the content of a package before I install it, so I issue a command like "installpkg -warn packagename.tgz > package.log". That gives me a file called package.log with information about what files will be installed and where; and I can now judge if I wish to install the package (if you omit the "> package.log" part the report will be dumped to your console). I rarely use the "-r" an "-m" options, they can be used to generate and/or install a subdirectory as a package, but I prefer to use makepkg for that.

[Notes on Installing more than one package. Let's say you have the kde1 dir and you want to install everything,
you can cd into the kde1 dir and "installpkg *.tgz". Or you can "installpkg pack1.tgz pack2.tgz pack3.tgz"]

explodepkg

Description:

explodepkg is used to extract a package into the current directory without running the "install/doinst.sh" script from the package and without updating the installed-packages database in "/var/adm/packages". [note that /var/adm is a sym link to /var/log, so /var/log/packages is also correct] It's a useful tool if you are maintaining/updating a package (probably one of your own) and want to change a few things. After using explodepkg and editing the package it is a simple matter to run makepkg to recreate the package with the new and/or updated content.

How I use explodepkg:

There is only one way to use this tool; cd to an empty directory and type "explodepkg packagename.tgz".

removepkg

Description:

removepkg is used to remove a currently installed Slackware package. It will look in "/var/adm/packages" for information about what files to remove (the entry under /var/adm/packages was created when the package was installed).

How I use removepkg:

Normally I just type`removepkg packagename`, but if I don't have the original package on disk or CD-ROM or if I'm not completely sure if the package is critical to normal systems operation I'll pass the "-preserve" option to removepkg (as in "removepkg -preserve packagename") to keep the entire tree of removed files under "/tmp/preserved_packages/packagename"). This allows me to reinstall the files if the system did indeed need them for something. Just as with installpkg you can pass the "-warn" option to get a dump of the files that would be removed by the operation. Read the man page for a description of the "-keep" and "-copy" options if you think you'll need them (I don't).

makepkg

Description:

This is the tool you'll use to create a proper Slackware package. What it does is to tar and gzip the content of the current working directory and it's subdirs while converting symlinks to script code on the fly (the script code will be placed in a file called "doinst.sh" in a dir called install) so as to be able to recreate links at package installation.

How I use makepkg:

Let me give you an example. Let's say we want to create a package called "readme.tgz" that contains 3 files called "readme1", "readme2" and "readme3".
When the package installs you want the files placed like this; "/usr/readme1", "/usr/sbin/readme2" and "/home/readme3". To do that you will need to create a new empty dir to create the package in, in this example I'll assume you create a new dir called "/home/pkgmanager/readme" and change to that directory ("cd

/home/pkgmanager/readme"). You can now start building your package. First we create the needed directories, execute these commands:,`$ mkdir usr $ mkdir usr/sbin $ mkdir home` Then copy the files to their right locations, like this:`$ cp ~/readme1 usr $ cp ~/readme2 usr/sbin $ cp ~/readme3 home` Now we need to build the package itself, execute the command "makepkg readme.tgz" to do just that. If there where any symlinks encountered (not in this example) you will find that an "install" directory has been created with the "doinst.sh" script to recreate them. This all goes into the package and if you feel you have some extra commands that should be run from doinst.sh you can explodepkg the package and add them to the script. You should now have a proper slackware package called "readme.tgz" that will install properly with "installpkg readme.tgz". Try it out and see if you can find the readme1, 2 and 3 files in their right locations. If you ever need to create a package of a program that you downloaded as source (or binary) you will have to create the correct directory structure for all the files in the package and follow the above procedure (remember to compile the source first ;-). If the program uses autoconf/automake, then life is a great deal simpler as you can usually just pass the "--prefix=" parameter to the configure script to have "make install" install it into some subdirectory of your choice. Then you can run makepkg in that directory afterwards and that's it. Here is a generic example of creating a package from a source based on autoconf/automake:`$ ./configure --prefix=/home/pkgmanager/builddir $ make $ make install $ cd /home/pkgmanager/builddir $ makepkg packagename.tgz`

upgradepkg

Description:

This tool is used to upgrade an installed Slackware package with a new version. All it does is to install the new package on top of the old one and then remove all files from the old package that are not present in the new one.

How I use upgradepkg: There are not that many ways to use it. Here's how it goes; If the old and new packages have the same name, the just type "upgradepkg packagename.tgz", if the names are different, then use a command like "upgradepkg oldpackage%newpackage.tgz". [You can also upgrade multiple packages with upgradepkg by doing "upgradepkg *.tgz", as long as the packages you want to upgrade have the same name as the Slackware packs in the dir.]

Other notes:

Both installpkg, removepkg and upgradepkg support an env. variable called "ROOT" that points to the root directory to use when installing or removing packages. This is not something you'll generally use, but it can be useful if you need to install packages to another harddrive/partition or stuff like that.

There are two other types of scripts that you can place in a Slackware package apart from "doinst.sh" (the configuration and only-once scripts) but most people don't need to use them. Read the man page for makepkg if you feel you need to know. Remember to use ash script syntax in your package scripts if they are to be used by anyone but yourself, as that's the shell that will be used if the packages are installed from a Slackware rescue disk (and not all people have bash installed ;-)

Jesper Juhl

jj@chaosbits.net

http://www.chaosbits.net/Articles/Historical/SlackwareTools/

Articles

Most of the articles listed here were moved from Userlocal. Some (well most) are a little out of date But I've put them here on the off chance that someone might still use them.

Installation

- Slackware-12.0 on Raid1
- Slackware-8.1 with raid0 /

Security

- Securing Slackware using exec-shield

Configuration

- LAMP on Slackwarept.2pt.3
- Slackware Printing Faq (2002)

Other

- Getting to know pkgtools

Slackware People

- Amrit
- Alan Hicks
- AlienBob
- Alphageek
- Fred Emmott
- Mozes

Slackware People

- Robby Workman
- PiterPunk
- Vincent Batts
- Follow @mrgoblin
- person

- person

Slackware Sites

- The Slackbook
- SlackBuilds.org
- LinuxQuestions Slackware Forum
- Sligdo
- ARMedslack
- Slack/390

Slackware tools

- Slackpkg
- sbopkg
- src2pkg
- slacktrack
- Slackpack