Software Installation and Management in Slackware

by Robby Workman
http://slackware.com/~rworkman/

Presented at:
Auburn University

February 25, 2011

Brief History of Slackware

- Derived from SoftLandingSystems (SLS) — Slackware is probably the first "fork" distribution.
- Volkerding initially had only some fixes for various bugs in SLS, but for whatever reason, Peter MacDonald (SLS maintainer) didn't accept the patches, so Patrick made them available on his university's ftp server for others.
- Over time, the patch collection grew, and that coupled with other factors led Patrick to release the cumulative effort as Slackware 1.0 on July 16th, 1993. Shortly thereafter, this new "Slackware" received a rewritten installer, and a new tool called "dialog" that you might have heard mentioned a few times (and which Patrick helped write) appeared in Slackware 1.1.0.

Slackware Philosophy

- Slackware is still very much a "traditional" linux distribution — this is expected due to its reputation as the most UNIX-like distribution of linux.
- Stability and ease of use are Slackware's primary goals
  - Attempt to ship unmodified upstream sources
  - Not bound to a predefined release schedule — it will be released when it's ready, but we try to do a new release at least once per year
  - Transparent configuration — well-commented configuration files
  - Each application is configured independently like the upstream developers intend — Slackware does not have a "global" configuration file or tool
  - No Slackware-specific "hidden" configuration — documentation provided by the upstream developers should be correct and complete

Slackware Philosophy (cont'd)

- Conservative development model — well-tested and functional software is not quickly replaced by newer, untested, and possibly less stable software without good reason

This does not mean that Slackware ships OLD software!

Slackware is generally just as "up to date" (if not **more** up to date) as any other distribution with the software that it includes

- Slackware is not intended to be a Windows clone or work-alike — there are some other linux distributions handling that quite well
- Slackware provides a system that is intuitive and easy to use for an experienced Unix administrator while also being relatively easy to learn for a new Unix user

Current Development Team

Patrick Volkerding — volkerdi@
Eric Hameleers — alien@
Piter PUNK — piterpunk@
Robby Workman — rworkman@
Stuart Winter — mozes@
Vincent Batts — vbatts@
Alan Hicks — alan@
Erik Jan Tromp — alphageek@
Fred Emmott — fred@
Mark Post — markkp@
Amritpal Bath — amrit@
Karl Magnus Kolstø — karlmag@
Leopold Midha — netrixtardis@
John Jenkins — mrgoblin@

Other Contributors

There are several others who wish to remain anonymous due to sensitive employment positions or otherwise...

Slackware Package Management

- Slackware **does** have a package manager.
  - `pkgtool(8)`, `installpkg(8)`, `upgradepkg(8)`, and `removepkg(8)` manage **packages** just fine.
- Slackware's native package management tools, however, do **NOT** attempt to manage package **dependencies** at all.
  - Packages are not split into *app-bin*, *app-lib*, *app-devel*, *app-doc*, and so on — every file that would normally be installed by `make install` will be present in Slackware's package.
  - Disk space is not as big of a concern today as it was in the past, so a full installation is generally recommended.
  - In a full installation, all dependencies needed by Slackware's native packages are provided.

Slackware Package Management (cont'd)

You might want to obtain information about Slackware packages that are already installed on your system.

Use `ls(1)` to list installed packages

```
# ls /var/log/packages
a2ps-4.13b-i386-2
aaa_base-12.1.0-noarch-2
aaa_elflibs-12.1.1-i486-1
...
```

Use `cat(1)` to show the contents of an individual package

```
# cat /var/log/packages/a2ps-4.13b-i386-2
PACKAGE NAME: a2ps-4.13b-i386-2
COMPRESSED PACKAGE SIZE: 859 K
UNCOMPRESSED PACKAGE SIZE: 3140 K
...
usr/share/psutils/md68_0.ps
usr/share/psutils/md71_0.ps
```

How Slackware's Package Management Utilities Work

- Slackware packages are basically just compressed tar archives. They are extracted to the "/" directory unless an alternate root is specified.
- If the environment variable "`ROOT`" is set, then the package management utilities will act as if its value is the real "/".
- You will notice that all of the files in the example package are listed with relative paths — this is so that they will be placed in the correct location when the package is extracted.
- Once the files in the package are extracted, then `doinst.sh` (the postinstall script) is executed from the `$ROOT` directory (which is usually "/".

Sample Package Contents

FILELIST

```
./
etc/
etc/hejaz.conf.new
usr/
usr/bin/
usr/bin/makehejaz
usr/doc/
usr/doc/makehejaz-1.0/
usr/doc/makehejaz-1.0/README
usr/doc/makehejaz-1.0/COPYING
usr/include/
usr/include/hejaz.h
usr/lib
usr/lib/libhejaz.so.1.0
usr/man
usr/man/man1/
usr/man/man1/makehejaz.1.gz
install/
install/doinst.sh
install/slack-desc
```

makehejaz doinst.sh contents

cat install/doinst.sh

```
 config() {
  NEW="$1"
  OLD="$(dirname $NEW)/$(basename $NEW .new)"
  # If there's no config file by that name, move it over:
  if [ !  -r $OLD ]; then
    mv $NEW $OLD
  elif [ "$(cat $OLD | md5sum)" = "$(cat $NEW | md5sum)" ]; then
  # toss the redundant copy
    rm $NEW
  fi
# Otherwise, we leave the new copy for the admin to consider...
}

config etc/hejaz.conf.new

( cd usr/lib ; rm -rf libhejaz.so.1 )
( cd usr/lib ; ln -s libhejaz.so.1.0 libhejaz.so.1 )
( cd usr/lib ; rm -rf libhejaz.so )
( cd usr/lib ; ln -s libhejaz.so.1 libhejaz.so )
```

Config File Handling in the Package

Note that the configuration file "`etc/hejaz.conf`" was actually installed with a "`.new`" extension. A proper Slackware package should almost always do this with config files, init scripts, and other such files that might be customized by the system administrator.

- This prevents package upgrades from overwriting config files already installed on the system.
- In the sample package listed above, the `config()` function in the `doinst.sh` file checks to see if the *hejaz.conf* file is present at `$ROOT/etc/hejaz.conf`.
  - If it does not exist, the *hejaz.conf.new* file installed by the package is moved over to *hejaz.conf*.
  - If it does exist, then the md5sum of it is compared to the new one.
    - If the md5sums match, then the two files are identical, so the new one is deleted.
    - If the md5sums do not match, the new file is left (with the `.new` extension) for the system administrator to check.

Symlink Handling in the Package

You may have noticed that the sample package contained a single library file — `libhejaz.so.1.0` — but apparently no files or symlinks named `libhejaz.so.1` or `libhejaz.so` pointing to it.

This is because Slackware's `makepkg(8)` utility removes the symlinks when creating the package — it then records them in the *doinst.sh* file so that they are created when the package is installed.

This does NOT mean that you should manually add symlink creation to a `doinst.sh` file!

Create any needed symlinks so that they actually exist in your temporary package installation directory (such as `$DESTDIR`), and then let `makepkg(8)` handle this — it's less error-prone than we are...

Installing Additional Software in Slackware

Slackware's official package set is adequate for many users, but there are quite a few additional pieces of software that are needed in many cases. The traditional "`configure && make && make install`" works, but it comes with potential problems...

- Possible conflicts with packaged software
- Difficult to track what is installed
- Possible problems with upgrading and/or removing
- Experienced users can manage these just fine, but others will eventually have problems in most cases

There are a few well-known third party package repositories...

- Many users do not want to install packages from third party sources at all.
- For those that do install packages from third party sources, it is important to find **trusted** sources.

Installing Additional Software (cont'd)

Trusted sources for packages — where are they and how can you know if they are trustworthy?

- Is the packager someone with a good reputation in the community?
- Check the package contents
  - Are the files in the correct locations?
  - Are config files properly installed?
  - Are man pages compressed?
  - Does the package include documentation (if applicable)?
  - Are the packages built on a "clean" Slackware installation?
  - Are all package dependencies documented?
- Does the packager provide complete sources and documentation of how the package was built (such as a build script)?

Sources are usually required to be hosted per the software's license!

In our opinion, packagers who do not provide sources and build methods should generally be avoided.

Trustworthy Third Party Package Repositories

- A few of the Slackware team members provide packages (and sources) for various applications and libraries
  - Eric Hameleers (alienBOB) — `http://slackware.com/~alien/`
  - Robby Workman (rworkman) — `http://rlworkman.net/pkgs/`
  - Erik Jan Tromp (alphageek) — `http://alphageek.dyndns.org/`
  - Piter PUNK — `http://piterpunk.info02.com.br/extra/`

This is not intended to be an all-inclusive list!

There are certainly other places to obtain unofficial Slackware packages. Package quality may vary — some are known for not providing sources, not providing build instructions, and/or not building on clean installations (which causes undocumented dependencies on other packages).

Using SlackBuild Scripts

Eventually, you will need some application for which either:

- no package exists, or
- you do not want to install one of the available packages

When that happens, you may be able to find a SlackBuild script for the application instead.

A SlackBuild script is essentially just a small shell script

- extracts the source code tarball
- prepares the source code for compilation (sets compile options)
- compiles the object code
- installs it to a temporary "staging" directory
- performs other needed operations (install documentation, compress man pages, add package description, etcetera)
- uses `makepkg(8)` to make a Slackware package of the application

SlackBuilds.org

This section will be covered by referring to the `http://slackbuilds.org` site.

Other resources

- Slackware Home Page — `http://slackware.com/`
- Official Slackware Book — `http://slackbook.org/`
- Unofficial Slackware Wiki — `http://slackwiki.org/`
- SlackBuilds.org Project — `http://slackbuilds.org/`
- Chess Griffin's sbopkg — `http://code.google.com/p/sbopkg/`

Credits and Acknowledgments

- Thanks to Jan Engelhardt for his tremendous contribution to my LaTex knowledge and giving me a "push" toward using it.
- Thanks to the Pat and the rest of the Slackware team for all of the fixes, suggestions, and general proofreading of this presentation, but especially to Pat for providing **much** historical perspective.
- Thanks to the SlackBuilds.org team for all the time and effort you guys spend making sure it's successful.

About the author

Robby Workman is a high school Physical Science teacher and lives in a rural area outside Tuscaloosa, Alabama, USA with his wife and two two daughters. He has been a Linux and Slackware user since July 2004, and a member of the Slackware development team since January 2007. He also is a founding member and current admin of the SlackBuilds.org project, which was created in July of 2006.

Feedback and suggestions are welcome at any of the following addresses:

`rworkman@slackware.com`
`rworkman@slackbuilds.org`
`rw@rlworkman.net`