Sitewide Unix Software Installation

Russ Allbery

March 20, 2004

Introduction

- Manage 550 packages for seven architectures (alpha_dux40, hp_ux110, i386_linux24, rs_aix43, sgi_65, sun4x_58, sun4x_59).
- Both free software (most of it) and licensed software, including a licensed software tree for only one Unix cluster.
- Three trees: pubsw for campus-wide software, newsw for software testing, and sweet for licensed software for one Unix cluster.
- Better Linux distributions making some of this obsolete. Expect to significantly decrease the amount of software that we support in the near future in favor of using software packaged by distributions.
- Something like this needed for some time for Solaris and licensed software.
- Method originally developed by Larry Schwimmer.

Contents

- Basic layout concepts and @sys
- Volume naming and release management
- Compiling the software
- Installing the software
- Handling software with its own layout
- Other interesting problems

Basic Layout

- How @sys works and why it's important
- Central area organized by software package
- @sys and share directories inside each package directory
- Organized to match the standard file system layout, in each software package directory
- Symbolic link trees for each platform into package
- package an additional mount point for central area
- Single /usr/pubsw to /afs/ir/systems/@sys/pubsw link on each client

Naming and Release Management

- Everything must have a version number
- Every package gets its own volume, no matter how small
- Packages are upgraded just by changing the symlinks, making it easy to back out of an upgrade
- Use `frak` to check for changes before release
- Can add a new minor platform by copying the link tree

Freezing a Platform

- Why you would want to do a freeze
- Copying the entire software pool
- Copying the link trees (not actually necessary)
- Can keep the frozen platform around as long as you want

Compiling

- Software compiled to look in /usr/pubsw for everything
- Shared vs. static libraries and library versioning
- Making sure shared libraries are found (LD_RUN_PATH)
- Be careful to compile on the oldest system
- Automating the compilation process (`wrap`)
- Special case: Perl and site module directories

Installing

- Two regular choices: make prefix= vs. make DESTDIR=
- libtool and library or binary relinking
- @sys directory vs. share directory
- Testing out of the read/write volumes
- Testing using the separate newsw tree (and when you can’t)
- ACLs: public, site-licensed, only some machines?
- Be careful of inodes and shared libraries
- share vs. lib and how much you want to fiddle

Packages with a Special Layout

- The basic idea of /usr/pubsw/apps
- Examples: Oracle, most licensed software, Java JDK
- Still create symlinks for particular binaries
- Alternative: use wrapper scripts
- It’s possible to edit paths inside binaries

Other Problems

- When you might need to change the @sys value for a platform
- Displaying information about installed packages
- Reverting changes to the same version (`frak`)
- man page indexes and info dir files
- X libraries: don’t do this yourself
- Where you want to avoid doing this, and the future of Linux, BSD ports, and other alternatives