Practical Debian Administration

Alexander Zangerl
Bond University
az@{debian.org,bond.edu.au,}
What is this all about?

- Things to get you going with your new Debian box
- Useful Gadgets (IMHO)

Overall: How to deal with a Debian system efficiently
Who am I?

- Professional Bugbear for students at Bond Uni, QLD
- Debian Developer, one of >1000 volunteers
- Sysadmin by choice
What is Debian?

“The Universal Operating System”

- a bunch of people with a common goal
- a mindset, expressed by some rules and policies
- an OS software distribution

“The Distribution for Sysadmins by Sysadmins”
Getting Started

You’re in a maze of twisty shell prompts …

- FS Layout: follows the Filesystem Hierarchy Standard
- config in /etc/, variable stuff in /var,...
- default web root in /var/www/
- PIDfiles in /var/run/
- /usr/local untouched but checked first in $PATH
Docs?! What Docs?

All in /usr/share/doc/X/:

- debian: FAQ, constitution, general docs
- debian-policy: the nuts-and-bolts documents
- newbie-doc: Debian for Dummies?
- apt-howto(-en): Guide to apt
- every package: must have copyright and changelog.Debian, often also README.Debian

Most docs gzipped: zmore wrapper is always installed
Tools Initially Encountered

**base-config**: handles last stages of initial install
- e.g. root pwd setup, `apt` setup, …
- can be rerun safely at need.
- `/usr/lib/base-config` provides code snippets, run via `runparts`

**tasksel**: coarse package selection tool
- only for installing big package groups ("tasks", eg. C Development)
- `tasksel -t`: shows `apt-get` call but doesn’t install.

**dselect**: avoid. There Be Dragons^WConflicts.


- low-level package management tool suite
- implements the core functionality
- no awareness of multiple sources for packages
  - deals only with files available locally
- detects dependency conflicts and omissions
  - but only flags errors, has no mechanisms for automatic resolution
- safety net can be overridden selectively
- main tool `dpkg`, variety of helpers
**dpkg** **Common Usage**

- `dpkg -l` show brief package status
  - "ii" is good, "rc" is removed, "pn" is gone.
- `dpkg -s name` show package control and status
- `dpkg -i pkgfile.deb` install package from `pkgfile.deb`
- `dpkg -r name` remove package
- `dpkg -P name` remove package and config files
- `dpkg -L name` list files belonging to package
- `dpkg -S file` find package owning file
What Next? Cleanup Time!

- make sure base install is not too fat for you
- list, review, remove, repeat. (repent?)
  - `dpkg -l` to find packages installed
  - `dpkg -p name` to see the description
  - `dpkg -P name` to remove the package
- *Do Not* use any of *dpkg*'s `force` options here!
apt

- works on top of dpkg
- adds high-level logic for resolving package relations
- handling of complex situations and upgrades
  - but all choices and safety features overrideable
- ability to retrieve packages if necessary
- knowledge of distribution streams
- knows about different package sources and access methods
- keeps cache of packages in /var/cache/apt/
  - config: /etc/apt/apt.conf
  - sources: /etc/apt/sources.list
  - streams: /etc/apt/preferences
apt Tools

- Two main tools
  - **apt-get**: the Debian “Swiss Army Knife”
  - **apt-cache**: cache manipulation, querying and searching
apt-cache Common Usage

- `apt-cache search phrases` find packages with matching descriptions
- `apt-cache show name` show the control header for a package
- `apt-cache policy name` show information about available versions of a package (e.g., upgradability, package source)
apt-get Common Usage

- `apt-get install name` installs package and dependencies
- `apt-get remove name` removes package (but not config)
- `apt-get update` updates the package cache
- `apt-get upgrade` installs newest versions of everything
  - as far as possible without conflicts,
  - won’t install any new packages,
  - won’t remove any installed packages
- `apt-get dist-upgrade` like upgrade, but schedules important packages first
apt-get in Action

# apt-get install snoopy
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
   ld.so.preload-manager
The following NEW packages will be installed:
   ld.so.preload-manager snoopy
0 packages upgraded, 2 newly installed, 0 to remove
   and 9 not upgraded.
Need to get 11.9kB/11.9kB of archives. After unpacking
   147kB will be used.
Do you want to continue? [Y/n]
...
Selecting previously deselected package ld.so.preload-manager.
Selecting previously deselected package snoopy.
...
Setting up ld.so.preload-manager (0.3.3-1) ...
Setting up snoopy (1.3-3) ...
...
Helpers and Tips

- **vrms**: must-have tool for zealots
- **apt-setup**: interactive user-friendly tool to create sources.list entries
  - do include [http://security.debian.org/](http://security.debian.org/)
- **apt-show-versions**: tells you about packages with updates available, outdated versions etc.
- **apt-get clean**: cleans out the package cache
- **set Apt::Get::Show-Upgraded "true"**: makes apt show packages before upgrading
- **apt-listchanges**: small tool that shows changelog entries before package installation
- **apt-listbugs**: queries the Bug Tracking System for serious bugs before installation
Example changelog

openssh (1:3.3p1-0.0woody2) testing-security;
  urgency=high

* NMU by the security team.
* Fix rsal key creation (Closes: #150949)
* don’t fail if sshd user removal fails
* depends: on adduser (Closes: #150907)

-- Michael Stone <mstone@debian.org>
    Tue, 25 Jun 2002 08:59:50 -0400
Getting Rid of Cruft

some helpers for making removal of packages easier:

deborphan: finds unneeded packages
  • looks for packages depending on library packages
  • if none: flags the unrequired library package
  • can be told to work on all packages, too.

debfoster: install only wanted packages
  • keeps track of packages you want installed
  • checks packages that are required as dependencies
  • if dependencies change, flags now removable packages
debconf

- all packages come preconfigured or with configuration script
- **debconf** provides unified configuration management
  - simple front-end independent config scripts
  - multiple front-ends: dialog, editor, noninteractive, ...
  - multiple backends to store past answers: flat file, LDAP, pipes, even stacked backends
- aids multiple deployments
  - configure once
  - make answers available to client boxes
  - run installation on clients with non-interactive frontend
- data stored in `/var/cache/debconf/config.dat`
- to rerun config phase: `dpkg-reconfigure pkgname`
Customising the Run-Time Env

- alternatives: for commands that come in multiple flavours
  - symlink-based setup, maps functional name onto actual program
  - eg. `/usr/bin/vi` → `/etc/alternatives/vi` → `/usr/bin/nvi`
  - admin tool is `update-alternatives`
- System-V init:
  - `update-rc.d`: manages symlinks and default runlevels
  - initscripts should not have config embedded
  - such data goes into `/etc/default/X`
  - most important example: `/etc/default/rcS` for boot process
Other Helpers

- lots of `update-something` tools
- most important:

  **update-modules**: kernel modules configured via files in `/etc/modutils`, combined into `modules.conf` by this script.

  **update-inetd**: deals with adding, disabling, removing of services, also reloads `inetd`

  **update-mime**: controls which programs are chosen to handle MIME objects, eg. image viewers. sibling `run-mailcap` is used to run handler.
Fooling the Package Management

• `dpkg-divert`: override files from packages persistently
  • to rename or remove a file that belongs to a package
  • eg. unwanted plugins that can’t be disabled otherwise
  • or replacing a program with your own version (at the same location)

• `dpkg-statoverride`: override permissions persistently
  • except for SUID stuff *rarely* necessary or a good idea...
Fooling the Package Management II

• Putting a package on hold: makes `dpkg` not touch it, ever
  • may be necessary for custom package with conflicting versions
  • or if newer versions are known to be buggy
• `dpkg-get-selections` and `dpkg-set-selections`
  • `equivs`: *really* fool the package mgmt
    • produces a dummy package with dependency information only
    • to tell the package mgmt about externally built software
• also available: various `force` options for `dpkg` and `apt`
Stability or Bleeding Edge?

- multiple distribution streams, mixable to a certain extent:

  **unstable:** what the developers work on
  - changes daily
  - occasional breakage due to being a moving target.

  **testing:** candidate for the next release
  - packages meeting certain criteria auto-migrate from unstable
  - is eventually frozen and becomes next release

  **stable:** the released distro
  - no updates except security fixes (often backported)
  - testing process is lengthy, releases infrequent
  - software is rock-solid but often outdated
Mix and Match

1. include other streams in sources.list
   • check [http://apt-get.org](http://apt-get.org)
   • backports: [http://www.backports.org](http://www.backports.org)

2. set source priorities in /etc/apt/preferences
   • see `man apt_preferences` and `apt-howto` (better)

3. install packages "as usual"
   • problem: library dependencies set at build-time
   • on the building system → generally refer to unstable
Example preferences file

Package: *
Pin: release a=stable
Pin-Priority: 990

Package: *
Pin: release a=testing
Pin-Priority: 980

Package: *
Pin: release a=unstable
Pin-Priority: 970

Package: *
Pin: origin backports.org
Pin-Priority: 400
Building Your Own Stuff

- most common: custom kernel
  1. get the sources, stock or from package `kernel-source-version`
  2. get `kernel-package`
  3. configure your kernel "the normal way"
     - eg. `make menuconfig`
  4. `make-kpkg kernel_image`
  5. install the resulting Debian package with `dpkg`

- goodies:
  - boot loader autoconfig
  - modules are taken care of
Questions?

- Feel free to ask now!
- ...or later: az@{debian.org,bond.edu.au}
- you can find the paper and these slides at
  http://people.debian.org/~az/vic-2004/