Debian and (large scale) System Administration

Alexander Zangerl

Bond University

az@{debian.org,bond.edu.au}

Last Minute Yuck

- If you find that the paper in the proceedings looks ugly...
- ...then please blame Word and not me!
- The *real* paper and these slides are available here: `http://people.debian.org/~az/sage-2003/`

What to expect of this next hour

ideally: *“Debian in a Nutshell”*

- An Introduction to Debian
  - History, Goals, Philosophy, . . .
- Key Characteristics
  - the Glossy Brochure
- What an Admin needs to know
  - Package Management
  - Larger Scale Environments
  - Daily Niceties
- Stretching the Envelope
  - Beating the OS into submission

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"

- {therefore | nevertheless} one of the most popular distros

Distributions?

- There is no one single ‘Linux OS’.
- Linux kernel != full OS
  - collection of libraries, basic programs, tools, support files needed
  - $\rightarrow$ “distribution”
  - earliest: Jim Winstead Jr’s Boot & Root Disks
  - larger systems following in Spring 92: MCC, SLS, TAMU...
  - but none very friendly or manageable

Why another Distribution?

- because early distros were virtually unmaintainable
  - software badly integrated
  - lousy upgrade paths
  - little automation
  - no package management

Package management?

Source coming in `tar`balls is quite ok, *but*:

- configure + compile takes time, rinse, lather, repeat...
- error-prone to setup all stuff yourself
- keeping track of new stuff is a lot of work
- keeping track of your changes is ugly
- duplication of efforts

A package manager

- keeps track of installed packges
- can install, upgrade and remove packages
- can verify and maintain system integrity

Enter Debian!

- project started 1993 by Ian Murdock
- key reason: distributions crucial for Linux's future
- but not enough high-quality distributions a/v
  - example: SLS
- making a distro is nasty, hard and non-glamorous work
  - integration of lots of 3rd party software
  - consistency hard to achieve
  - lot of ongoing work necessary to keep up to date and bug-free
- tackling *this* is main focus of the Debian Project

Debian: The People

- non-commercial, open volunteer organisation
- started as small group of Free Software hackers
- grew globally, ’95: 60, ’98: 400, ’02: 900+ developers
- almost no hierarchy:
  - lots of developers = project members
  - one elected project leader
  - plus some special task teams
  - powers and procedures: set in the Debian Constitution
- basic consensus:
  - Debian Social Contract
  - Debian Free Software Guidelines

Organisational Foundation

- Debian Constitution
  - describes org structure and decision making process
  - developers, elected leader, tech committee
- Debian Social Contract
  - commitments Debian makes towards the world
  - includes guarantee to remain 100% free software
- Debian Free Software Guidelines
  - sets the rules for what is acceptably free software
  - only software *fully* meeting this can be part of Debian!
  - basis of the Open Source Definition
- $\rightarrow$ a reasonable guarantee of continuity to Debian user
- “hostile takeover” of Debian pretty much impossible

Technical Foundation

- Debian Policy Manual
  - technical policies for system and distro behaviour
    - e.g. Filesystem Hierarchy Standard,
    - consistent UIDs, GIDs and ownerships,
    - SysV init conventions, syslog usage, . . .
  - how to package software
  - how packages have to interact
  - how, what and where to document things
  - library packaging, naming and versioning
  - what a user can rely on on a Debian system
    - related effort: Linux Standards Base
    - important for commercial development

Result: The Distribution

- lots of packages, currently ~8900
- (plus non-free stuff outside the main distribution)
- uses the Linux kernel
  - but GNU hurd, Free/NetBSD support underway
- 11 architectures supported:
  Intel 386, ARM, 68k, SPARC, Alpha, MIPS, PA-RISC, S/390, Super-H, IA-64, Powerpc
- software is released for all architectures at the same time

Some Distribution History

- before 1996: some pre-1.0 releases
- 1996: 1.1 “Buzz”, 2.0 kernel, 474 packages
- 1998: 2.0 “Hamm”, 1500 packages, i386, m68k
- 1999 2.1 “Slink”, 2000 2.2 “Potato”
- 2002: 3.0 “Woody”, 8900 packages, 11 architectures
- What about those names?
  - characters from movie “Toy Story”
  - unstable distro: “Sid”

Key Characteristics for System Admin

This may sound like a sales pitch, but it's all true ;-)

- YOU are in charge, always!
- Debian maintained by its users
- Multi-platform Availability
- Continuity and Consistency
- Remote Maintenance
- Choice between Stability or Bleeding Edge
- High Quality, Security
- Installation flexible and fairly simple
- Automation well-supported
- Lots of software packaged
- Constant Evolution and Development

The admin is in charge!

- no “padded cell” environment
- flexibility is paramount
- shooting yourself in your foot is possible
- everything is well (if sometimes concisely) documented
- configuration is mostly flat-file-based
- automation-friendly environment
- user-friendly frontends do exist
  - but you are never forced to use these
  - manual config changes *guaranteed* to be preserved!

Debian is maintained by its users

- Debian is used to make Debian
  - network of colocated servers all over the world
  - run auto-builders,
  - provide archive mirrors,
  - access for developers,
  - and of course “boring” things like mail, web, cvs, . . .
- Debian also run by developers
  - and gazillions of users contributing bug reports and suggestions

Multi-platform Availability

- provides Linux environment for most architectures
- currently 11 architectures supported
- ranging from ARM via the usual to PA-RISC and to S/390
- OS is consistent for all those platforms
  - except a few things like bootmanagers
- software is kept in sync between architectures
  - often major porting efforts necessary
  - i386'isms, big vs. little endian, 32 bit vs. 64 bit, . . .
- updates released at the same time
  - auto-builder system produces binary packages for all architectures automatically

Consistency

- fine-grained dependencies between packages
  - allows installing less-than-everything without loss of function
  - (mostly) automatically dealt with by package mgr
- cooperation between packages ensured by policy
- packages must come with sensible defaults or have to ask
  - but you are encouraged to adjust things to your needs
  - ...and your modifications will survive!

Continuity

- upgrades are seamless
  - can be made whenever software becomes available
  - when and to the extent *you* want
  - no need to wait for major releases
  - upgrades no longer drastic “make or break” tasks
- all software upgradable in place
  - no reboot or single-user mode (except for kernel upgrades)
  - minimal service downtime
- package mgmt never overwrites your configuration
  - worst case: being prompted with a `diff` between files

Remote Maintenance

- server far away withouth physical access? no problem!
- all administration work can be done remotely
- this includes upgrades
  - eg. upgrading `sshd` while logged in via `ssh`
  - or fiddling with `libc` and `ld.so` while services are running

Stability or Bleeding Edge?

- multiple *mixable* distribution streams:

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 occasionally outdated

Quality and Security

- maintainers have strong personal interest in quality
  - reputation in an all-volunteer group
  - maintainer can be a group of persons, too
  - mentoring for newcomers available
  - many upstream authors become Debian developers
- Debian has a dedicated Security Team
  - ...but everybody does contribute (reports, “NMU”s, . . .)
  - team follows relevant channels and sources,
  - produce security fixes (usually only for stable),
  - often backport fixes to old software versions in stable,
  - release advisories
  - auto-builders produce updated packages rapidly
- Debian values testing and integration over releasing often

What an Admin needs to know

- Package management & tools
- Keeping Things Up to Date

Package Relationships

Depends, Pre-Depends: absolute dependency. Can be versioned.

Recommends, Suggests: less stringent relationship. Combination “a good idea” according to maintainer.

Conflicts: general incompatibility or same functionality/files provided.

Replaces: package replaces file from old package. Used for cleanly renaming or outphasing packages.

Provides: declares “virtual packages” which implement the same core functionality in different flavours.

Example Package `control` Header

```
Package: kuvert
Priority: extra
Section: utils
Installed-Size: 128
Maintainer: Alexander Zangerl <az@debian.org>
Architecture: i386
Version: 1.1.7
Depends: libc6 (>= 2.3.1-1), gnupg (>= 1.0.6) | pgp,
  sendmail | mail-transport-agent, perl,
  libmailtools-perl, libmime-perl (>= 5.212),
  libterm-readkey-perl
Recommends: gnupg (>= 1.0.6)
Suggests: quintuple-agent, pgp
Size: 30506
Description: A wrapper that encrypts or
 signs outgoing mail
 kuvert automatically signs and/or encrypts
 outgoing mail using the PGP/MIME standard (rfc3156),
....
```

dpkg

- 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
- `dpkg -s` *name* show package `control` and status
- `dpkg -L` *name* list files belonging to package
- `dpkg -S` *file* find package owning file
- `dpkg -i` *pkgfile.deb* install package
- `dpkg -r` *name* remove package
- `dpkg -P` *name* remove package and config files

Package Status

`dpkg` keeps track of

installation status: the current situation

- `not-installed`, `config-file`
- `installed`
- `half-installed`, `unpacked`, `half-configured`

selection status: the intended state

- `install`
- `deinstall`, `purge`

flags: extra tags

- `hold`
- `reinst-required`

dselect

- old-style frontend for `dpkg`
- extends `dpkg`
  - supports multiple package sources
  - conflict resolution
  - interprets `Recommends:` and `Suggests:`
  - offers package browsing
- unintuitive interface
- mostly replaced by `apt` and its frontends

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
  - source config: `/etc/apt/sources.list`
  - stream config: `/etc/apt/preferences`
  - keeps cache of sources and packages offered

`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 (eg. 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-mar
Selecting previously deselected package snoopy.
...
Setting up ld.so.preload-manager (0.3.3-1) ...
Setting up snoopy (1.3-3) ...
...
```

Virtual Package Example

```
# apt-get install mail-reader
Reading Package Lists... Done
Building Dependency Tree... Done
Package mail-reader is a virtual package
provided by:
  communicator-smotif-475-libc5 4.75-2
  communicator-smotif-475 4.75-2
  mutt 1.3.28-2.2
  kmail 4:2.2.2-14.6
  balsa 1.2.4-2.2
  ...
  emacs20 20.7-13.1
  chaos 1.13.1-6
  af 2.0-6
You should explicitly select one to install.
```

Beyond `apt`

- `apt` is a command line tool
- not very beginner-friendly
- new frontends are being developed:
  - `aptitude`
  - `deity`
  - `gnome-apt`, `kpackage`, ...

Keeping Things Up to Date

- basically just a periodic run of `apt-get update` and `apt-get upgrade`
  - `apt` figures out what can and may be upgraded
  - `dpkg` does the package replacements
  - `dpkg` handles simple configuration situations, prompts with `diff`s between old and new version
  - “Maintainer Scripts” take care of more complex migrations
- Your configuration changes are guaranteed to be migrated between versions, nothing is ever lost.

Larger Scale Environments

- various systems for unattended installation a/v
  - `fai`, `systemimager`, `replicator`, ...
- strict policies and automation help with central admin
  - eg. read-only `nfs` root filesystem is not too hard,
  - configuration always in `/etc/`
  - tools like `cfengine` tie in very well
- installation can be from most media
  - including `http`, `ftp` or `nfs`
  - Debian packages mirrored all over the world
- support for building local mirrors...
  - `apt-proxy`, `apt-move`, `debmirror`
- ...or personalized filesystems/images
  - `debootstrap`, `rootstrap`, `jigdo`

Automation

- lots of `update-`*something* scripts
  - building config files from fragments in directories
  - no need to parse and `sed` config files
- `run-parts` used almost everywhere!
  - idea from SysV `init`
  - look at scripts in directory, run them in sequence as per filename prefix
- permission-fiddling with devices: rarely needed
  - pre-setup for access by appropriate groups
  - `adduser` can quickly add users to groups
- SysV `init` with `update-rc.d` (non-interactive) frontend

debconf

- all packages come preconfigured or with configuration script
- but ad-hoc scripts interfacing with user problematic
  - no history of answers, not consistent for the whole distro
- `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

Other Niceties

some other nice features/tools I find useful:

- `apt-listchanges`: shows changelog information before package is installed
- `apt-listbugs`: queries the Bug Tracking System and displays warning before installing package with serious bugs
- `jablicator`: produces meta-package depending on currently installed packages
  - allows fast installation of your preferred packages on another box
- `dpkg-repack`: re-packages installed software into a Debian package
  - includes also configuration files of the package
  - allows fast backout of upgrade

Example changelog

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

 * NMU by the security team.
 * Fix rsa1 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
```

Stretching the Envelope

- no one configuration fits everybody
  - Debian supports you in adjusting things *efficiently*
- various mechanisms to customize at run-time
- custom compilation is also supported well
- ...and weird things like `auto-apt`:
  - monitors file accesses and automatically installs referenced packages

Run-time adjustments

- 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`
- `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)

Fooling the Package Management

- `dpkg-statoverride`: override permissions persistently
  - except for `SUID` stuff *rarely* a good idea...
- 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 `-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`

Custom compile-time configuration

- all of Debian is available in source form
  - upstream and Debian-specific data in separate files
- packages must be automatically buildable from this source
  - needed for auto-builders to work
  - sources come with `Build-Depends` information,
  - and build process is policy-regulated
- locally adjusting and building a package is really simple:
  - `apt-get source` *name* to get the sources
  - `apt-get build-dep` *name* to install the pre-requisites
  - maybe adjust Makefile `debian/rules`
  - run `debian/rules binary` to build a `.deb` package
- or use `apt-build` for a more streamlined process

Some Pointers

Project Home: `http://www.debian.org`

G. Williams' book:

`http://edm.act.cmis.csiro.au/debian/book`

My paper and these slides:

`http://people.debian.org/~az/sage-2003/`

Questions?

- Feel free to ask now!
- ...or later: `az@{debian.org,bond.edu.au}`