Debian Kernel Roundup

http//www.vergenet.net/linux/debian_kernel/

Simon Horman aka Horms

horms@valinux.co.jp, horms@debian.org, horms@verge.net.au

January 2006

Debian_Mini-Conf@Linux.Conf.Au

The Debian Kernel Team

Loose collection of Debian Developers and non-Debian Developers

Wiki: `http://wiki.debian.org/DebianKernel`

Handbook: `http://kernel-handbook.alioth.debian.org/`

Mail: `debian-kernel@lists.debian.org`

IRC: `irc.oftc.net`, `#debian-kernel`

Source: `svn://svn.debian.org/kernel/`

Old Sarge-Style Packaging

Each kernel version has its own source package: 2.4.27, 2.6.8, ...

Each architecture has its own image package: powerpc, i386, ...

Image packages have to be manualy built and uploaded

Esoteric dependancy system to allow source package to move forward and still give the image packages the source they expect

Ok, for development

Horrible for stable or security updates

Unified Packaging: Overview

Each major kernel version has its own source package

All packages are built from this source package.

All other architectures, including amd64 are

Massively streamlined kernel maintenance

As of 2.6.12 in Etch

Unified Packaging: Advantages

For architectures that closely follow upstream, which is most of them, there isn't much need for per-architecure package maintainers any more

Time previously spend on per-architecture packaging issues can now be focused on unified packaging issues

Reduces duplication of effort

Reduces divergance in packaging

Allows uptream to be tracked more closely

Unified Packaging: Disadvantages

For architectures that do not follow upstream, there is a lot of work to keep up with packaes that are tracking upstream: m68k

Some trouble reaching consensus on how to handle sub architectures, multi-arch, or whatever: mips, mipsel

Unified Packaging: Tracking Upstream

Tracking upstream is important because of the stable-development model that upstream adopted for 2.6

Reduces the number of patches that Debian needs to carry

Reduces divergance between upstream and Debian

Allows Debian's Bug Reporting Army (users) to do QA on the kernels that upstream care about

Which in turn greatly increases the number of parties interested in resolving bugs

Unified Packaging: Tracking Upstream (cont)

2.6.14 and 2.6.15 — the two most recent releases as of writing — were released into Debian the same day as upstream made the release

Achived by staging -rc relases in experimental

Which also provides valuable feedback on bugs that have already been fixed in -rc releases

Etch

What about Etch?

Etch: 2.6

At some point, stop putting new upstream versions into sid, and work on stabalisation.

Thats about it, nice and simple

Etch: 2.4

IMHO, legacy for Etch

But there seems to be enough need to warrant including it

Which leaves us looking down the barrel of supporting 2.4 into the next decade

Etch: 2.4: Plan A

Plan A:

Make a linux-2.4 package using the unified packaging and 2.4.32

2.4 is frozen upstream, so tracking it from there should be trivial

But interested in 2.4 might be smaller than the amount of effort required

Etch: 2.4: Plan B

Plan B:

Minimal updates to 2.4.27 which is being maintained for Sarge

Sarge

Ok, what about Sarge?

Sarge: Security Updates

Both 2.4.27 and 2.6.8 are being actively maintained for security fixes

“sarge1” updates were finally released in December

Dialogue between the security and kernel teams is slowly increasing

“sarge2” updates will hopefully happen in the next month of so

Sarge: Patch Tracking

Security and other stable patches are tracked in the patch-tracking/ directory in svn.

`http://svn.debian.org/wsvn/kernel/patch-tracking/`

One file per problem

Usually tracked by CVE

Allows us to track which bug effect and are fixed in which packages

Allows tracking which bugs need a CVE

Allows tracking which bugs aren't upstream yet (or which ones are missing from 2.4)

Coordinates information and

Tries to stop things from slipping through the cracks

Simple and evolving

Sarge: 2.6.12 Backport

Very minimal backport of the then sid 2.6.12 package to Sarge

Quantum leap in upstream version from Sarge's 2.6.8

Because of the high rate of upstream development backporting is very difficult

Recently updated to 2.6.15

Now available on `Backports.Org` allong with `initramfs-tools` and `yaird`

Woody

And Woody?

Woody: Security Updates

Dan Frazier and Martin Shulze are slowly working on updates

http://wiki.debian.org/DebianKernelWoodyUpdateStatus

Progress is being made

Though I do not know when they will be released

Fun Playing with Kernel Packages

Information from the Kernel Handbook

Package Types: Architecture Independant

- linux-source-*X.Y.Z*: The kernel patched source
  e.g: `linux-source-2.6.15`
- linux-doc-*X.Y.Z*: Kernel documentation
  e.g: `linux-doc-2.6.15`
- linux-patch-debian-*X.Y.Z*: Debian patches
  e.g: `linux-patch-2.6.15`
- linux-tree-*X.Y.Z*: Dummy package for build depenancies
  e.g: `linux-tree-2.6.15`

Architecture Dependant Packages: I

- linux-image-*FLAVOUR*:
  Virtual package depending on the latest (recommended) kernel image for a flavour
  e.g: `linux-image-686`
- linux-image-*X.Y-FLAVOUR*:
  Virtual package depending on the latest (recommended) kernel image for a major version and flavour
  e.g: `linux-image-2.6-686`
- linux-image-*X.Y.Z-N-FLAVOUR*:
  Kernel image and modules
  This is the kernel image used to boot
  e.g: `linux-image-2.6.15-1-686`

Architecture Dependant Packages: II

- kernel-image-*X.Y-FLAVOUR*:
  Transitional packages that depend on the linux-image equivalent
  e.g: `kernel-image-2.6.15-1-686`

Architecture Dependant Packages: III

- linux-headers-*X.Y.Z-N*:
  Common headers for an architecture (or sub-architecture)
  e.g: `linux-headers-2.6.15-1`
- linux-headers-*X.Y.Z-N-FLAVOUR*:
  Flavour specific headers for a flavour
  Mostly symlinks to the common headers
  e.g: `linux-headers-2.6.15-1-686`
- linux-header-*X.Y-FLAVOUR*:
  Virtual package depending on the latest (recommended)
  kernel headers for a major version and flavour
  e.g: `linux-image-2.6-686`

Obtaining the Source

```
# apt-get install linux-source-2.6.15
$ tar jxf /usr/src/linux-source-2.6.15.tar.bz2
$ cd linux-source-2.6.15
```

`make-kpkg` can be used to build binary kernel packages from here

```
# apt-get install kernel-package build-essential
$ cp /boot/config-2.6.15-1-686-smp .config
$ make menuconfig
$ make-kpkg --append-to-version hls --initrd --us --uc kernel\_image
```

Obtaining the Patches

You only need this if you want to examine the patches

Or patch *down* the source

```
# apt-get install linux-patch-debian-2.6.15
```

In `linux-source-2.6.15`, patch down to `2.6.15-1`

```
$ /usr/src/kernel-patches/all/2.6.15/apply/debian 2.6.15-1
```

Patch back up to `2.6.15-2`

```
$ /usr/src/kernel-patches/all/2.6.15/apply/debian 2.6.15-2
```

Making "Official" Kernel Packages

Download the source and build dependancies

```
# apt-get source linux-2.6
# apt-get install build-essential fakeroot
# apt-get build-dep linux-2.6
```

... and build

```
$ cd linux-2.6-2.6.15
$ debuild -us -uc
```

Series Files

Patches are based on a series file, with the following syntax

```
# Comment
+ patch-to-apply.patch
- patch-to-remove.patch
```

They are processed in order when patches are applied

And reverse order when patches are removed

Thus patches can be reversed in a subsequent series

While still keeping the original debian kernel version, in tact

Patching "Official" Kernel Packages: I

First decide on a debian version
e.g: `2.6.15-2` → `2.6.15-2hls`

Next, create a series file in `debian/patches/series`

It should be named after the trailing portion of the version

In the series file, list actions to take

```
cat << EOF > debian/patches/series/2hls
+ linux-2.6.15.pants.patch
- buslogic-pci-id-table.patch
EOF
```

Patching “Official” Kernel Packages: II

And create a changelog entry for 2.6.15-1hls

```
dci -i
```

Reconfiguring "Official" Kernel Packages: I

Get `split-config`

```
# apt-get install subversion ruby
$ svn cat \
  svn://svn.debian.org/kernel/dists/trunk/scripts/split-config \
  > /tmp/split-config
$ chmod u+x /tmp/split-config
```

Run `split-config` for a given flavour in the root of unpacked kernel source package

```
$ /tmp/split-config debian/arch 686
```

`split-config` will spawn make menuconfig (by default)

Reconfiguring “Official” Kernel Packages: II

Once editing is finished, it will ask for the scope of the change

```
CONFIG_CRYPTO_DEV_PADLOCK_AES has been changed from 'y' to 'n'.
Do you want to make this change [G]lobal, per-[A]rch, or per-[F]lavour?
[GAF]
```

Has a few niggles as it doesn't massage Kconfig quite the right way

Might need to be run multiple times until all the options come up for scope confirmation

Tweaking "Official" Kernel Packages: I

Various options can be configured by editing '`find debian/arch/ -name defines`'

In particular, `debian/arch/ARCH/defines` sets:

- dependancies, suggests, recommends (per-arch or per-flavour)
- text descriptions of a flavour
- flavours to be built
  (trimming this is good if you only care about one flavour)

Tweaking "Official" Kernel Packages: II

If you edit any of these files, you should rebuild the control files

```
$ make -f debian/rules debian/control debian/rules.gen
```

And finally, build the packages

```
$ debuild -us -uc
```

Improving The Kernel Team's Processes

Nightly builds from SVN

Tools to isolate packaging problems:
missing symbols, ABI changes, stray files

More timely response to bug reports

Single source kernel

Out of Tree module infastructure - includes non-free

Questions?