Debian-Kernel Team Overview and Status
DebConf 5

dann frazier
dannf@debian.org
dannf@hp.com

Simon Horman
horms@debian.org

August 30, 2005

1 A Brief History

In the beginning, or at least as far back as our changelogs go, Herbert Xu maintained the Debian kernel-source package:

```
kernel-source-2.0.29 (2.0.29-6) frozen unstable; urgency=medium

  * Merged Debian changes from 2.0.30.

 -- Herbert Xu <herbert@debian.org>  Sun, 25 May 1997 22:26:30 +1000
```

Among his other Debian accomplishments, he also maintained the i386 and alpha kernel-image packages, and was the original author of Debian's initrd-tools.

Maintaining these packages was (and still is) a time-intensive task. Herbert was, in my experience, a maintainer who was very responsive to bug reports and e-mail, and excelled at keeping up with security issues and new upstream releases.

Since Herbert maintained the i386 and alpha kernel-image packages, builds for those architectures closely tracked new kernel-source releases. However, each of the other architectures had its own maintainer. Those maintainers may or may not use a kernel-source package as their base. Those that did use kernel-source did not all stay synchronized on the same version.

As an example, let's take a look at the woody release. There are 10 kernel-source packages in 3.0r5. An audit of the source packages that are included in 3.0r5 show 31 that produce kernel-image packages. Of those, 3 do not build-depend on any kernel-source package. Instead, these 3 packages provide their own linux source. Therefore, the security team would need to port a security patch to 13 source trees and rebuild 31 packages in order to fix a vulnerability in the woody distribution! And this is ignoring things like ABI changes which may require rebuilds of module packages, and other packages that include their own kernel binaries.

To attempt to solve this and other coordination issues, the creation of a debian-kernel list was requested by Francesco Paolo Lovergine and Sven Luther.

Herbert resigned from Debian on May 19, 2004. Andres Salomon suggested that a kernel team be created, citing the gnome team as an example of successful team package maintenance. Herbert handed over the responsibility of finding a new maintainer for the kernel packages to Martin Michlmayr, the presiding Debian Project Leader at the time. Martin's response was indeed an attempt to assemble a kernel team. Later that month, William Lee Irwin III announced his intent to NMU Herbert's kernel packages on behalf of the newly formed debian-kernel team.

Since then, much has been accomplished. I've captured some of the more notable milestones here.

- 2003.11.03: debian-kernel list requested
- 2004.05.19: Herbert Xu resigns
- 2004.05.22: William Lee Irwin III announces his intent to NMU Herbert's package, changing the Maintainer field to the kernel team
- 2004.05.23: Christoph Hellwig splits the monolithic Debian kernel patch into functional pieces
- 2004.06.15: First kernel-source upload by the kernel team (2.6.6-2)
- 2004.06.18: kernel project started on alioth
- 2004.06.21: initrd-tools added to svn
- 2004.06.23: Jeff Bailey sets up DebianKernel wiki
- 2004.07.12: source, ia64, powerpc and sparc in svn
- 2004.07.15: i386 added to svn
- 2004.07.27: amd64 added to svn
- 2004.07.28: mips and alpha added to svn
- 2004.08.09: hppa added to svn
- 2004.08.13: upstream 2.6.8 released
- 2004.08.14: kernel-source-2.6.8 uploaded
- 2004.08.17: s390 added to svn
- 2004.08.25: 2.4.27 chosen as the 2.4 kernel for sarge
- 2004.10.10: linux-kernel-di (0.1) released

- 2004.10.20: m68k added to svn
- 2004.11.02: kernel-source-2.6.9 ACCEPTED
- 2004.12.05: kernel changes ABI and breaks d-i for the first time
- 2005.01.08: kernel-source-2.6.10 ACCEPTED
- 2005.02.02: first formal debian-kernel irc meeting; 2.6.8 vs. 2.6.10 in sarge
- 2005.03.19: Andres Salomon commits kernel-source-nonfree-2.6.11-2.6.11
- 2005.03.26: kernel-source-2.6.11 ACCEPTED
- 2005.04.12: irc meeting, deciding that sarge kernels are now frozen

2 Who We are and What We Do

The Debian Kernel Team is made up of a loose collection of Debian Developers and non-Debian Developers. We maintain a number of kernel and kernel-related packages. This includes kernel-source-*, cramfs, initrd-tools, mkvmlinuz and the kernel-images for most architectures. We currently only maintain Linux kernels.

Historically, architectures that were maintained outside of the kernel.org tree would provide a kernel-patch package for their architecture, and apply that on top of the latest kernel-source package when preparing an image build. With the creation of the kernel team, this practice has become discouraged. It is our goal to build as many architectures as possible from a common source tree. This requires more active collaboration between porters, but has the benefit of additional peer review and avoids surprising conflicts between the kernel-source and the architecture patches later.

Like other debian teams, we communicate primarily via e-mail and irc. Though we have had two formally scheduled IRC meetings, we do not hold them regularly.

The kernel team uses svn.debian.org for source control.

3 Kernel Security

The Debian Kernel Team owns security support for Debian kernels in unstable and testing. Security fixes to kernels in stable are managed by the Debian Security Team. Like with any other Debian packages, users should not assume that all known security issues have been fixed in the unstable and testing versions. Less severe issues may be committed into svn but not uploaded until some more significant event triggers a release. Fixes that change the ABI maybe delayed to avoid breaking an existing release of DebianInstaller.

3.1 Security Maintenance

The Debian Kernel Team is notified of security vulnerabilities and fixes through multiple channels. The testing-security team tracks known security vulnerabilities in the whole of Debian/testing. The Ubuntu kernel team shares patches with us, and other issues come in through the bug tracking system. The Debian Kernel team doesn't currently monitor closed lists such as vendor-sec, so security fixes are not normally made available in unstable immediately after they become public.

Security is high-maintenance for the Linux kernel - a quick count of the changelog entries in kernel-source-2.6.8 that are clearly identified as security yields 55; and this is just one of the three kernel-source trees currently maintained by the Debian Kernel Team. The kernel team, principally Simon Horman and Andres Salomon, have been very diligent about getting vulnerabilities fixed as they come in.

3.2 Kernel Team/Security Team Maintenance Hand-off

Sarge will be the first stable release since the kernel team came into existence. As such, the hand-off process between the kernel and security teams is still being defined. The biggest issue is how we manage security maintenance during the freeze. Within this period, the security team hasn't taken over maintenance yet, but the kernel team is no longer able to make changes that will automatically propagate into testing.

During the freeze, the kernel team is still actively tracking security updates for unstable. Also, the kernels in testing more closely resemble the kernels in unstable than the kernels in stable. These arguments suggest that the kernel team is the logical maintenance unit during this period, and this will hopefully save some work for the security team once the release occurs. The current procedure is to continue uploading security fixes to unstable. This makes the packages available in an environment that is likely to receive user testing. Once sarge is released, these packages can be used as the basis for security uploads.

4 Bug Tracking

Due to the multiple layers in the existing kernel build process, a user discovered bug may exist in any one of a number of packages. Most bugs should eventually be assigned to the associated kernel-source package. Bugs related to the kernel config should remain with the kernel-image package.

Since new package names are used when we move to a new upstream release, we often need to reassign bugs against older packages. These bugs get reassigned to the pseudo-package "kernel" so that they aren't left behind and forgotten.

Andreas Barth put together a web page to track kernel bugs in sarge with a severity of important or greater: http://bts.turmzimmer.net/kernel.php

5 Debian Patch Set

The Debian Kernel team aims to provide kernels for our users that are functional, free, and secure, while minimizing our divergence from upstream.

5.1 Where Can I Find the Debian Patches?

Despite attempting to stay close to upstream, Debian applies a number of patches to the tree for one reason or another. These patches are contained in the kernel-patch-debian-VERSION package.

```
# In the kernel-source-2.4.27 directory unpacked previously
# Patch back to pristine source
/usr/src/kernel-patches/all/2.4.27/unpatch/debian
# Patch the tree back up to the latest debian version
/usr/src/kernel-patches/all/2.4.27/apply/debian
# Patch to an arbitrary release
/usr/src/kernel-patches/all/2.4.27/apply/debian 2.4.27-7
```

kernel-patch-debian-VERSION installs patches under `/usr/src/kernel-patches/all/2.4.27/debian/` Most patches include a comment at the top, describing briefly where they are from, what they do, and their status with respect to upstream.

For example:

```
$ bzcat /usr/src/kernel-patches/all/2.6.8/debian/ia64_cyclone_nosmp.dpatch.bz2
#! /bin/sh -e
## <PATCHNAME>.dpatch by <PATCH_AUTHOR@EMAI>
##
## All lines beginning with '## DP:' are a description of the patch.
## DP: Description: fix cyclone build on IA64 for UP
## DP: Patch author: Jesse Barnes <jbarnes@sgi.com>
## DP: Upstream status: submitted

. $(dirname $0)/DPATCH
```

And the meta-files in the `series//` subdirectory indicate which patch went into which version.

- A line beginning with a `+` denotes an added patch
- A line beginning with a `-` denotes an removed patch
- A line beginning with a `X` denotes removed file, usually to make the tree DFSG free.

5.2 Patch Acceptance Guidelines

Patches the kernel team like:

- Security Fixes
- Driver Fixes
- Stability Fixes

... actually, more or less any type of fix
Patches the kernel team generally reject:

- New features
- Out-of Tree Drivers
- My favourite patch-set

The best way to get a patch into the Debian kernel is to get it accepted upstream first. Not only does this signify a certain level of quality, but it gives us a guarantee that we will be able to drop this patch from our Debian-specific tree at some point.

6 Kernel ABI Changes

Occasionally, and usually due to a security issue, a change is accepted into a kernel package that changes its ABI[1] The ABI change is represented by a change in the SONAME portion of the kernel-image package name. For example, in `kernel-image-2.6.8-3-686-smp`, the SONAME is `3`. Changing the SONAME has a significant impact, even outside of the kernel team. Obviously, any kernel-module packages need to be recompiled against the updated kernel-headers.

Since a change in the SONAME changes the package name, apt will not automatically upgrade a system to the new kernel. In order for users to track the latest available kernel (meaning the latest kernel security fixes in the stable distribution), installing one of the kernel meta packages is strongly recommended. For example, `kernel-image-2.6-k7` currently depends upon kernel-image-2.6.8-2-k7. If the kernel ABI should be incremented, a newer version will be made available with a dependency on kernel-image-2.6.8-3-k7.

Changes to the kernel ABI also have an impact on the installer, which will be discussed in the next section.

[1]The upstream ABI interface policy does not discourage ABI changes. In fact, changing an interface to resolve a security issue can have positive characteristics. Quoting from Documentation/stable_api_nonsense.txt: “Security issues are also a very important for Linux (SIC). When a security issue is found, it is fixed in a very short amount of time. A number of times this has caused internal kernel interfaces to be reworked to prevent the security problem from occurring. When this happens, all drivers that use the interfaces were also fixed at the same time, ensuring that the security problem was fixed and could not come back at some future time accidentally. If the internal interfaces were not allowed to change, fixing this kind of security problem and insuring that it could not happen again would not be possible.”

7 Working with the Debian Installer Team

The Debian Installer architecture is very modular. The highly granular components (a.k.a., udebs) and their well defined interfaces provide a powerful mechanism for developers who wish to add or remove features, port the installer to new hardware, or provide new installer features.

To illustrate this, let's take a look at the components that are required to permit a user to install Debian onto an XFS filesystem.

- partman-xfs - partman is Debian Installer's partitioning and filesystem engine. partman-xfs plugs into partman and provides methods for selecting, creating, and mounting an XFS partition. The partman-xfs udeb has dependencies upon the xfsprogs and xfs-modules udebs.
- xfsprogs-udeb - This udeb provides the mkfs.xfs binary.
- xfs-modules - The udebs that provide this virtual packages provide the linux kernel modules necessary for XFS support.

This level of granularity makes it possible to build a tiny installer by stripping out all of the components that you know you're users will not need. An installer can be built that is limited to supporting reiserfs on IDE devices, simply by removing all of the unnecessary modules from the build. If additional components are needed, they can be pulled from a network repository (assuming the components necessary for accessing the network were not excluded, of course!).

Like many udebs, kernel modules udebs were originally built from the same source package as their deb counterparts. However, since kernel-images have a different source package for each architecture, creating these udebs created redundant work with inconsistent results. This is a class of problem that Joey Hess excels at solving.

Joey's solution was to create the kernel-wedge and linux-kernel-di packages. linux-kernel-di creates the necessary udebs by declaring a build-dependency upon the appropriate kernel-image, and copying its modules into the appropriate udebs. A separate linux-kernel-di source package exists for each architecture and kernel version. However, most of the required code lives in kernel-wedge, and is shared by all linux-kernel-di packages.

The number of indirections between a build of the Debian Installer and the actual source code used to create its kernel has led to a couple of interesting issues:

7.1 Kernel ABI Dependency

The Debian Installer may pull kernel module udebs from multiple locations during an installation, and these modules must be compatible with the runtime kernel of the installer. As mentioned in section 6, occasionally a change is included in a kernel build that breaks the module ABI. If the kernel module udebs corresponding to the kernel used in a given build of Debian Installer are superseded by modules conforming to a different ABI within a release, then certain classes of network installs will break.

7.2 Version Skew and GPL Compliance

Each link in the chain of packages that produce kernel udebs has its own lifecycle. Consider the following chain of events:

1. kernel-image-2.6.8-ia64 (2.6.8-1) is built against kernel-tree-2.6.8-1
2. linux-kernel-di-ia64-2.6 (1.0) is built against the above kernel-image
3. All of the above packages propagate into testing
4. kernel-image-2.6.8-ia64 (2.6.8-2) is built against kernel-tree-2.6.8-2
5. kernel-image-2.6.8-ia64 (2.6.8-2) propagates into testing

At this point, the source code to recreate the Debian Installer build would no longer exist in testing. Because of the "rollback" facility in kernel-tree, it is believed that the GPL terms are satisfied if a new kernel-tree propagates into testing before kernel-image is rebuilt against it. However, kernel-image does not provide a similar feature. Thanks to James Troup and the release team, this issue was resolved with the creation of a special sarge-r0 distribution containing packages used to build Debian Installer.

8 In the Works

8.1 Architecture Synchronization

Earlier 1, I noted that a major problem with the woody release was the large number of different kernel-source packages used to build kernel-images. Minimizing the number of active kernel-source packages is one of a number of architecture synchronization issues the kernel team is attempting to resolve. A daily-generated page provides a window into how closely synchronized architectures are on a given kernel-tree release: http://people.debian.org/ dannf/kernel-stats/kernel-avail.html. In the future, Dann plans to update this page to track linux-kernel-di packages as well.

kernel-source skew isn't the only architecture synchronization issue we face. Other issues include:

- Consistent Kernel Configs - Each architecture maintains kernel config files independently. It would be nice if all architectures had a common feature set and the same things were built as modules on all architectures, where feasible.
- Build System Bugs - Many of the kernel-image packages diverged from the same code base. When a bug in the build system is discovered it may be fixed within one of the packages, only to pop up on a different architecture later. Structural changes, such as the introduction of the kernel-tree concept or the building of udebs for Debian Installer, usually take a long time to complete (if ever), since the changes must be made to multiple packages. `kernel-package` provides a code base that is shared among multiple architectures, but more commonality is desired.

These are some of the reasons the kernel team is looking to create a unified source package that builds kernel packages for multiple architectures.

8.2 A Common Source Package

At the time of this writing, Jurig Smakov has announced a working prototype of a common kernel-source-2.6.11 package that is capable of building i386 packages. The 2.6.12 timeframe is being suggested as a goal for a first upload of this package.

8.3 Moving drivers to non-free

With 2.6.11, the kernel team has started the process of moving drivers which contain binary blobs to the non-free section. These drivers have been removed from main, and Andres Salomon has created a kernel-nonfree-modules package. However, many of these firmware blobs exist in source files with only a GPL boilerplate. Since these firmware blobs are not provided in "the preferred form of the work for making modifications to it", as required by the GPL, Debian cannot legally redistribute these files. Sven Luther and other members of the kernel team are working to contact the associated copyright holders, requesting they modify the license to explicitly grant redistribution rights of this data in "blob" form. At the time of this writing, this process is beginning to produce some results. Both QLogic and Broadcom (tg3 driver) have provided promising responses.

The current status of this process can be found at: http://wiki.debian.net/?KernelFirmwareLicensing.

8.4 initramfs

initramfs is a feature new to 2.6 linux kernels, and aims to solve many of the problems associated with initial ramdisks (initrds).

Jeff Bailey has been working on an initramfs-based replacement for `initrd-tools`.

Quoting from http://www.ubuntulinux.org/wiki/Initramfs, some of the benefits of initramfs include:

- CPIO archive, so no filesystems at all are needed in kernel. The archive is simply unpacked into a ram disk.
- This unpacking happens before do_basic_setup is called. This means that firmware files are available before in-kernel drivers load.
- The userspace init is called instead of prepare_namespace. All finding of the root device, and md setup happens in userspace.
- An initramfs can be built into the kernel directly by adding it to the ELF archive under the section name .init.ramfs
- initramfs' can be stacked. Providing an initramfs to the kernel using the traditional initrd mechanisms causes it to be unpacked along side the initramfs' that are built into the kernel.
- All magic naming of the root device goes away. Integrating udev into the initramfs means that the exact same view of the /dev tree can be used throughout the boot sequence. This should solve the majority of the SATA failures that are seen where an install can succeed, but the initrd cannot boot.

By including hotplug-ng and udev, we will be able to perform device autodetection from within the initramfs; however it is unclear if Debian users will be able to handle the memory footprint required to have all possible modules available at initramfs time. `initrd-tools` tries to solve this problem by detecting the necessary modules at initrd creation time, but this process is error-prone.

Jeff has also been able to reduce the size of an initramfs when compared to a typical Debian initrd by using klibc instead of glibc.

Unfortunately, not all Debian architectures have a 2.6 kernel, so it will be some time before we can do a universal switch to an initramfs process.

8.5 Automated/Nightly Testing

Automated testing from source control is a goal for the kernel team, but development is in the early stages. Maybe we'll see some progress here at DebConf5!

9 How Can I Help?

Bug reports are always welcome - the more information you can provide, the better. Try to do as much investigation as you can prior to reporting a bug.

- STFW :) Check LKML/Google to see if the bug you found has already been discussed.
- Test the latest upstream version of the kernel to see if your bug still exists.
- If the bug came about after an upgrade, try to narrow down the version in which it was introduced.
- Take the time to make sure someone else hasn't already filed your bug by searching the bug tracking system for bug