1 Debian Kernel Packages and Boot Loaders for i386

Debian has very good support for building packages of kernels, managing patches, and updating boot loaders, all in a fairly automated fashion. This article will cover how to build a kernel package (with optional kernel patches), and how to update the boot loader configuration.

2 Kernel Packages

The package needed for creating kernel packages is called 'kernel-package'. There are many advantages to creating packages for kernels, quite apart from the convience of using dpkg to keep track of what versions you have installed. It gives you the ability to easily compile a kernel for a different computer locally, and move it across to the destination with very little effort.

It also integrates very well with debian packaged kernel patches, allowing you to keep your local kernel sources without patches, and have them automatically applied for compilation. It also integrates neatly with the various modules that may be required, such as pcmcia-cs or graphics drivers.

Debian kernel patch packages are named 'kernel-patch-blah', where blah is representative of what the patch does. To see what kernel patches are available, simply type:

```
$ apt-cache search kernel-patch-
```

Note that all of the following assumes you have your kernel source tree in /usr/src/linux, and are in that directory.

To begin building a kernel, make sure you have the kernel-package package installed, as well as any patches you may require. To configure the kernel, run the following command:

```
$ export PATCH_THE_KERNEL=YES  # Only needed if you have patches
$ make-kpkg clean
$ make-kpkg --config=menuconfig --revision=host1.0 configure
```

This will apply any appropriate patches, and bring you into your chosen kernel configuration method (in this case, menuconfig) and give the kernel package the appropriate revision (in this case, host1.0). Simply configure the kernel as required, and then exit out. The process will then continue by running a make dep for you.

To create the actual kernel package, now run:

```
$ fakeroot make-kpkg binary
```

This will now actually compile the kernel, and create the kernel image, header, source and doc packages. If you only want the header and image packages, replace binary with binary-arch. Note that the fakeroot command can be replaced with whatever you need to run the program as root.

The files will be output in /usr/src, and will be named in the following manner:

```
kernel-{image,header,source,doc}-x.y.z_host1.0_i386.deb
```

It is now a simple matter of installing the package. This will install the kernel into /boot/vmlinuz-x.y.z, and update the /vmlinuz and /vmlinuz.old symlinks. The kernel configuration is kept in /boot/config-x.y.z.

Now, before we reboot we need to ensure that the boot loader configuration is correct.

3 Boot loaders

Under i386, there are 2 main popular boot loaders - lilo, and grub. They both have their advantages and disadvantages which will not be covered fully here.

3.1 LILO

Kernel package comes configured assuming you will be using lilo. If you have something like the following lilo.conf, you will have to do very little work.

```
boot=/dev/hda
root=/dev/hda5
install=/boot/boot.b
map=/boot/map
delay=20
vga=normal
default=Linux
image=/vmlinuz
        label=Linux
        read-only
image=/vmlinuz.old
        label=LinuxOLD
        read-only
        optional
```

3.2 Grub

After installing the grub package, you will need to do something like the following:

```
$ sudo grub-install /dev/hda # or your install device
   OR if you have a seperate /boot
$ sudo grub-install --root-directory=/boot /dev/hda

$ sudo $EDITOR /boot/grub/menu.lst
$ sudo update-grub
```

Now update your kernel-img.conf to include the following lines:

```
postinst_hook = /sbin/update-grub
postrm_hook = /sbin/update-grub
do_bootloader = no
```

Now, whenever you install a new kernel package, it will automatically update the grub configuration as appropriate.

4 Conclusion

As you can see, creating a kernel package under Debian is very simple and convenient, and has many advantages from doing it the old way. Keeping the kernel image, modules and configuration together makes it easy to build the kernel on one host, and move it to another. There is also the facility for given a kernel package arbitary post install scripts, which is also very useful.

For more information, see kernel-package(5), make-kpkg(1), lilo(8), lilo.conf(5), grub(8), grub-install(8) and update-grub(8).