Nori's mini-HOWTO:
Debian Woody on the Dell Inspiron 8000

Nori Heikkinen

August 13, 2003

Rationale

All my previous installs of Debian have been really easy, but then again they've all been on PC desktops. The deal has been:

1. Download the first ISO image onto CD
2. Stick the CD in your drive
3. (Make sure you have the BIOS set to boot off the CD before the hard drive)
4. Follow instructions, generally hitting 'okay' for the next half hour
5. Ignore tasksel by and large
6. Ignore most of the dselect list, and only put on what you need later
7. ...and *voilà*, a Debian install!

It was not this easy this time 'round. For whatever reason, this little i8k did *not* seem to want Debian on it! From start to finish (or rather, as finished as it's going to get, because I need to be doing work), it took a week and a half, and I ended up scrapping the entire install and beginning from scratch twice.

This document is half for myself, in case I need to do this install over again, and half for anyone else out there, who has had as much trouble as I have with the whole process.

Good luck.

Configuration

I was working with the following system:

Dell Inspiron 8000

- PIII, 697 MHz

- 256M RAM
- ATI Technologies Inc Rage Mobility M4 AGP
- ESS Technology ES1983S Maestro-3i PCI Audio Accelerator (rev 10)
- a Dell Logitech scrolly-wheel infrared mouse, model M-UR69

I was working off a Debian Woody image with the stock kernel 2.2.20-idepci, downloaded from the Rutgers mirror here:

```
ftp://ftp.rutgers.edu/pub/debian-cd/images/i386/debian-30r1-i386-binary-1.iso
```

This image had a corrupt library (`libpcap0_0.6.2-2_i386.deb`), which prevented me from using the CD as my primary installation source (I did a network install instead). I do not recommend it!

1 Video Issues

I had two large problems with the video card (or, for lack of my understanding of what that incorporates, let's say "the graphical elements of the system"). The first prevented me from even beginning the install in a reasonable manner, so I'm addressing them first.

1.1 LILO parameters

When I stuck my fresh new Debian CD in my drive and rebooted, a big blue welcome screen like I am accustomed to did *not* pop up. Instead, as soon as the CD started loading its default kernel (2.2.20-idepci), the screen split itself in four, overlapping its text at the horizontal quarters. To get an idea of what this was like (if you're not already staring at it), picture the normal screen, then copy it, and scoot the second image down so it overlays the first, at the 1-quarter mark. Do the same again at the halfway mark, and the 3-quarter mark, and you have something that is utterly illegible by the bottom.

As I now understand it, this has something to do with framebuffer support. But instead of disabling that in the kernel (which you might want to do if you ever get around to rolling your own, for some *really good reason* (I'm not a fan — see why in Section 5!)), you can just pass a couple options to your boot loader.

Debian uses LILO by default, so at this point, you can just use the following (I don't know how to do it in grub, but I'm sure it's possible):

```
video=vga16:off
```

This means that instead of hitting 'ENTER' as usual and watching the stock kernel load, you will need to type:

```
linux video=vga16:off
```

(The kernel image, followed by the boot parameters.)

You can now boot up okay, but you'll need to be wary of this when you're glibly 'ENTER'ing through menus later in the install — when you're asked if you want to append anything to the kernel image for the boot loader, you need to type that line (`video=vga16:off`) again. If you don't, the screen will get split again.

You can fix a split screen if you already have an installed system by editing your `/etc/lilo.conf`. In your `linux` stanza, just put `append="video=vga16:off"` (this is what would have happened if you'd specified it in the base system configuration). Then run `lilo` — your changes won't take effect until you do. (This will also tell you if you've screwed it up somehow — it's good to double-check.)

1.2 Xserver-XFree86 Configuration

Even after I had the base system installed and happy, I couldn't get X working as easily as I had in the past (which was basically it coming up upon rebooting after the base installation). I must have configured something wrong, because instead of starting X as usual, the screen flashed a couple times, and then kicked me back to a console.

The way to reconfigure the X-server is by using `dpkg` (what else?). Do:

```
dpkg-reconfigure xserver-xfree86
```

to reconfigure X. It's best just to change one parameter at a time, so you can see what's working and what's not.

1.2.1 Which Video Card

Even though I had an ATI card, choosing `ati` in the video card section did not work — I had to choose `r128`. Strangely enough, *this is not an option during the base install!* I have no idea why this is. So if you're using the same image, you will need to reconfigure the X-server no matter what if you need r128 support.

When you reconfigure (as root, of course), choose `r128`. Then restart X with a `'/etc/init.d/xdm restart'`.

Having `r128` as your video card might make X come up, but unless you've magically specified everything else correctly (and the important parameters — at least the ones I had trouble with — are the horizontal and vertical refresh rates, and the screen size), it may be too small (which I never saw), distorted (which I saw a lot), or even turn white and freeze the entire computer (which I also saw a lot).

To prevent having to do a hard reboot (because when X crashes into the mysterious white screen of death, nothing — not even the three-finger salute — can send the machine the signals it needs to cleanly shut down), have your fingers ready over the CTRL+ALT+F1 keys to take you back to your console if the display is not **perfect** — it might not crash if it's not, but if it does, you don't want to have to not only wait for it to check all its inodes (if you haven't yet `tune2fs`'d it to ext3 — see Section 7.1), but perhaps screw up in the process of so many crashes.

If that's perfect, great! If not, read on . . .

1.2.2 Horizontal and Vertical Refresh Rates; Screen Size

If it's not perfect, quickly switch back to the console, and try again. I had to play with the screen sizes a bit. A friend suggested starting with the lowest option (640×480) and work up from there, but I found that only the exactly correct one worked. Maybe this is a good approach for a desktop system, but not a laptop.

Another friend, however, tell me that " that's almost guaranteed to fail on a laptop, as I understand it. Laptops have LCD screens, which are more finicky than LED. When going through the dpkg-reconfigure, you need to specify the screen-size that it is capable of, and tell it that it's an LCD screen, and it ought to be able to get the rest." That seemed to have been more correct in my case.

I wanted a screen at 1600×1200, so I set that for my screen size. The first time I installed Debian on this machine, that plus the r128 video card did the trick, and I was set. The second time, however, I also had to play with the horizontal and vertical refresh rates. I ended up setting them as high as I could, and that plus the large screen size *plus* the r128 card did the trick. It felt like voodoo.

2 Samba and Interfacing with the Windoze Network

Since my workplace has a Windoze share that they use all the time, my biggest concern with this install was interfacing with that in a reasonable manner. By "reasonable" I do not mean big blue-and-green buttons with "click here to view all files" tabs and talking paperclips (though you're welcome to those if you want — see `vigor` if you really want the paperclip!), but without having to maintain a dual boot to Windoze.

This should have been easy. But since I didn't know anything about kernel modules, and little things went wrong, it was harder than it should have been.

2.1 Installing Samba

Like anything in Debian, installing samba is easy:

```
apt-get update
apt-get install samba smbclient
```

But, because you'll probably want to print, too, you'll need all of the following (or so a website told me, and it worked):

```
   apt-get install cupsys cupsys-bsd cupsys-client foomatic-bin samba
smbclient gs-esp a2ps
```

(You don't have to do that now — if you don't want to print, or don't want to do it through CUPS, only bother with the `samba` and `smbclient` packages.

2.2 Configuring Samba

I followed the instructions in the wonderful SMB HOWTO[1] for the basic configuration. The `netbios` lines were already in my `/etc/services`, and I uncommented the relevant `netbios` lines from my `/etc/inetd.conf`. They differed slightly from the ones in the HOWTO . . . mine looked like this:

```
netbios-ns      dgram   udp     wait    root    /usr/sbin/tcpd /usr/sbin/nmbd -a
netbios-ssn     stream  tcp     nowait  root    /usr/sbin/tcpd /usr/sbin/smbd
```

I think these lines are for running samba servers, which I'm not trying to do (I just wanted to access samba shares as a client machine). You probably don't need them if you're just a client machine, too, so you could safely skip this section.

I didn't need to do anything to my `/etc/samba/smb.conf` file (note the slightly different location — no longer is it just in `/etc`) — I chose to manage it with `dpkg`, and it generated a great one for me.

2.3 Using Samba

At this point, I could see the Windoze share I was going for:

```
smbclient -L <servername>
```

But I couldn't mount it. Maybe you can — before you futz with other things, try to mount the samba share you want so:

```
mount -t smbfs -o username=name,password=passwd //server/share /mountpoint
```

For this to work, however, **you have to have smbfs in your kernel**. I didn't, and I didn't know how to put it there.

2.3.1 Getting `smbfs` in your kernel

This was the hard part for me. The first time, I tried to reroll my kernel with smbfs support, and install that — I'd done it before, and `make-kpkg` (in

[1] http://www.tldp.org/HOWTO/SMB-HOWTO.html

`kernel-package`) makes it really easy. But it was not to be. I managed to screw up my whole install (and the RedHat and Windoze partitions on the box as well, by accidentally installing MBRs in them!). So, don't do that.

What you want to get down with is **LKMs — Loadable Kernel Modules**. In a nutshell, what is actually compiled into the kernel is part of the base kernel, and then you can stick modules (which you need compile separately) into a running kernel. This is what you'll need to do with smbfs.

I'm sketchy on The Debian Way on how exactly to compile the smbfs module and sticking it into my kernel, although apparently i accomplished it. What worked for me was the following:

(Assuming you're using a stock kernel:)

- `apt-get` the *source* (not image) of your current kernel (`apt-get install kernel-source-‘uname -r‘`.
- `cd /usr/src/kernel-source-‘uname -r‘`
- `cp /boot/config-‘uname -r‘ ./config`
- `make menuconfig` (or whatever kind of config you want)
- Here, it's as if you're rolling your own kernel, but in fact all you want do is choose SMBFS support as a module. It's under "filesystems" or something pretty intuitive.
- Save & quit ...
- `make-kpkg modules`

That, plus a reboot, totally did it for me (I think). YMMV — it's along those lines, at least.

2.3.2 Mounting a Samba Share

The rest is cake. Read `man mount` if you haven't, and then try a:

```
   mount -t smbfs -o username=<name>,password=<passwd> //sambashare
/mountpoint
```

The samba share should mount on your mountpoint.

If that doesn't work — like if you have a username with a space in it (argh! the Windoze aesthetic is becoming the standard!), you'll need to create a so-called **"credentials file"** and put your info in there. The formatting goes like this:

```
username=8TRACKO/Nori Heikkinen
password=mypassword
```

. . . where “8TRACK0” is my domain, “Nori Heikkinen” is my username, and “mypassword” is my password (not really, ha ha). You can call this file whatever you want to, and put it wherever you want. I called mine `.smbmount-ned` and put it in my homedir; it really doesn’t matter.

Now that you’ve got that file, you can use it to mount the share:

```
   mount -t smbfs -o credentials=/home/nori/.smbmount-ned //sambashare
/mountpoint
```

Once this works and you have the share mounted, you can then stick a similar line in your `/etc/fstab`, and it will be mounted automatically every time you boot up. My line looks like this:

```
//8track/NED    /8track/ned     smbfs credentials=/home/nori/.smbmount-ned,rw
```

2.3.3 Voodoo Magic

On the third install, I ran into problems mounting the samba share. I had done everything exactly as before, so I was miffed. I was getting the error:

```
  mount: wrong fs type, bad option, bad superblock on //8track/NED,
         or too many mounted file systems
```

In the end, I realized I didn’t have `smbclient`, which is in the `smbfs` package. Again with the voodoo — I just did a

```
   apt-get update
   apt-get install --reinstall smbfs
```

and all was well. Weird.

2.4 Printing to a Windoze Printer with CUPS

This part was surprisingly easy once I had samba running. I was prepared to start grokking the printcap system, and printer daemons and the like — when in fact, all I needed to do was apt-get install cups and all its peripherals:

```
   apt-get install cupsys cupsys-bsd cupsys-client foomatic-bin gs-esp
a2ps
```

(These are just all the things you might have installed in Section 2.1, minus the samba packages themselves.)

CUPS then has a lovely little interface for you through your browser.

2.4.1 Using the CUPS browser interface

CUPS' printer controls are located, conveniently, here:

```
http://localhost:631/
```

There's a menu of tasks to choose from, from which you can configure new printers and manage the jobs of already-configured ones. You'll have to be root for this, but working through a browser that a user owns. (So, just open up your favorite browser in an X-session as you, and then enter root's name and password at the prompt.)

From here, it's easy. Go to "Manage Printers," then "add printer." You'll be prompted for a "device URI," which apparently stands for Universal Resource Identifier. Since you're printing to a Windoze printer over samba (you choose this somewhere), it'll start you off with `smb://`. Just fill in the path of your remote printer from there. My printer was on
`BRN_31D11E` (trust that Windoze to come up with some weird, arbitrary printer domain name!), and the printer name was `BINARY_P1`, so my device URI looked like this:

```
smb://BRN_31D11E/BINARY_P1
```

Pretty straightforward.

My printer model (Brother HL-1270N) didn't show up in the list, so I just clicked "generic printer" (or whatever it was).

And, presto! The thing's configured! Try printing a test page — mine worked perfectly.

Now, you can use the BSD-style `lpr` commands to queue things up. Either remember to specify the printer each time (with the `-P` switch, so: `lpr -P printername foo`), or set your `$PRINTER` environment variable to the name of your printer (which, in my case, was `BINARY_P1`).

2.4.2 Getting OpenOffice to print

Getting the newly-configured printer to work with OpenOffice was a little bit trickier, but still easy. Definitely non-intuitive.

After a bit of googling, I found that you have to run the `spadmin` script that pops up in the OpenOffice install in order to get it to talk to you printer.

As you (not root), do the following in X:

```
/usr/lib/openoffice/program/spadmin
```

A little win-like interface pops up, and it's easy to click through it to add your new printer, which should show up. It did for me, at least :-).

3 External Mice

This took me a while to figure out. For whatever reason, I couldn't just plug in my external USB mouse (which I *needed* in order to not kill my wrist on that damn trackpad![2]) and have it work. Instead, like with the rest of this install, I went through a whole song and dance about it. The results follow.

CAVEAT: This is with the Dell Logitech scrolly-wheel mouse I have. YMMV.

3.1 Necessary kernel modules

To see what's loaded, do a '`/sbin/lsmod`'. For USB input devices in general, you will need the following:

- usb
- evdev
- mousedev
- keybdev

I'm pretty sure that for just USB mice, just `usb` and `mousedev` are necessary, but putting them all in as modules can't hurt — an usused module doesn't matter.

Also make sure that you're loading these modules at boot time: put them in `/etc/modules`.

USB by itself isn't a module; it needs to be aliased to either `usb-ohci` or `usb-uhci`. For the Dell I8k, you want the `usb-uhci`.

Alias the module by editing `/etc/modutils/aliases`, and put in the following lines:

```
alias usb usb-uhci
post-install usb modprobe hid
```

Then run `update-modules`, and those aliases will get stuck in your /etc/modules.conf. **Don't** just edit this file by hand; your changes will get overwritten!

Got all that? Excellent. Reboot if you want (but you don't need to – this is Linux!), or just `modprobe usb evdev mousedev keybdev`, and continue! (You don't need to `modprobe usb-uhci`, because now you've aliased that to `usb`.)

[2] As it has been noted, what works for some may not work for others. I personally cannot use a flat keyboard or a trackpad mouse, but others say that my ergonomic keyboard and external mouse hurt their wrists. As with all matters of personal preference, YMMV.

3.2 Configuring X

Dman, of debian-user, made a nice little summary[3] that I’m going to copy wholesale:

> If you want mouse in console and X:
>
> - gpm reads from the mouse device itself. With a PS/2 mouse this is /dev/psaux. With a USB mouse (and devfs, I haven’t used USB without devfs) it is /dev/input/mice. gpm needs to be told the correct protocol to use. For many modern mice (including mine) the protocol is ’imps2’.
> - gpm needs to repeat with protocol ’raw’. It repeats through a named pipe, named /etc/gpmdata.
> - X needs to read from /etc/gpmdata and use the same protocol that gpm is using. In the case of my mice that protocol is ”IMPS/2”.
>
> If you want mouse in X only:
>
> - X reads from the mouse device itself. With a PS/2 mouse this is /dev/psaux. With a USB mouse (and devfs, I haven’t used USB without devfs) it is /dev/input/mice. X needs to be told the correct protocol to use. For many modern mice (including mine) the protocol is ’IMPS/2’.
>
> Thanks, Dman!

I also needed to edit my `/etc/X11/XF86-Config-4` manually to include the other mouse, as per a suggestion on debian-laptop[4].

I added:

```
Section "InputDevice"
        Identifier      "Logitech Mouse"
        Driver          "mouse"
        Option          "SendCoreEvents"        "true"
        Option          "Device"                "/dev/input/mice"
        Option          "Protocol"              "ImPS/2"
        Option          "Emulate3Buttons"       "true"
        Option          "ZAxisMapping"          "4 5"
EndSection
```

and the “Logitech” line in the following section:

```
Section "ServerLayout"
        Identifier      "Default Layout"
        Screen          "Default Screen"
        InputDevice     "Generic Keyboard"
```

[3] `http://lists.debian.org/debian-user/2003/debian-user-200306/msg02922.html`

[4] `http://lists.debian.org/debian-laptop/2003/debian-laptop-200306/msg00150.html`

```
        InputDevice     "Configured Mouse"
        InputDevice     "Logitech Mouse"
EndSection
```

For reference, my `/etc/gpm.conf` looks like this:

```
device=/dev/input/mice
responsiveness=30
repeat_type=raw
type=imps2
append=""
sample_rate=
```

The important clauses there are the device, the repeat type, and the type. Mine is ImPS/2 (`imps2`, because it has a scrolly wheel; yours might just be plain old PS/2 `ps2`).

4 Configuring Sound

Of the least importance to functionality, but perhaps the most importance to long-term productivity in the work environment being configured, I need to be able to listen to music. I have access to a great music share on Windoze, so I needed to get samba (see Section 2) working first.

I've had many battles with sound cards, and this one was the least of all of them. There are specific Linux sound HOW-TOs out there[5], and they were kind of useful — but this was a pretty trivial configuration.

First off, just try playing something logged in as yourself. If that works, you're good to go. If not, read on.

4.1 Inserting the Sound LKM

Check out what sound card you have (‘`/sbin/lspci`‘ shows the devices you have installed). The relevant line looked like this for me:

```
Multimedia audio controller:  ESS Technology ES1983S Maestro-3i PCI
Audio Accelerator (rev 10)
```

Of course, to be able to do anything with sound, you'll have to have the right drivers or modules in your kernel. I was using the plain old stock `2.4.20` kernel, which had the Maestro enabled as a module (see Section 5 for kernel info & caveats), so all it took was a little creative guessing and a ‘`locate maestro`‘ to figure out which module it would be. Then all I had to do was insert it:

```
spycellar:#̃ modprobe maestro3
```

[5] `http://new.linuxnow.com/docs/content/Sound-HOWTO-html/Sound-HOWTO.html`

and to make sure it got inserted at boot time, by adding it to `/etc/modules`.

Make sure your sound devices exist (`/dev/dsp*`). Mine did.

At this point, try catting a wave file to your sound device, to see if that did the trick:

```
spycellar:~# cat wavfile.wav > /dev/dsp
```

Now you're almost there. You just need to add your username to the audio group, and then you should be set.

4.2 Adding Yourself to the Audio Group

The easy, hackish way to give yourself permissions to use the sound devices would be to just chmod the devices to 644[6]. While this will work, however, it's not The Right Way to do it.

Instead, if you look at the sound devices, you'll see that they're owned by the group 'audio'. The Right Way to do sound is to add yourself to the audio group, and then to change the permissions on the sound devices so only users in that group can read from and write to them. That way, you have control over the sound I/O going on in your machine. (It's perhaps a little paranoid approach for any single-user laptop, but it's good practice, and should be implemented on any desktop system.)

To add yourself to the audio group, do:

```
spycellar:~# adduser yourname audio
```

To make sure this worked, you can list the groups you're a part of like this:

```
spycellar:~# groups yourname
```

You should see 'audio' as one of those.

To change the permissions of the sound devices so only audio users can read and write to them, do:

```
spycellar:~# chmod 0660 /dev/dsp*
```

After which, mine looked like this:

```
crw-rw----    1 root     audio     14,   3 Jun 30 12:17 /dev/dsp
```

[6] see an explanation of permissions in Section 7.2.

```
crw-rw----    1 root     audio     14,  19 Jun 30 12:17 /dev/dsp1
crw-rw----    1 root     audio     14,  35 Jun 30 12:17 /dev/dsp2
crw-rw----    1 root     audio     14,  51 Jun 30 12:17 /dev/dsp3
```

You might have to restart X for the changes to take place within X, though they should work immediately on the console.

This was all it took for me. If you're still at a loss, you need more help than I can give sound-wise — good luck!

5 Rerolling your Kernel

Avoid this.

Even if you don't avoid it, before you attempt to re-install a kernel you rolled, read the manpages on the following:

- lilo
- install-mbr
- modprobe, insmod, lsmod

and check out the content of `/proc/filesystems` (to see what filesystems you're supporting in your current kernel).

To see what's in your current kernel, look at your config file, which lives in `/boot`. It'll be `/boot/config-‘uname -r‘`.

And again, I urge you not to recompile. Maybe you know more about this stuff than I do (which is likely), but you want to know a **lot** more than I do before you do it.

Which is weird, because all my other kernel rerollings and installations in debian — *on desktop systems without other OS partitions* — have gone really smoothly.

Whatever. Let the buyer beware, and know what you're getting into. But more power to you.

6 If You Decide to Reinstall ...

During this frustrating endeavor, I twice had screwed up my install so far that I decided it would be simpler to scrap what I had and to reinstall the whole shebang than to try and fix what I currently had. I think this was a good call both times, and it felt easier the second and third times around — what had taken me a day or two now took me an hour or two.

6.1 Trashing It

I made the mistake both times of forgetting to back up parts of my system. If you reinstall, **back up your homedir and `/etc`**. The homedir, because if you've configured your shell, window manager, or any programs you've downloaded, your settings are saved there; /etc, because the system files that you spent so long getting right are saved there.

The first time, I just plain forgot; the second time, I purposely didn't back up /etc, because the system files I'd screwed up were in there, as well as the ones I wanted. Turns out I should have taken the five minutes to weed through which were which, and to back up only the good ones, as that would have saved me hours of headache down the road.

Live and learn.

6.2 Rescuing an Install

Instead of totally trashing your system, though, and having to reinstall, if you've only screwed up part of it (like LILO, for example), you can rescue it. Ironically, I'd only ever before done this with KNOPPIX (which is great, that's just not its purpose)! Turns out the Debian CD from which you installed works just as well.

If you want to rescue your install, but bypass your kernel loading and use the stock one, you can type:

```
rescue root=/dev/hdaX video=vga16:off
```

at the prompt instead of just hitting ENTER. This will use the stock kernel on the disc to boot up your install, if it's rescue-able. (Don't forget the video option!)

If you don't even want your filesystem mounted (in order to convert it to ext3, or something) you can hit ENTER at the first screen (loads the image 'linux'), and then when the big blue welcome screen pops up, switch to tty2 (CTRL+ALT+F2). This is a small shell, ash, without any big editors, and with only a few commands (to see which, as it says, do a 'ls /sbin /usr/sbin' (and more; I forget)). Unfortunately for me, there is no loadkeys (I'm a Dvorak dork), but I discovered that if you mount your root filesystem and then chroot to it, you can 'loadkeys dvorak' from there and the setting will persist even when you exit an unmount it.

7 Miscellaneous

7.1 ext2 vs. ext3

At some point in your install, you'll probably want to switch filesystem types. In the base install, you're only given a choice of ext2 (short for ext2fs, or "second extended filesystem," which is the "standard" UNIX filesystem[7]. Ext3fs[8] is the same as ext2, but provides journaling.

[7] http://www.tldp.org/HOWTO/Filesystems-HOWTO-6.html#ss6.2

[8] http://www.tldp.org/HOWTO/Filesystems-HOWTO-6.html#ss6.3

For those as sketchy on filesystem types as I am, it seems to be pretty basic. In the README on the original ext3 download page, the author answers the journaling question:

> **Q:** What is journaling?
> **A:** It means you don't have to `fsck` after a crash. Basically.

This is useful, because it means that every time your screen whites out and crashes while choosing the right video card (Section 1.2.1), you don't have to sit through an entire filesystem check of every inode. The filesystem still `fsck`s itself every X mounts or Y days, but doesn't put you through the entire wait every time you crash it.

To convert partition,s to the ext3 filesystem, you need to cleanly unmount them, boot something else (like the Debian CD you installed from — see Section 6.2 on how to do this), and then, on a console, do:

```
tune2fs -j /dev/hdaX
```

wherein `/dev/hdaX` is the partition you want to add journaling to (hence the '-j' flag).

Don't forget to modify the lines in your `/etc/fstab` to reflect that the partitions in question are to be mounted as ext3, not ext2. When cleanly unmounted, they can still be mounted as ext2, but the whole point of changing them was so they wouldn't be.

That's it. When you reboot, your partitions should come up as ext3.

7.2 An Explanation of UNIX Permissions

UNIX permissions, while seemingly complex, are really easy. There are 10 fields in any set of permissions, arranged as follows:

- 1 special bit
- 3 user bits
- 3 group bits
- 3 others bits

The special bit is used for things like directories (when it's set to 'd'), links ('l'), and other things. Don't worry about it; you never need to explicitly set it.

The other three groups are all divided in the same manner — the first bit means that the user/group/other has **read** permissions ('r'), the second **write** ('w'), and the third **execute** ('x').

There's a cute binary way of setting permissions that I find most intuitive. The read bit is worth 4, the write 2, and the execute 1. (Hold up the leftmost three fingers on one hand and pretend you're counting in binary). Therefore, any combination of u/g/r permissions can be specified by three unique numbers. '700' means that the user has read, write, and execute permissions (6+4+1) but that no one else does; '644' means that the user has read and write permissions, and that everyone in the same group and all others have only read permissions.

It's easy to change permissions using this notation. To change a file to 644, for example, just do:

```
chmod 644 file
```

8 Other Resources

Had I been on my own for this install, I would probably still be going. I had many patient people around me, who indulged me not only in letting me put Linux on this box in the first place (it *is* a company laptop, after all!), but then in helping me sort out my issues with it.

The best thing I can recommend are LUGs (Linux Users' Groups), and there are two relevant ones for Debian: debian-user[9] and debian-laptop[10], and they are excellent resources. Beware, though — especially debian-user is quite high-volume, so you want to filter your mail before you subscribe!

Thanks

The debian-user and debian-laptop lists[11] in general gave me lots and lots of helpful suggestions, as did the Swarthmore Linux Users' Group (SLUG).

[9] `http://lists.debian.org/debian-user/`

[10] `http://lists.debian.org/debian-laptop/`

[11] http://lists.debian.org