

Painless Debian GNU/Linux

Stephen van Egmond

**Principal
Tiny Planet**

`svanegmond@tinyplanet.ca`

Painless Debian GNU/Linux

by Stephen van Egmond

Copyright © 2001 Stephen van Egmond

This article is a review of Debian GNU/Linux from the point of view of end-users, and serves as a gentle introduction into the pain-free installation and administration of the OS. The article leads the user through the installation of Debian GNU/Linux and ReiserFS, security updates, an upgrade to Woody, and the installation of X11, KDE, Koffice, and other applications. The reader will also learn some of the Zen of Debian, and gain an understanding of the routine maintenance that is required.

This article should be useful to a computer user who has some experience messing around with alternative operating systems like the BeOS. The article does not cover all Debian architectures, nor everything that could possibly go wrong, nor all the things you might need to think about. Only the stuff that's fairly common or really important.

The most recent version of this article is available on the Web (<http://tinyplanet.ca/pubs/debian/>).

This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, v1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>). Distribution of the work or derivative of the work in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder.

Table of Contents

1. Introduction	1
1.1. Why I wrote this article	1
2. Discovering Debian	3
2.1. What to expect	3
2.2. Structure and Interpretation of Debian	3
3. Installing Debian	5
3.1. Inspecting your system	5
3.2. Clearing out some space	5
3.3. Grab and burn the installation floppies	5
3.3.1. Burning installation floppies in BeOS	6
3.3.2. Burning installation floppies in Windows	6
3.4. Install the base system	6
3.5. Reboot and configure the base system	7
4. Securing and upgrading your system	9
4.1. Securing your system	9
4.2. Upgrading to Woody	9
4.2.1. Something is complaining about libdb2	10
4.2.2. /etc/exim/exim.conf: no such file or directory	10
5. Installing X11 and applications	12
5.1. Sound	12
6. Care and feeding	14
6.1. I want to mount my DOS partition	14
6.2. I want to make sure I'm running the latest version of everything	14
6.3. I need some software, but I don't know the name of the package	15
6.4. I want to (re)configure a package	15
6.5. I want to hack the source code of a package	15
6.6. I want to remove a package	16
6.7. I removed a package, but there's still stuff lying around	16
6.8. I [think I] found a bug!	16
7. I think I feel like rebuilding my kernel	18
7.1. Kernel configuration tips	18
7.2. It won't boot: Rescuing your linux system	19
8. Conclusion and further reading	20

Chapter 1. Introduction

When I was an undergraduate at the University of Waterloo, I still clung to my ancient, aging Amiga system. Once a thundering paragon of computing power and design, it had lately been reduced to playing pinball and connecting to the University's Unix systems to do assignments. It was 1996. I was a regular on the Amiga newsgroups, and I watched debates break out about the BeBox: was it the next Amiga? Did Amiga users, repelled by Windows, and discouraged by Apple's horrible hardware, finally have a new home? I was curious. PowerPC processors? Two of them? And more I/O ports than you could dream of using? Cool.

In late 1996, a poster went up around the Math building: Be was coming to town to show off the BeOS and the BeBox, on an invitation from the Computer Science Club. When I showed up to the auditorium, it was packed, and I had to sit on the stairs. Scott Patterson and William Adams were at the front, bouncing around, chatting with students, and giving sacrifices to the Demo Gods before their presentation.

At the time, I didn't know much about Silicon Valley's culture, and the prized status that the Demonstration holds for those who live there. The Demo is nearly everything that matters. If you don't have a good demo, or worse, your demo crashes, you're doomed. If you have a good demo, you have a shot at glory. Thus, Be has always had excellent demos that stuck with you, its visceral display of tech prowess resonating in your head for months afterward. Scott and William had a really, *really* good demo: funny banter, physical comedy ("oops! was that the power cord?"), pretty slides (produced and shown on the BeOS), and jaw-dropping video and audio effects produced live on the BeBox. Everyone there, perhaps 150 computer science nerds from one of the nerdiest schools in North America, was sold on the OS.

This close to graduation, I already had an employer lined up. I gave them a demo (imitating Scott and William), and convinced them that it was worth their trouble to pay me to port our image editing application to the BeOS. For 2 years after graduation, for two different employers, I got to work and play in the BeOS all day -- and got paid for it. Not bad at all. But events would eventually derail all my projects, and hints that something was seriously wrong began to show up in 1999, shortly after Be's IPO.

In March 2000, Be announced it was leaving the desktop OS market to focus on the Internet Appliance market. The response from the community was mostly denial, although lots of people made noises about abandoning ship. Through 2000 and 2001, while Be tuned its staffing and operations towards the BeIA platform, the community played cruel jokes on itself. BeShare and the news sites were filled with leaked betas, forged screen shots, incessant rumours, and inflated egos who claimed to be informed, but were almost always wrong.

And so we arrive at August 2001, when Be agreed to sell most of its assets to Palm Computing. Palm makes the successful, though increasingly baggage-laden and rudderless Palm OS and some hardware that runs it. I wish them well and look forward to the gadget lust their products will most likely inspire, when they appear. In the meantime, I have a technical life to live, a business to build, and things to do on the Net. I need a new OS to fill the gaps in the BeOS.

1.1. Why I wrote this article

It's becoming increasingly clear that the limitations of the publically-available BeOS are really beginning to chafe against the users. This article aims to:

1. fill the gaps -- graphics software, office software, and good browsers among others -- that BeOS users have been complaining about since the first release.
2. give BeOS users a new realm to explore, without wasting mountains of time exploring dead ends.
3. save people the grief of running Windows.

4. save people from getting their machines hacked because they installed Red Hat.

Chapter 2. Discovering Debian

Neal Stephenson's essay *In the Beginning was the Command Line* (<http://bang.dhs.org/be/beginning.html>) was, characteristically, right on the money. He characterized the BeOS as a Batmobile compared to Windows' colossal station wagon and MacOS's European sedan qualities. The Linux crew were building a tank: compelling and powerful, but scary and dangerous. If you weren't careful and didn't know what you were doing, you could easily run over your dog and punch a hole in your house's front wall.

Of particular interest in Stephenson's essay is his treatment of the moral aspects of the Debian GNU/Linux distribution. Debian is the only distribution to date to have a constitution (<http://www.debian.org/devel/constitution>), and a social contract (http://www.debian.org/social_contract) expressing its commitment to its users. Its leadership is elected by developers. Its developers are all volunteers. Its bug system (<http://bugs.debian.org/>), much like Be's, is a catalog of technical failure and redemption. Unlike Be's, you get to see a lot more of the sausage-factory aspects: developers get into debates and the occasional flame war, which the bug system records for posterity.

2.1. What to expect

On the plus side, Debian is an efficient, stable system which uses *reasonable* defaults for most things, and asks mostly sensible questions when it comes to configuration. The package management system is enough of a reason to use Debian over Red Hat and Mandrake-based systems. For instance, installing a package will automatically download and install any required packages. Removing a package will cause the packages that depend on that package to be removed. You can ask the system which package owns a given file. The close adherence to the Linux Filesystem Hierarchy Standard means you can probably guess, blindfolded, which directory a particular configuration or data file should be in.

On the down side, Debian is still Linux, which means that X11 ("The X Window System") is pretty well it for a GUI. X11 has one of the more silly cut-and-paste schemes ever invented, and its focus behaviour can occasionally be irritating. Unix enthusiasts might respond "But you can configure it any way you like" - to which I would say "You have missed the point; it should do something sane out of the box". Whatever. You'll find the recommended setup bearable, and will gradually learn to tweak it to your needs.

After following these instructions, you'll end up with a system running a modern kernel, effective graphics system, with a usable development environment, browsers, mailers, web servers, databases, and office-type applications. It should take you about 3 hours to be completely set up, if you have a high-speed Net connection.

Linux runs about as efficiently as BeOS on the same hardware, and with some effort, the user experience can be almost as good.

2.2. Structure and Interpretation of Debian

Debian, like any other OS, lets you install a vast amount of software. Most of the software you use will be bundled into packages, often termed deb archives or debs. A Debian archive is a collection of programs, libraries, manual pages, and so on, *plus* scripts that install and remove the package. For instance, if you install Netscape, these scripts set up file associations and put a link to Netscape in the system menus.

Debian packages also include a *control file* which describes the package, identifies the maintainer, and indicates which packages the package depends on and conflicts with. There's more in the control file, but we'll skip it for now.

If you put a bunch of Debian `.deb` files into a directory tree, and have a web or ftp server dish them out, you have yourself a *package source*. The Debian project operates a number of package sources, called *distributions*, which gradually solidify into a *release*.

Debian's releases are named after characters from Pixar's movie *Toy Story*. So far, they have released *buzz*, *rex*, *bo*, *hamm*, *slink*, *potato*, and soon they will ship *woody*. There is another tree, *sid*, which we will get to in a minute. In Debianspeak, there's two symbolic names: *stable* refers to the most recently released distribution, and *testing* refers to the tree which will, at some glorious point in the future, become the new stable tree. At the time of writing (August 2001), the current stable tree is *potato*, and the current testing tree is *woody*.

You, as a Debian user, get to choose which tree you want to use for your packages. You can also mix and match between trees, and even use sources from people or organizations other than Debian (this is what Corel did with their Debian-based distribution). We won't be doing any of that in this article, though: I recommend the testing tree for nerds, and the stable tree for people who can't stand any degree of brokenness or experimentation. We won't draw on any non-Debian package sources.

All packages begin life in the special Debian distribution *sid*, which is something of a mosh pit. When a package maintainer uploads a new package, or new release of a package, if it passes a few basic tests, it's stuck into the *sid* tree. The author probably tested it, but they may not have caught some bugs or conflicts with other packages. As such, using *sid* for your packages is a pretty brave choice, and not recommended unless you have lots of spare time, or plan on being a Debian developer.

If a package lasts a few weeks in *sid* without getting a severe bug filed against it, it's automatically moved into the *testing* distribution.

Eventually, all the important bugs are ironed out of the testing distribution, and it becomes the new stable distribution. Debian policy is to avoid unnecessary changes to the stable tree; only security improvements are allowed. This can result in some bizarre situations. For instance, the current stable release of Apache is version 1.3.9. If new releases come out that fix security problems, the security fixes will usually be backported to 1.3.9, producing package version 1.3.9-2, 1.3.9-3, and so on. It's odd, but it's correct, and better yet, it works.

Chapter 3. Installing Debian

This style of installing Debian is straightforward: bring up a core system that can manage packages and communicate on the Net, then download the rest. If you don't have a free Net connection, there are huge number of places you can buy a Debian CD from. Nearly everyone will like Cheap bytes (<http://www.cheapbytes.com>), who have 3-CD sets for USD\$10, and ship to Europe for USD\$8, and to North America for USD\$5. If you opt to use a Debian CD, you should still look through this chapter for various hints and tips, and certainly read the later chapters. If you have a CD burner, visit <http://cdimage.debian.org/> to download an ISO.

In these instructions, I assume you've got a fairly typical computer system: Pentium-class or better, with IDE storage, maybe an internal or external modem (but *not* winmodem), and a fairly typical sound card and network card.

3.1. Inspecting your system

Using an operating system already on your machine, inspect your system to identify the chipsets of the various devices that are installed. This shouldn't be anything new to BeOS users: hardware drivers are written for particular hardware chipsets, which may appear in a diverse range of products. For instance, it's not enough to say "I have a D-Link network card". You need to know which *model* it is, and what chipset is on the card.

Make sure to note your graphics hardware, audio hardware, and network or modem hardware. Note the parameters of your network connection: any DHCP or PPP settings, or if you're on a LAN, what your gateway, netmask, and DNS servers are.

If you have an ISA network or sound card, you will have to take note of its IRQ, IO, and optionally DMA settings. You can find them by looking around in any other OS, or perhaps by peeking at the jumper settings.

3.2. Clearing out some space

You need some hard drive space, at least a gigabyte. Some suggestions:

- Maybe you kept some free space aside while partitioning your disk?
- Choose a partition that you don't care about anymore; while installing Debian, you can nuke it.
- Install a new hard drive, to live alongside your existing one.
- Borrow a copy of Partition Magic from work.

3.3. Grab and burn the installation floppies

Note: If you are installing Debian from a CD, you don't need to do this, but you should know you won't be able to format your partitions with ReiserFS, and will have to use ext2. The difference between the two is that ReiserFS is journaling, and ext2 is not. Thus, with ReiserFS you don't have any long file system checks if your machine should be rudely shut down or rebooted.

The installation disks provide a kernel, a small set of utilities, and some device drivers. I recommend that you use the potato floppies which have *ReiserFS* included on them. From BeOS or Windows, download these files:

rescue.bin (<ftp://acs-mirror.ucsd.edu/linux/debian/reiserfs-boot/current/rescue.bin>) , root.bin (<ftp://acs-mirror.ucsd.edu/linux/debian/reiserfs-boot/current/root.bin>) and driver-1.bin (<ftp://acs-mirror.ucsd.edu/linux/debian/reiserfs-boot/current/driver-1.bin>) . Windows users will also need the rawrite (<http://www.tux.org/pub/dos/rawrite/rawrite.exe>) DOS application.

Make sure you keep these floppies; they double as rescue disks.

3.3.1. Burning installation floppies in BeOS

1. Format the floppy disks. Don't skimp on this. Using unformatted floppies has a 50% chance of failing pathetically. DriveSetup (located under the Preferences menu) can do the formatting.
2. Bring up a terminal and type `dd if=rescue.bin of=/dev/disk/floppy/raw bs=10240`
3. Repeat for root.bin and driver-1.bin

3.3.2. Burning installation floppies in Windows

1. Format the floppy disks. Don't skimp on this. Using unformatted floppies has a 50% chance of failing pathetically. Just `format a:` from the command line will be fine.
2. Move the .bin files and rawrite.exe to somewhere convenient, like C:\
3. Reboot into DOS. Don't argue, just do it.
4. Run `c:\rawrite.exe`, and follow the prompts to burn the floppies.

3.4. Install the base system

In the interest of speed, I'll focus on the interesting or error-prone parts of the installation procedure. The full, boring details are available on the Debian website (<http://www.debian.org/releases/stable/i386/ch-init-config.en.html>) .

1. Put the first (rescue.bin) floppy or CD in and boot from it.
2. Create a swap partition. Choose a size of 128 mb if you have basic needs, 192 mb if you expect to be busy, and 512 mb if you just have too much space to spare.
3. When creating a Linux partition, tell the system to make a *ReiserFS* partition.

Tip: Unix system administrators have frequent religious wars over the appropriate number and size of partitions to create. One big partition will probably be enough for most uses. If your system will operate as an unattended server, you may want separate partitions for /home, /var, and /tmp. See the Linux partition HOWTO (http://www.ibiblio.org/pub/Linux/docs/HOWTO/mini/other-formats/html_single/Partition.html#NUMBER) for more information.

4. The floppies have some common (notably 3com and some Adaptec) drivers built right into the Kernel, and can optionally load in any of dozens of other network or SCSI drivers. When offered the chance, you should look through the driver list and make sure your devices are installed. If you install a driver, the installation

should complete silently, except for maybe a slight banner message from the driver. If you see errors from `modprobe`, then your driver didn't load. You either chose the wrong driver, or your driver needs some parameters you didn't give it.

If you choose a wrong driver, nothing bad will happen. So don't be afraid to try a driver you think might be applicable to you.

Make sure to add the drivers for your sound card if you have one.

Note: People with ISA cards will have to specify the IO, IRQ, and maybe DMA address for their cards when they load the driver module.

Note: You must have a working network driver to configure the network connection and complete your installation.

5. When configuring the network, make sure to re-enter the information you noted in Section 3.1> .
6. When you are asked how you want to install the base system and Kernel, you'll have choices like "first floppy drive", "CD-ROM", "partition on the hard disk", or "network". Choose the network install, accept the defaults, and let your system download and install about 15 megs of stuff.
7. Pay close attention while choosing how to make Debian bootable from the hard disk: Debian's boot manager is as good as any other, so it's acceptable to write the Debian boot record into the MBR. This also makes installation easier, since you won't have to manually add Debian into your old boot manager to finish up installation.

3.5. Reboot and configure the base system

The installation system will eventually finish its first set of questions and reboot. Once rebooted, it will ask you yet more questions about what kind of software you want to install and where you want to get it from:

1. Use contrib software? Yes.
2. Use non-free software? Yes, unless you're political about your software licences.
3. Use non-US software? Yes, if you don't mind breaking US laws or patents.
4. Create a user account besides root? Of course.
5. Simple or advanced package selection? *Simple*, or else you will lose 3 hours of your life.
6. In simple package selection, which package sets do you want to install? Probably none, unless you're absolutely sure you're going to be using it. These "task" packages are just a little bit beyond comprehensive. I recommend (and will assume) you choose nothing at this point, and opt to install packages later.
7. Which package source to use? `http`. Specify your country from the list of mirrors.

Once you've chosen what to install, Debian will install and configure everything. At a minimum, it will ask you questions about how to route mail using `exim`, the built-in mail transport agent. `Exim` is like `sendmail`, only it's not written by aliens. Most people will want to pick option 2 or 3 while configuring `exim`. If you get the questions wrong, or your mail doesn't work, edit `/etc/exim/exim.conf` to adjust things, or run `eximconfig`.

At this point, you should be able to log into your fresh, new Debian system, edit files with "vi", and not much else.

Congratulations!

Chapter 4. Securing and upgrading your system

If you're running Potato, your system is running a set of packages that were current in mid-2000. For many people that's fine. But for many, it's not, and this chapter will walk you through upgrading to the Woody distribution. Here, we do the upgrade in two small steps: first, by adding the security-updates distribution, and next, by adding the woody distribution.

This section makes reference to a text editor. The basic Debian install includes `vi` and, for those who don't know `vi`, a visual editor called `ae` which is good enough for manipulating configuration files and is easy to use.

4.1. Securing your system

1. Log in as root.
2. Edit `/etc/apt/sources.list`
3. Add the line

```
deb http://security.debian.org potato/updates
      main contrib non-free
```

to the end of the file, then save it.
4. Type `apt-get update`. Watch while it pulls down a new list of packages.
5. Type `apt-get -u upgrade`. It will indicate which packages are about to be upgraded. Say Y. Watch the pretty upgrade process.
6. This step is optional, but a good idea if your system is always connected to the Net. Type `apt-get install harden-servers harden-clients harden-localflaws`.

“What just happened?” you might be asking. In step 3, you told the Debian package management system that there is a *new* source of packages out there in addition to the ones already defined. In step 4, you downloaded an updated list of packages from the sources listed in `/etc/apt/sources.list`. After step 4 completes, your system can now perform upgrades and package installations from the new source, which is exactly what you do in step 5.

The final step, the installation of the 'harden' series of packages, is a useful aspect of security, though it doesn't (directly) do anything. The hardening packages have intentional *package conflicts* with known non-secure packages. This means that the Debian packaging system won't let both be installed. Thus, when you do something like install a telnet daemon, you'll be told:

```
# apt-get install telnetd [...]
The following packages will be REMOVED:
harden-servers
The following NEW packages will be installed:
telnetd
```

... which should set off alarms in your head, and make you reconsider your installation.

There. Your system is now running the latest security patches, on the Potato tree. It's not an impenetrable fortress of hard-core cryptographic security, but it will stand up to more far more unwanted attention than your average Red Hat system.

4.2. Upgrading to Woody

This procedure, once started, is mostly automatic, although you can't just start it and walk away. Woody is, after all, the 'testing' distribution: if it was perfect, it would be the 'stable' distribution.. In the next section, we outline some of the problems people have seen while upgrading to woody.

1. Log in as root
2. Edit `/etc/apt/sources.list`
3. Copy and paste the lines for potato, renaming them to Woody, except for the security line. (There is no security source for the Woody distribution: Woody is only automatic updates from sid.) For instance, if you had something like this in your `sources.list`:

```
# potato
deb http://ftp.ca.debian.org/debian/ potato main non-free contrib
deb http://non-us.debian.org/debian-non-US potato non-US/main non-US/contrib non-US/non-free
deb http://security.debian.org potato/updates main contrib non-free
```

You would add these two lines:

```
# woody
deb http://ftp.ca.debian.org/debian/ woody main non-free contrib
deb http://non-us.debian.org/debian-non-US woody non-US/main non-US/contrib non-US/non-free
```

4. Type **`apt-get update`**
5. Type **`apt-get -u dist-upgrade`**
6. Go for a walk while everything downloads.
7. Come back to find that something bombed during the upgrade.
8. Maybe try something to fix it, then type **`apt-get -u dist-upgrade`** again. Repeat this step until it completes without errors.

“What’s a dist-upgrade?” A normal upgrade operation can’t install new packages or remove existing ones. Only new versions of existing packages will be installed. The dist-upgrade operation signals to the package management system that it’s okay to remove or install something to complete the upgrade. This is almost always required when switching distributions, hence the name dist-upgrade.

“What? You expect something to bomb?” Yes. This is the testing distribution, so stuff is (by definition) still not all working right. The following Q&A catalogs some of the things I’ve found broken during an upgrade, and how I fixed them.

If you encounter a problem during an upgrade that’s not listed here, first consult with the Debian bug system (<http://bugs.debian.org/>) by searching for bugs filed against the problem package. Try some things. Ask on IRC. File a bug. If you come across a solution, and know it happens for nearly everyone one, you can e-mail the problem (and solution) to svanegmond@tinyplanet.ca and I’ll add it to this document.

4.2.1. Something is complaining about libdb2

You need to force its installation, since something has (for some reason) decided not to install it first. Type **`cd /var/cache/apt/archives/`** followed by **`dpkg -i libdb*.deb`** Then resume the dist-upgrade as described above.

4.2.2. /etc/exim/exim.conf: no such file or directory

This a bug that should be fixed in exim RSN. Type `mkdir /etc/exim/` then resume the dist-upgrade as described above.

Chapter 5. Installing X11 and applications

You're now going to install a fairly hefty set of packages, which I recommend as a core set of useful tools to get around on the Net.

1. `apt-get install task-x-window-system-core` will bring in the basic X-Window system and fonts.
2. `apt-get install xserver-xfree86` will install the video drivers for your video card, and configure them. During installation, you will go through an intensive question and answer session where you describe your video hardware.

Tip: While choosing a video driver, please note that `mga` refers to Matrox video cards, and `tdfx` refers to 3dfx Voodoo cards.

Tip: While choosing a mouse, if you have a PS/2 mouse, you want to choose the `psaux` mouse device. You probably want to pick the first mouse protocol it offers you.

Tip: It's worth looking up your monitor's specifications so you can go through the "Advanced" refresh rate configuration. You'll get a better picture. If you don't have the information, though, even "simple" will do. You can reconfigure later by typing `dpkg-reconfigure xserver-xfree86`.

3. `apt-get install kdebase konsole kdm` will install the KDE (K Desktop Environment) program, console application (like BeOS's Terminal), and login prompter.
4. `apt-get install konqueror netscape links lynx ssh mutt kmail licq kaim xchat xmms freeamp gimp qiv gs koffice` will install several web, mail, IRC, ICQ, and AOL Messenger clients, along with two mp3 players, a graphics editor, graphics viewer, Postscript/PDF reader, and a complete office suite with spreadsheet, word processor, slide presenter, illustration program, etc. BeOS users are permitted to be appalled at how easy this step was.

5.1. Sound

These are the joys of open-source operating systems: you have 3 or 4 different implementations of the Media Kit to choose from. The big ones include esound, ALSA, libarts, and OSS. Esound and OSS appear to be the most mature and supported by the most applications.

There is a set of drivers which are identified as OSS drivers; if you compile or load one into your Kernel, then OSS-aware applications (such as xmms) will just work. If your driver is not an OSS driver (not all have been rewritten), then you will probably have to use esound.

To bring up esound, you need to ensure your sound card driver is loaded. If you identified the sound driver while installing the operating system, it should be. Type `lsmod` as root and see if it's listed in the modules the kernel has loaded in. If not, add its name to `/etc/modules.conf` or consider rebuilding your kernel to compile it right in, as described in Chapter 7>.

If you're sure you have your network driver loaded, switch to root, and type `apt-get install esound`. This will download and install the Linux equivalent of the Media Kit. Start it by typing `esd` as root. Your esound

applications should work now, though in some cases (like xmms), you need to go into the configuration and *tell* it to use the esound output driver.

Chapter 6. Care and feeding

Debian is an amazingly low-maintenance system. Most packages will install automatic log file rotation, compression, and deletion. The database packages (PostgreSQL in particular) will configure the system to do nightly maintenance. If you install the `tmpreaper` package, your `/tmp` directory will be cleaned nightly.

The only procedure you need to do routinely - about once a week - is to upgrade your system to the latest package versions. Think of this as hygiene - brushing your teeth. If you let it go for too long, you could end up with problems. Doing regular upgrades big-bang upgrades, which tend to break somewhere in the middle and need some careful attention. Lots of small upgrades seem to go better.

Getting around your new Debian system will take some getting used to. The most important step is understanding the layout of your directories: the Filesystem Hierarchy Standard (<http://www.pathname.com/fhs/>) is a good place to start. Here's the important bits:

- `/etc` gets configuration files which almost never need to change. The rules for your mail server, your network configuration, and which package sources you're using are here. Configuration files are in `/etc/package.conf` or the `/etc/package/` directory.
- `/var` gets data files which change routinely. Databases, mail, and web servers keep their stuff here. System logs are put into `/var/log`. Backups of various files go into `/var/backups`. Data files are in `/var/lib/package/` directory, usually.
- `/bin` and `/usr/bin/` get applications users need and want to run (respectively).
- `/sbin` and `/usr/sbin/` get applications superusers need and want to run (respectively).
- `/lib` and `/usr/lib` has handy libraries used for operating the system (`/lib`), and running applications (`/usr/lib`). Perl modules get installed into `/usr/lib/perl5`.
- Software that you compiled yourself (ewwwwwwww - do you have to?) goes into `/usr/local/`. If it insists on installing itself into some other directory (like `/usr/bin`), throw it out. You didn't want it anyway.

Finally, there are some common tasks that people will want to do sooner or later:

6.1. I want to mount my DOS partition

1. Log in as root, then add this line to your `/etc/fstab`:

```
/dev/hda1 /dos vfat defaults 0 0
```

Those parameters are: the partition where the DOS partition is, where in your filesystem to put it, which filesystem driver to use, a logging flag, and a flag indicating whether to automatically mount that partition on boot.

2. Type `mkdir /dos` to give the filesystem some place to put your file system.
3. Type `mount /dos` to mount the filesystem.

6.2. I want to make sure I'm running the latest version of everything

1. Assuming you're logged in under your regular user account, type `su -`. Type your root password. You're now the superuser.

2. Type **apt-get update**. This will pull down the latest lists of packages. It won't change any software currently installed -- just tell the packaging system what's new.
3. Type **apt-get -u upgrade**. This will download the latest versions of whatever you've got installed, install and configure them.
4. Type **exit** to go back to your user account.

Done! If upgrading would require that some packages be removed or new ones be installed, it will be marked as "held back". These packages will be upgraded if you do an **apt-get -u dist-upgrade**

6.3. I need some software, but I don't know the name of the package

1. **su -**. Type in your root password.
2. Type **apt-cache search searchterm** where searchterm is some word that describes what you're looking for. For instance, if you're looking for a SETI@Home client, type "seti". If you want a PDF reader, try "pdf".

If your first search returns a huge list of results, put them through one or more grep steps: **apt-cache search mail | grep client**

3. If you want to look at a package to see in detail what it does, type **apt-cache show packagename**.
4. Once you've found a package you want, install it with **apt-get install packagename**.
5. Type **exit** to go back to your user account.

6.4. I want to (re)configure a package

As it is with software everywhere, this part is kind of lumpy and uneven, and takes some guesswork.

- Many packages come with automatic configuration scripts, which you can summon with **dpkg-reconfigure packagename**. This works, for instance, with `xserver-xfree86`, which holds your monitor and video card configuration.
- Some packages come with configuration tools which you have to run manually, sometimes (but not always) called `[packagename]config`. For instance, the `exim` package includes a configuration tool called **eximconfig**. To find out which files were installed by a package, type **dpkg -L packagename | more**. Look for packages installed into `/usr/sbin/`, since those are only usually run by the superuser.
- Finally, the majority of packages leave some kind of configuration files in `/etc`, either a file called `/etc/package.conf`, or a number of files in a directory called `/etc/packagename/`. Go there and look around.

6.5. I want to hack the source code of a package

Before building anything, you need to install the **build-essential** package, which will ensure that you have the appropriate compilers and tools. While building a package, you need to be logged in as root.

1. `apt-get build-dep package`, where *package* is the package you intend to compile from source. This will make sure you have all the required stuff to compile *that* package. For instance, to compile an image manipulation program, you probably need the header files for for the JPEG library. This will make sure they're installed.
2. `apt-get source package`, where *package* is the package you intend to compile from source. This downloads the source code and expands it into the current directory.
3. 'cd' into the source directory, and hack around. The `debian` directory thereunder has Debian-specific patches and installation scripts. When you're ready to compile, type `dpkg-buildpackage` in the source directory. The build should complete, and you should end up with a `.deb` file in the parent directory of the source directory.
4. `dpkg -i ../*.deb` should install the `.deb` file that you built.

6.6. I want to remove a package

1. Log in as root.
2. `apt-get remove packagename`
3. If you don't know the package name, type `dpkg -l` to see a list of packages known to the system and look for it.
4. If you still can't figure out the package name, you might be able to identify it by searching for a file that belongs in that package with `dpkg -S`. For instance, if you want to get rid of `/usr/bin/wget`, type `dpkg -S /usr/bin/wget` to see which package owns that file.

6.7. I removed a package, but there's still stuff lying around

Whether a package is removed turns out to be more a question of degree rather than the yes/no you might have hoped. When packages are removed, their configuration and occasionally their data files are left behind. Really, it's a feature: This lets you reinstall them, and get the same configuration back again. For instance, if you `apt-get remove exim` then `apt-get install exim`, you won't have to reconfigure the package again: your previous answers will stand. Remove; install. Remove; install. Remove; install. Fun!

In practice, this usually means one or two files are left lying around in the `/etc` or `/var` directories. If this few kilobytes of stuff really offends you, you can remove it with `dpkg --purge packagename` to finally, and for-real, remove anything the package left behind.

6.8. I [think I] found a bug!

Great! Bugs happen, and don't let anyone try to convince you otherwise. There's several things you can do.

The most likely to achieve success, is to visit the Debian Bug System (<http://bugs.debian.org/>) and look around to see if your problem is already known. If you search by the name of the package giving you trouble, the bug system will show you recently-opened and -closed bugs. One of them may be applicable. By reading the bug details, you may discover a work-around or solution.

Other approaches that might work include looking at the software package's home page. Most packages come from outside Debian and are repackaged by Debian developers. Find the home page through Google, and look around to see if there's a list of known bugs. Be sure to try Google Groups, too.

When all else fails, Debian lets you Use The Source. See Section 6.5> .

If you've really found a bug that nobody else has found, follow the Debian bug submission rules available on the bug database pages.

Chapter 7. I think I feel like rebuilding my kernel

Just so you know, the first time you do this, it will take you 2 hours to get it right, and you may get it wrong a few times at first. Section 7.2 explains how to rescue your system.

Note: Note that the approach I'm recommending is not the True Debian Way. The Debian Way is to install and use the `kernel-package` package, which lets you build a `.deb` with a kernel image and modules. It is worth investigating, but I can't recommend it since it requires some fussing with `/etc/lilo.conf` that just isn't worth going into.

1. Log in as root, and do `apt-get install gcc make libncurses5-dev bin86 ncftp bzip2 kernel-package`. Those packages are needed to do a kernel build.
2. If you've already got some kernel source lying around, type `rm -rf linux` to get rid of it.

Note: If you do frequent kernel builds, this is suboptimal: learn how to patch kernels from release to release, and you'll save lots of downloading time and effort spent redoing the same configuration over and over.

3. `ncftp ftp.countrycode.kernel.org`, substituting your country's code. United States = us, Canada = ca, United Kingdom = uk, etc. You should be automatically logged in anonymously.
4. `ncftp` has tab-filename completion, so use it. Type `cd pub/linux/kernel/v2.4`. This assumes that 2.4 is the latest stable Kernel series, which it will be until at least 2003. Identify and download the latest kernel source file, which is called `linux-2.4.x.tar.bz2` for some value of x.
5. Once the kernel source is downloaded, quit `ncftp` and type `bzip2 -dc linux-2.4.x.tar.bz2 | tar xv`. This will expand the source under a "linux" directory.
6. `cd linux`
7. `make menuconfig`. Refer to the next section to learn how to configure your Linux Kernel build.
8. Once you've left the configuration system, type `make dep && make bzlilo && make modules && make modules_install && reboot`. Go for a long walk, or wait for some water to boil. When you come back, if it worked, you'll be running a new kernel. If you're unlucky, you'll see whatever caused the compilation to break. The command line is structured so that if something dies, the remaining make targets won't be built. If you're very unlucky, you'll see LI on the boot loader screen, two letters you will come to hate.
9. If, indeed, your machine gets stuck on boot with LI just after the boot loader, something probably bombed in the kernel build. To rescue your system, press the reset button, and when the MBR prompt comes up, press the left SHIFT key or TAB until you get its attention. It's now asking you which operating system you want to load. Say `LinuxOLD` to load the previously-working kernel. Once you're in, do a `make clean` in the `linux` source directory, then do each of the above "make" steps (`dep`, `bzlilo`, ...) one at a time and watch what they have to say.
10. If you think you were successful, type `uname -a` on the command line to be sure you're running the new Kernel.

7.1. Kernel configuration tips

When you type **make menuconfig**, you will be slapped in the face with a long, scary list of configuration options. And that's just the top level; the tree can go quite deep. These are the ones to pay attention to.

Note: Linux lets you compile things directly into the kernel, or as add-ons which can be loaded once the system is running. Infrequently used items should be left as add-ons, to reduce the size of your kernel and free up memory. To choose to compile something into the kernel, press Y. To choose to compile something as a module, press M. To turn off a setting, press N.

- Under Code maturity level options, turn on "prompt for experimental". Even though it's quite stable, ReiserFS is still marked experimental.
- Under Processor Type, make sure you've got the right type selected, If you're not sure what CPU type you've got, choose something conservative. Maybe type **cat /proc/cpuinfo** . Turn off SMP if you don't have two CPUs.
- General setup: Turn off PCMCIA if you don't have a laptop. Turn off support for a.out and MISC-format binaries. Turn on support for power management, and software power-off if your hardware supports it.
- Parallel support: Only if you have a parallel printer.
- SCSI support: Turn it off, unless you have SCSI (build it into the kernel) or need to use a USB mass storage device (build it as a module).
- Network device support: make sure your network card driver is being built into the kernel (or as a module).
- Character devices: look carefully at the AGP and Direct Rendering options, and make sure your video hardware is selected.
- Multimedia: your bt848 video capture cards and USB webcams are supported here.
- Under file systems, turn on (all the way on, not as a module) ReiserFS. Turn on the DOS (and vfat) options if you have a FAT partition, perhaps as modules. Turn on SMB to mount Windows shares.
- Sound: turn on sound, and make sure your sound card chipset is selected. Module or built-in.
- USB: Only if you use it.

7.2. It won't boot: Rescuing your linux system

1. Insert the first floppy you created in Section 3.3, then boot from it. Insert the second one when prompted.
2. Select 'Mount a previously initialized swap partition'.
3. Select 'Mount a previously formatted Linux partition'. Repeat for all your Linux /var, /home, etc. partitions.
4. Press ALT-F2 to get a new console.
5. Type **chroot /target** . You have just changed Linux's concept of where your root directory ("/") is, from the floppy to your hard drive.
6. Fix your system. Type **reboot** and remove the floppies when you're done.

Chapter 8. Conclusion and further reading

Linux (and X11 desktops in particular) can be painful to use, but the wealth of applications and enthusiasm of its community can make it sometimes seem worthwhile. Open Source software has numerous pragmatic advantages, one of the most important being the reduction of technology risk, something many, many BeOS users can relate to lately. Linux will not suffer a business crisis, run out of money, or suddenly change its direction. It will always do what its users want it to, which at the moment seems to be the *best* Internet server operating system, that runs on even modest hardware, and has a certain - perhaps reasonable - level of usability.

A good place to start learning about Debian is Debian Planet (<http://www.debianplanet.org/>). If you're interested in a healthy, non-rabid Linux/software/nerd community check out k5 (<http://www.kuro5hin.org/>) or advogato (<http://www.advogato.org/>).

Linux won't be your last operating system. Until (and if) the BeOS makes its comeback with its new owners, it can certainly give us what we all really need: which is browsers, communication tools, office apps, and of course, a good game of quake: **apt-get install quake-lib quake-x11**

Happy hacking!