Table of Contents

Acknowledgments

AU: could you supply a couple of sentences here--see Wireless Hacks, 2ed for an example

Jonathan

The biggest thanks definitely have to go to my wife Ann and our children Amelia and Thomas, who for several months barely saw me from one week to the next. Writing Ubuntu Hacks has been one of those periods when everything else, including sleep, became secondary to just getting the job done and my family were amazingly supportive and understanding through everything.

Thanks also to my co-authors Kyle Rankin and Bill Childers who so willingly shared their knowledge of all things Linux, and the contributing writers who put in a big effort to supplement the body of the text with their particular areas of expertise. And the whole Ubuntu Hacks circus wouldn't have been possible without our editor, Brian Jepson, acting as ringmaster and keeping all our performances on schedule while even managing to contribute some of his own.

Finally, without the Canonical team there wouldn't be an Ubuntu to hack on, and without Debian there would never have been Canonical, and without the whole Free/Open Source Software community there would never have been Debian so the ultimate thanks have to go to the amazing community that we're all part of. To every person who has ever written Open Source software, or submitted a bug report, or written a how-to, or maintained a Debian package, or stood on a street corner and handed out Ubuntu CDs: Thank you. This book is written in your honour.

Kyle

First I'd like to thank my wife Joy for helping me yet again through the crunch period of this book. I'd also like to thank David Brickner for bringing me on this project along with Brian Jepson for his guidance in editing the book.

This book was the result of a great team effort so many thanks to Bill and Jon for all their hard work to make the book happen and thanks to all the contributing writers.

Finally I'd like to thank Ubuntu's amazing community of users and developers for their hard work in making Ubuntu a success in such a short amount of time.

Bill

I want to kick off the acknowledgments section by thanking and recognizing the most important people in my life—my family. Gillian and Conner, this book is for you. This is why Daddy's been at the keyboard for so many nights and weekends. Special thanks to Kelly for putting up with me while I undertook this project amongst all the other things I do—I love you, honey. You've been a tremendous influence and source of support, and I couldn't have pulled any of this off without you.

Thanks to my parents and grandparents for getting me my first computer and supporting my initial "addiction"—I wouldn't be where I am today without them.

Thanks to all the programmers, documentation people, bug testers, and everyone else who contributes to the Ubuntu and Debian projects. Every one of you should be proud of your work—you've created something truly special. Thanks to the crowd of `#linux` too, particularly

Jorge, whose ongoing pursuit of shiny stuff led me to run the prerelease of Warty way back when. Also thanks to my fellow writer Kyle—all the stuff we've been through has been a total blast, and I'm looking forward to the future.

Finally, thanks to David Brickner and Brian Jepson for giving me this shot and for editing all my mistaeks. [sic]

Preface

The first release of Ubuntu, the Warty Warthog, was made available to the world on 20 October 2004. Less than two years later, Ubuntu is now the number-one most popular Linux version at DistroWatch.com, far ahead of the distribution in second place. Countless articles, reviews, and blog postings have been written about Ubuntu and its sister distros, Kubuntu and Edubuntu. In Macedonia, Ubuntu will be installed in 468 schools and 182 computer labs. In South Africa, HP is going to offer desktops and notebooks with Ubuntu on them. Around the world, hundreds of thousands of people have installed Ubuntu, and in many cases, it was the first Linux distro they'd ever tried. For many of those new Linux users, Ubuntu has been so good that they've switched to Linux. For a Linux distro that's still an infant, this is remarkable stuff!

Why has Ubuntu been so successful? Technically, it's based on Debian, which is an excellent foundation for a Linux distro, but Ubuntu has added a level of finish and polish that has made it a joy to use for newbies, while still a powerhouse for more experienced users. It's incredibly up-to-date; a team of dedicated developers ensures that everything "just works," with regular updates to the various packages that make up the distro and a roughly six-month release schedule between distros.

But the secret behind the phenomenal success and growth of Ubuntu is really one man: South African Mark Shuttleworth. After founding Thawte, a company providing digital certificates, when he was 22, Shuttleworth sold the company four years later to VeriSign for a large amount of money. After fulfilling his dream of going into space, he decided to fulfill another and build the best Linux distro in the world. In that he has succeeded.

But it's also about principles with Shuttleworth. He has plenty of money, and he wants to do things with his fortune that will change the world. Consequently, Ubuntu will always aim for the highest quality, and it will always be free. The name *Ubuntu* itself is laden with meaning, as it is an African word meaning both "humanity to others" and "I am what I am because of who we all are," while *Kubuntu* means "towards humanity." Shuttleworth has promulgated the Ubuntu Code of Conduct, which states that members of the community must practice consideration, respect, and collaboration.

This is a book written by passionate Ubuntu and Kubuntu users who are excited to talk about a powerful, cool distro that meets the needs of novice, intermediate, and experienced users in a wide variety of ways. The hacks in this book cover the essential areas of Ubuntu, and they'll help you maximize your use of the distro. Whether you want to play music and movies, or use Ubuntu on your laptop as you travel, or install just about any software package you could ever want, or run other operating systems inside Ubuntu, we've got it all covered.

We know you'll get a lot out of Ubuntu Hacks, but we also want to encourage you to give back to the community and help grow Ubuntu:

Visit the main Ubuntu and Kubuntu web sites, at http://www.ubuntu.com and http://www.kubuntu.org. The entire sites are worth exploring in depth, but the Wikis especially offer a wealth of useful information.

Download Ubuntu and offer it to friends, family, and acquaintances. Heck, offer it to total strangers! The more people who try Ubuntu, the more people who will use Ubuntu.

If you don't want to download the distro, you can request free CDs at https://shipit.ubuntu.com. Don't be shy—ask and ye shall receive!

If you know how to program, consider becoming a Ubuntu developer. If you don't know how to program, there's still plenty of work you can do. Either way, head over to http://www.ubuntu.com/developers. And if you think you have the right stuff, you can even apply for work at http://www.ubuntu.com/employment.

Buy some Ubuntu swag from the Ubuntu Shop (http://www.cafepress.com/ubuntushop/), or donate money at http://www.ubuntu.com/donations.

Most importantly, tell the world about Ubuntu! Let's get the word out: there's an awesome, free, super-powerful operating system that anyone can use named Ubuntu, and it's made for you.

Why Ubuntu Hacks?

The term *hacking* has a bad reputation in the press. They use it to refer to people who break into systems or wreak havoc with computers as their weapon. Among people who write code, though, the term *hack* refers to a "quick-and-dirty" solution to a problem, or a clever way to get something done. And the term *hacker* is taken very much as a compliment, referring to someone as being *creative*, having the technical chops to get things done. The Hacks series is an attempt to reclaim the word, document the good ways people are hacking, and pass the hacker ethic of creative participation on to the uninitiated. Seeing how others approach systems and problems is often the quickest way to learn about a new technology.

How to Use This Book

You can read this book from cover to cover if you like, but each hack stands on its own, so feel free to browse and jump to the different sections that interest you most. If there's a prerequisite you need to know about, a cross-reference will guide you to the right hack.

How This Book is Organized

This book is divided into ten chapters, organized by subject:

Chapter 1, Getting Started

> This chapter shows you how to get started with Ubuntu. Whether you want to give it a whirl with a live CD, or you're ready to jump right in and install Ubuntu on your computer, you'll find what you need here. In addition to getting all the information you need to install Ubuntu on your system, you'll also learn how to get started with the Linux command line, set up your printer, file a bug report, and more.

Chapter 2, The Linux Desktop

> You're going to spend a lot of time in front of a mouse, keyboard, and monitor, working with one of the Linux desktops. This chapter helps you get the most out of the GNOME and KDE desktop environments for Linux, and even helps you find out about a few others that are worth checking out. You'll also learn such things as how to get Java set up, how to work with files on remote computers, and how to get Ubuntu talking to handheld computers.

Chapter 3, Multimedia

> This chapter gets the music and movies running so you can have some fun in between all the work you get done with Ubuntu. You'll learn how to play nearly any kind of audio and video, and burn files, music, and movies to optical discs.

Chapter 4, Mobile Ubuntu

> If you're using Ubuntu on a notebook computer, you're probably going to want to cut a few wires. This chapter helps you get going with various wireless cards. You'll also learn how to get the most out of your laptop, from saving energy to installing add-on cards.

Chapter 5, X11

> This chapter shows you how to tweak X11, the windowing system that lurks beneath the shiny veneer of KDE and GNOME. You'll learn how to get your mouse and keyboard working just right, and also how to get X11 configured so it takes full advantage of the graphics adapter in your computer.

Chapter 6, Package Management

To some extent, any Linux distribution is a big collection of packages held together by a whole lot of interesting and useful glue. Ubuntu's great advantage is the quality of those packings and all the testing and improvement that goes into them. This chapter shows you how to work with packages, whether you're installing them, finding new ones from beyond the edges of the Ubuntu universe, or creating your own.

Chapter 7, Security

This chapter shows you how to tighten up security on your system. You'll learn the basics of how the *sudo* command keeps you and your fellow users out of trouble, find out how to protect your network from intruders, and even keep your data safe if one of the bad guys does make it in.

Chapter 8, Administration

Every now and then, you're going to have to take a break from the fun of using Ubuntu and do some administrative tasks. Whether you're adding a new user, tweaking your system's configuration, or doing those backups you should have done long ago, you'll find what you need in this chapter.

Chapter 9, Virtualization and Emulation

This chapter shows you how to run Ubuntu inside of other operating systems, and other operating systems inside of Ubuntu. It's all made possible by a combination of emulation and virtualization, which effectively lets you run a computer inside of a computer.

Chapter 10, Small Office/Home Office Server

Ubuntu isn't just a great desktop operating system; it also makes a fantastic basis for a server. In this chapter, you'll learn everything from doing a basic server install to installing network services such as DNS, mail, Apache, and more.

Conventions Used in This Book

The following is a list of the typographical conventions used in this book:

Italic

Used for emphasis and new terms where they are defined, as well as to indicate Unix utilities, URLs, filenames, filename extensions, and directory/folder names. For example, a path in the filesystem will appear as */usr/local*.

`Constant width`

Used to show code examples, the contents of files, and console output, as well as the names of variables, commands, and other code excerpts.

`Constant`

`width`

`bold`

Used to highlight portions of code, either for emphasis or to indicate text that should be typed by the user.

`Constant width italic`

Used in code examples to show sample text to be replaced with your own values.

Gray type

Gray text is used to indicate a cross-reference within the text.

A carriage-return icon is used to indicate a line of code that carries over to the following line because of space limitations. These lines should be entered on one line at the command prompt.

You should pay special attention to notes set apart from the text with the following icons:

This is a tip, suggestion, or general note. It contains useful supplementary information about the topic at hand.

This is a warning or note of caution, often indicating that your money or your privacy might be at risk.

The thermometer icons, found next to each hack, indicate the relative complexity of the hack:

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

ED: Please let me know what the correct author order is

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Ubuntu Hacks* by Bill Childers, Jonathan Oxer, and Kyle Rankin. Copyright 2006 O'Reilly Media, Inc., 0-596-52720-9."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

Safari Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at http://safari.oreilly.com.

How to Contact Us

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). As a reader of this book, you can help us to improve future editions by sending us your feedback. Please let us know about any errors, inaccuracies, bugs, misleading or confusing statements, and typos that you find anywhere in this book.

Please also let us know what we can do to make this book more useful to you. We take your comments seriously and will try to incorporate reasonable suggestions into future editions. You can write to us at:

O'Reilly Media, Inc.
1005 Gravenstein Hwy N.
Sebastopol, CA 95472
(800) 998-9938 (in the U.S. or Canada)
(707) 829-0515 (international/local) (707) 829-0104 (fax)

To ask technical questions or to comment on the book, send email to:

bookquestions@oreilly.com

The web site for Ubuntu Hacks lists examples, errata, and plans for future editions. You can find this page at:

http://www.oreilly.com/catalog/ubuntuhks/

For more information about this book and others, see the O'Reilly web site:

http://www.oreilly.com

Got a Hack?

To explore Hacks books online or to contribute a hack for future titles, visit:

http://hacks.oreilly.com

Chapter 1. Getting Started

An operating system takes some getting used to. Whether you are new to Ubuntu or new to Linux itself, there are some basic things you need to get familiar with before you can move on. The hacks in this chapter cover those basics and then some.

The Ubuntu Live CD is a good way to explore Ubuntu without changing anything on your hard drive. This chapter explains how to get up and running with the Live CD, and even shows you how to use it with a memory stick to keep your settings and documents around between reboots. You'll also learn how to install Ubuntu, whether you want to make it the sole operating system on your computer or want to dual-boot between Ubuntu and Windows. You'll even learn how to install Ubuntu on a Macintosh.

This chapter also covers fundamentals such as getting your printer set up, getting help (and helping Ubuntu by submitting bug reports), getting started with the command line, and finding the most important applications you need to start "doing stuff" with Ubuntu.

Hack 1. Test-Drive Ubuntu

Use the Ubuntu Live CD to get to know Linux before installing it on your system. This is simply the fastest and safest way to try out Linux.

Though Linux on the desktop looks and behaves a lot like Windows, the simple fact is it isn't. Your favorite Windows programs probably won't run in Linux, it may be difficult to migrate data from your Windows install [Hack #7], and the years you've spent getting used to how Windows does things will prove mostly useless when it comes to understanding how Linux works. With all of this in mind, wouldn't it be great if you could try out Linux without spending hours or days getting it installed and configured on your system? Well, you can. With the Ubuntu Live CD, you can take Linux for a test-drive to be certain you really want to commit the time and resources to running it full-time. This hack shows you how to download the Ubuntu Live CD and boot your system using it. Other hacks in the book show you how to get around in GNOME [Hack #15] or KDE [Hack #16], the two popular graphical environments that run on top of Linux.

Downloading the Live CD

A *live CD* is a complete installation of Linux that runs entirely from CD. While you are using a live CD, nothing is written to your hard drive, so your Windows or Macintosh installation is not affected in any way. However, because you're running from a CD, you're limited to using only the programs that are installed on the CD, and everything will run a bit slower because CD access is much slower than that of a hard drive. Still, even with these limitations, it's undeniable that a live CD is the easiest way to try out Ubuntu.

You can obtain the Ubuntu Live CD from the main Ubuntu web site (http://www.ubuntulinux.org). There is a convenient Download link that takes you right to the download page to get the latest released version of Ubuntu. This hack, indeed this entire book, was written for the Dapper Drake release, version 6.05, because it is the release that will be supported for the next five years (previous Ubuntu releases were only supported for 12 months). Ubuntu versions are numbered according to the year and month of release; therefore, this version of Dapper Drake was released in May of 2006. Regardless of which version you download, the hacks in this book should be valid for a long time to come.

AU: Please verify dates and version numbers in above paragraph

The file you want to download is the ISO image that corresponds to the computer type you are using. If you're on a PC, this probably means the x86 version, but if you happen to be using a 64-bit AMD or Intel processor, you want to download the 64-bit PC version. Finally, if you're a Mac user, you want to get the PowerPC version. It is unknown at the time of this writing if Ubuntu will support the new Macintoshes with Intel processors.

You can burn the ISO image to disc using any CD burning software you have installed on your computer. Make sure you choose the option that burns the image to disc; don't select the option to burn a data CD that will just copy the image over as a file. The difference is that the former will create a bootable disc, and the latter will not.

Booting the CD

To use a live CD, you typically need do nothing more than boot your computer with the CD already in the optical drive. Most Windows computers these days are preconfigured to boot from a CD or DVD before booting from the hard drive. I fancy this is because users often need

to restore or repair their Windows installation using the OEM-provided restore CD, and this configuration saves a lot of calls to technical support.

If you are using a Mac running OS X, you need to hold down the C key to boot from a CD.

But, if for some reason your Windows computer doesn't want to boot from the CD, the fix is usually quite simple. You need to boot into your computer's BIOS and modify the setting that specifies the boot order. Getting into the BIOS usually requires you to press a key early on in the boot sequence. The key you press depends upon the make of your computer and BIOS, but it is typically displayed on the splash screen that comes up when your computer starts (the one that announces the manufacturer of the computer, not the Windows splash screen). If your splash screen doesn't tell you this information, try one of these keys: Esc, Del, F2, F10, or F12.

On some computers, F12 launches you directly into a boot selection menu, offering options such as booting from hard disk, floppy drive, USB drive, optical drive, or the network. This lets you boot from a different device without making changes to your BIOS configuration.

Once you're in the BIOS, you should look for a menu called Boot or one labeled Advanced Configuration. Under this menu, you should see a setting that allows you to specify that the CD or optical drive boot before the hard disk. There are hundreds of BIOS variants, so I can't be more specific than that, but if you look at every option screen, you will eventually see the setting you need to change as well as instructions for how to do so. Once you've made the change, save it, and then reboot your computer.

Hopefully, this will be the only problem you have booting from the Live CD. If you've got the BIOS configured correctly, shortly after boot you should see a splash screen with the following options:

Run preinstalled live system

This option loads the Live CD environment so you can test-drive Ubuntu. If you don't press any keys within 30 seconds of getting to this screen, this option will automatically execute.

Rescue a broken system

Choose this option to load a minimal Linux enviroment that you can use to troubleshoot a nonworking Linux installation.

Memory test

You can use this option to run a test of your computer's RAM. Many people don't realize it, but many odd computer problems can be traced to bad RAM modules. If your computer exhibits erratic behavior, such as frequent freezes or an inability to consistently finish booting, your RAM may be the culprit, and running this program may save your hours of frustration.

Boot from the first hard disk

Select this option to continue booting from the hard drive.

Unless you're troubleshooting, about the only other option of interest right now is pressing F2 to select a language. This setting determines the language and keyboard layout that will be used for the rest of the test-drive; the default is English.

Once you've made any necessary language selections, you should use the arrow keys to select "Run preinstalled live system," and press Enter. This begins the loading of Ubuntu. You'll see a lot of messages flash by on the screen and eventually be faced with a text dialog to configure your screen resolution. You can use the tab and arrow keys to move the selection cursor, the spacebar to toggle a selection, and Enter to accept your input and move on to the next screen. You can select multiple entries, depending on what your monitor supports. Ubuntu will use the highest selected and supported resolution as the default.

After this Ubuntu continues to load, and if all goes well, you'll automatically be logged into a GNOME desktop less than a minute later. Depending upon your hardware (network, sound, printer, etc), you may find everything preconfigured and working. If you don't, some of the hardware-configuration hacks later in this book may be useful even in the Live CD environment.

Another Use for the Live CD

The Ubuntu Live CD also includes Windows versions of several open source programs. To access these program installers from within Windows, just insert the live CD while logged in. Within a few seconds the autoload feature of Windows should display a window that lets you launch each installer. If this doesn't happen you can just open Windows Explorer, navigate to the CD, and use the installers found in the program directory. The programs on the CD are:

OpenOffice.org 2.0

> This is a free office suite that includes a word processor, spreadsheet, database, drawing program, and web page creator. OpenOffice.org (the *.org* is really a part of its name, but you can abbreviate it to OOo) can open and save to Microsoft Office formats, which means you may be able to use it in place of that office suite, or at the very least collaborate with others who do. You can learn more about OOo at the OpenOffice.org web site (http://www.openoffice.org).

Mozilla Firefox 1.5

> Firefox is a web-browsing alternative to Microsoft's Internet Explorer. This secure and feature-rich web browser took the computer world by storm in 2005 and became the first browser to gain market share against IE since the mid-90s. To learn more about Firefox, visit the Mozilla web site (http://www.mozilla.org). Pay particular attention to the information about tabs and extensions, two features that can dramatically enhance your browsing experience.

Gaim 1.5.0

> Gaim is a multiprotocol instant-messenger program. This means it can connect to multiple networks, such as AOL, MSN, Jabber, and Yahoo! all at the same time, making it easy for you to stay connected to your friends without having to run a separate chat client for each network.

Each of these programs is also part of the Ubuntu Live CD experience, so you can try them out before installing them to Windows. If you like the Live CD so much that you want to keep using it, be sure to check out "Make Live CD Data Persistent" [Hack #3].

Hack 2. Get Help

Find out where to get more help on using Ubuntu. Forums, Wikis, IRC chat rooms, and a built-in help system stand at the ready.

Everybody needs a place to turn to when he gets stuck. One of the nice things about Ubuntu Linux is the amount of help you can receive, if you know where to look. The development team at Canonical has put together an excellent support infrastructure that includes both free and nonfree support solutions.

Web-based Documentation

Your first stop on the support train should be the Ubuntu Support page at http://www.ubuntulinux.org/support. This page contains links for all the currently possible support methods, both official and unofficial, paid-for and free.

Of course, Ubuntu has excellent documentation. The official documentation effort at http://help.ubuntu.com has both a Quick Tour section and a comprehensive Start Guide. The Quick Tour page is a great flyer that advertises the high points of Ubuntu and shows off some screenshots, while the Start Guide is more of an overall how-to document.

The next place to visit if you're stuck should be the Ubuntu Wiki (https://wiki.ubuntu.com). The Wiki is extremely comprehensive and is constantly updated by Ubuntu users and developers. As a result, it's typically more up-to-date than the official documentation. One of the side benefits to the Ubuntu Wiki is the Laptop Testing area at https://wiki.ubuntu.com/LaptopTestingTeam. If you're about to install Ubuntu on a laptop, you might want to see if your model is on that page, since the Laptop Testing Team puts all their installation notes and tweaks down on that area of the Wiki. It might save you a lot of work and could very well help you get a troublesome feature like wireless or power management working correctly under Ubuntu Linux.

Interactive Help and Support

If you've got a question that you can't find the answer to, you can ask it in either the Ubuntu Forums or the Ubuntu IRC chat room. The Ubuntu Forums at http://www.ubuntuforums.org provide a nearly real-time support venue that you can also search. Odds are, if you're having a problem, someone else has already had that problem and asked for help on the forums. If you've got a more urgent issue, or just want instant gratification, you can ask for help in the IRC chat room. The IRC room is located on the *freenode* network (*irc.freenode.net*), and it's called *#ubuntu*. If you've never used IRC before, just click on the Applications menu, select Internet, and launch Xchat. Log in to *irc.freenode.net* and join the *#ubuntu* channel. Once you're online, ask your question, but be sure to provide as much detail as possible for the people in the room. Please note that most of the people there are volunteers who are contributing to the Ubuntu effort by trying to offer support, so be friendly and be prepared to answer questions that anyone in the room may ask in return, since they may need more information to figure out your issue. You might want to lurk in the channel for a while and read the messages that scroll by to get a feel for the tone and flow before you ask your question.

A lot of the work that makes Ubuntu what it is happens on mailing lists. There's a comprehensive list of mailing lists at https://lists.ubuntu.com/mailman/listinfo; you can either search the archives of these lists, or you can add yourself to them and post your question there. If you choose to post a question to one of these lists, please show proper etiquette and ensure your question is targeted at the correct mailing list. As with IRC, it's worth spending some time to get familiar with the mailing lists: read some older posts and responses, and pay attention to which questions get answers and which ones don't.

Traditional Pay-per-Incident Support

If you can't get a solution to your problem through the aforementioned free methods, there's always paid-for support through Canonical and other organizations. The page at http://www.ubuntu.com/support/supportoptions/paidsupport details the various options open to you for paid support. If you're considering using Ubuntu in a corporate environment, you should become familiar with this page. One thing to note is that with the paid-for support, Canonical has service-level agreements, which is something that you won't get on the free side of the support house.

AU: What are service-level agreements? Are these types of support contracts? Any further clarification needed in previous paragraph?

Whatever your need, the Canonical team and the larger Ubuntu community should have it covered. The support community is widespread, knowledgeable, and ready to help, so don't let a snag in your installation damage your Ubuntu experience!

Hack 3. Make Live CD Data Persistent

Take your desktop with you on a USB stick and access it anywhere with the Ubuntu Live CD

Wouldn't it be handy if you could walk up to any random computer, insert a copy of the Ubuntu Live CD, plug in a USB key, boot it up, and have a fully working system with your own documents, settings, and programs—without modifying the computer in any way?

A little-known feature of the Ubuntu Dapper Drake Live CD allows you to do exactly that. When it starts up, it searches for a volume that has been given the label *casper-cow* and uses it to store documents, themes, and even extra programs that you install. This is far more powerful than just booting up a live CD and mounting a memory stick as your home directory because it's not restricted to just storing your documents. It gives you the flexibility of a fully installed system, while retaining the "go anywhere" feature of a live CD.

You can perform this trick with just about any storage device, including removable USB hard disks and compact flash drives, but for this hack I use a USB memory stick because they're cheap, portable, and commonly available in increasingly large capacities.

Set the Label on Your USB Memory Stick

Connect the USB memory stick to your regular Ubuntu computer. Ubuntu will probably mount it automatically, so the first thing to do is to find the device name that it has been assigned. Open Applications→Accessories→Terminal and type the following at the shell prompt:

```
$ df -h
```

to see a list of mounted volumes. The output should look something like this:

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda3              54G   19G   35G  36% /
varrun                506M   84K  506M   1% /var/run
varlock               506M     0  506M   0% /var/lock
udev                  506M  116K  506M   1% /dev
devshm                506M     0  506M   0% /dev/shm
/dev/hda1             221M   28M  181M  14% /boot
/dev/sda1             498M  214M  285M  43% /media/usbdisk
```

USB storage devices are emulated as SCSI devices by Linux, and you can see the last device is listed as */dev/sda1*. This means SCSI device A, partition 1. If you have anything on the memory stick that you want to save, now is the time to copy it onto your computer, because you're about to totally erase it.

Once you've backed up your files, it's time to unmount the device:

```
$ sudo umount
                /dev/sda1
```

Ubuntu is smart enough to figure out if you are "in" the device (either on the command line or using the file browser), so if the system refuses to unmount because the device is still in use, just close any other windows you have open and try again.

Then create a new filesystem with the correct label:

```
$ sudo mkfs.ext3 -b 4096 -L casper-cow
                  /dev/sda1
```

You must replace `/dev/sda1` with the actual device name used by your memory stick. If you have other USB devices attached, it is possible that one of them has claimed this device name. If in doubt, run the command *dmesg* right after you plug the memory stick in. You should see a message indicating the name of the device that was used to represent your memory stick.

This will create an *ext3* journaling filesystem, which is a good choice for general purpose use, but if you prefer you can use any filesystem that's supported by the Live CD. The *mkfs.ext3* command will report some statistics about the new filesystem and then you're ready to try it out.

Boot the Live CD In Persistent Mode

Plug your USB memory stick into the target machine, power the computer up, and quickly insert the Dapper Drake Live CD. If the computer is not configured to boot from CD-ROM, you may need to press a key (typically DEL or F2) at startup to enter the BIOS settings menu; you then need to change the order of the boot devices to put CD-ROM at the top of the list, and then select the Exit option (the one that saves your changes to the BIOS) from the BIOS menu. The computer will then boot up again and look for the Live CD before attempting to boot from the hard disk. Some computers have a menu (often activated by F12) that let you choose which device to boot from without having to make changes to your BIOS.

If you are using a Mac running OS X, you need to hold down the C key to boot from a CD.

When the Live CD starts up, you will see a menu. Normally you would just press Enter to start the boot process, but instead press F4 to access the Other Options menu that allows you to start up the Live CD in special modes. You'll see a list of the arguments that will be passed to the kernel on startup: just add a space and type `persistent`; then hit Enter.

That's it!

Testing Persistence

The computer will now boot from the Live CD in persistent mode, but you won't see anything different. In fact, it can be quite hard to tell if it even worked or not. As a simple test, you can try changing something obvious, such as your desktop picture, and then you can log out and reboot the computer back into persistent mode. If everything worked properly, your desktop picture will still be set as you specified.

Try changing other things on your system such as creating documents or even installing extra software. Changes you make should be preserved even after you reboot the system.

How It Works

The Live CD is a read-only environment, so of course you can't save changes made to the running system straight to the CD. However, when running in persistent mode, the system on the Live CD allows items on your memory stick to override items within the Live CD environment. In the test described in this hack, you changed the desktop image; this caused Ubuntu to save your new desktop picture and settings onto the *casper-cow* device. The next time the Live CD sets the desktop, it detects that a new setting has been stored on the device and applies it instead of the default setting. The Live CD therefore provides the basic data for a complete, functional environment, and any changes you make to that environment are written to the removable device and used to override the default settings.

Hack 4. Customize the Ubuntu Live CD

Rip, burn, and boot to create a personalized version of the Ubuntu Live CD with your choice of software and documents.

The Ubuntu Live CD [Hack #1] contains a complete Ubuntu installation that can run directly from the CD itself, without needing to be installed onto a hard disk. It's ideal for demonstrating Linux on computers with another operating system installed because after you take the CD out and reboot the computer, it returns to exactly the state it was in originally. It's a totally painless way to take Linux for a test run with no risk.

The Live CD is also extremely useful for recovering an unbootable machine: just pop in the Live CD and reboot, and you will have a fully running Linux system from which you can access the internal hard disk, copy files across the network, or do whatever else you need to do to fix the system. And as you saw in "Make Live CD Data Persistent" [Hack #3], you can even use a memory stick to store changes made inside the Live CD environment.

The Ubuntu Live CD starts up a full desktop environment that's functionally identical to a standard Ubuntu installation, but perhaps you want a Live CD that contains specific software or documents to suit your environment. For example, you may want to create a Live CD that boots up a machine as a fully configured router and firewall with no hard disk. Or maybe you want a forensics disk preloaded with virus-scanning and network-analysis tools plus the checksums of important files.

No problem, you can create a customized version of the Ubuntu Live CD configured exactly the way you want it.

Basic Requirements

Building the disk image for the Live CD takes a huge amount of storage so you'll need up to 5 GB of swap plus at least another 3 GB of disk space for storing the image. You'll also need tools for creating and mounting disk images.

Add extra swap

While the disk image is being compressed, *two copies* of it are held entirely in memory, so without a huge amount of swap, you won't be able to do the compression necessary to generate the ISO.

Don't worry if you don't already have a 5 GB swap partition. You can set up a temporary swapfile inside one of your existing partitions without having to reformat. Assuming you have at least 5 GB of space free inside */tmp* (usually in your root partition), you can create the extra swapfile with *dd*:

```
$ sudo dd if=/dev/zero of=/tmp/swap bs=1M count=5000
```

It can take a very long time to create the swapfile, so you'll need to be patient. Once the file itself has been created, you can set up a swap filesystem on it and activate your new swap:

```
$ sudo mkswap /tmp/swap
$ sudo swapon /tmp/swap
```

You don't need to disable your existing swap first. Linux is smart enough to handle multiple swap partitions simultaneously, so your system should now have a total swap space comprising the new 5 GB swapfile plus your existing swap.

Install the tools

To mount a disk image as a loopback device and generate the ISO for your custom Live CD, you will need the *cloop-utils* and *mkisofs* packages, and to work with the *squashfs* compressed image on the Live CD, you'll need the *squashfs-tools* package:

```
$ sudo apt-get install cloop-utils mkisofs squashfs-tools
```

Standard Live CD

While it's possible to build a Live CD from scratch, it's much easier to start by modifying the standard Ubuntu Live CD. You can download the Live CD ISO disk image from http://cdimage.ubuntu.com/ or use one of the CDs available from Canonical through the ShipIt program (https://shipit.ubuntu.com/).

Prepare Original Image

Make sure your locale is set to `C` to prevent Unicode problems with the build process:

```
$ export LC_ALL=C
```

Mount the original Ubuntu Live CD ISO image as a loopback device:

```
$ mkdir ~/mnt
$ sudo mount dapper-live-i386.iso ~/mnt -o loop
```

This will mount the CD image inside your home directory at *~/mnt*. You can use an alternative location or mount the actual Live CD in your CD-ROM drive if you prefer.

Copy everything from the mounted image into a working directory, but make sure you skip the *filesystem.squashfs* compressed filesystem because you'll need to extract that separately. You can use *rsync* to make it easy:

```
$ rsync --exclude=/casper/filesystem.squashfs -a ~/mnt/ ~/extracted_cd
```

Next extract the compressed filesystem. The Dapper Live CD uses the *squashfs* read-only filesystem, unlike previous Ubuntu Live CDs, which just used *cloop* filesystems. To work with *squashfs*, you will need to load the *squashfs* kernel module:

```
$ sudo modprobe squashfs
```

Now you can mount it and copy it onto your local hard disk:

```
$ mkdir squash
$ sudo mount -o loop ~/mnt/casper/filesystem.squashfs squash
$ sudo cp -a squash extracted_fs
```

Be prepared to wait quite a while for this to run. Once it's finished, you will have a complete, extracted copy of the Live CD image, so you can unmount the original:

```
$ sudo umount ~/mnt
```

Set Up the Target Filesystem

Mount the *proc* and *sys* virtual filesystems into your target:

```
$ sudo mount -t proc proc ~/extracted_fs/proc
$ sudo mount -t sysfs sysfs ~/extracted_fs/sys
```

In a moment, you'll be chrooting into the CD image, so if there are files you will need on your customized CD, the easiest thing to do is mount */home* into it:

```
$ sudo mount -o bind /home ~/extracted_fs/home
```

Then once you are in the chroot, you will have full access to any files stored in your home directory.

Apply Customizations

Use *chroot* to enter the filesystem image:

```
$ sudo chroot ~/extracted_fs/ /bin/sh
```

Now, as far as you're concerned, you're running on a read/write installation of the Live CD. From there, you can use the usual package tools to update programs installed on the Live CD.

Delete unnecessary packages

The default Live CD is fairly full, so if you want to install extra packages, you will probably need to make some room first. If you want some ideas about which packages to remove, you can create a list of installed packages sorted by size using this command:

```
$ dpkg-query -W --showformat='${Installed-Size;10} ${Package}\\n' | \\
             sort -gr | less
```

Be very careful, though, because some packages are essential for the system to work at all. The GNOME Live CD is based on Ubuntu, so if you're looking for inspiration for which packages you can safely remove, you can start by looking at its configuration file, available at http://cvs.gnome.org/viewcvs/livecd-project/livecd.conf?view=markup.

Once you've settled on some packages to remove, you can uninstall them using *dpkg*:

```
$ sudo dpkg -r --purge packagename
```

Install additional packages

The regular network-based package tools won't work inside the chroot, so unfortunately it's not as simple as `apt-get install` *`foo`* to add packages. There are a number of ways around it, such as copying in a *hosts* file with the addresses of repository servers pre-resolved, because you can't do DNS lookups inside the chroot.

The simplest way, though, is probably just to predownload some packages into your home directory and use *dpkg* to install them after entering the chroot.

One very cool trick to simplify this process is to run the Synaptic package manager on your host system, find and mark the packages you want to install on your Live CD, and then select File→"Generate package download script." You will then have a script that you can execute to fetch and save the packages locally, storing them in your home directory for access from the chroot.

Customize the home directory

When the Live CD boots, it creates the user's home directory from scratch each time, using the files in */etc/skel*. If you have specific files you want to include in the home directory, you can put them in *skel*.

Unmount Customized Image

Now that all your changes have been applied, exit the chroot and then unmount the various filesystems:

```
$ exit
$ sudo umount ~/extracted_fs/home
$ sudo umount ~/extracted_fs/sys
$ sudo umount ~/extracted_fs/proc
```

Your customized filesystem is now ready to recompress, but first you need to generate a new manifest file that reflects the changes you have made to the list of installed packages. If you didn't actually install or remove any software, you can skip this step.

You can't perform this step using *sudo* (you have to really be running as *root*), so get a *root* shell with `sudo -s`:

```
$ sudo -s
```

Now generate the new manifest:

```
# chroot extracted_fs dpkg-query -W \\
              --showformat='${Package} ${Version}\\n' \\
              > extracted_cd/casper/filesystem.manifest
```

You can exit the *root* shell now.

Repack the Filesystem

The new *squashfs* filesystem that will go inside the CD needs to be created:

```
$ sudo mksquashfs extracted_fs extracted_cd/casper/filesystem.squash
```

Once again, this stage can take a really long time.

The Live CD also needs to contain a checksum file that can be used to verify the integrity of the compressed filesystem. The checksum needs to be calculated from inside the CD image:

```
$ cd ~/extracted_cd
$ find . -type f -print0 | xargs -0 md5sum > md5sum.txt
```

Build the ISO

Everything up until this point has been architecture-independent, but the final stage of building the ISO itself depends on what type of system you are running:

x86 (i386) and x86_64 (amd64)

Use the following command:

```
$ sudo mkisofs -r -V "Custom Ubuntu 6.04 Live CD" \\
                    -cache-inodes \\
                    -J -l -b isolinux/isolinux.bin \\
                    -c isolinux/boot.cat -no-emul-boot \\
                    -boot-load-size 4 -boot-info-table \\
                    -o custom-dapper-live-i386.iso extracted_cd
```

PowerPC

For PowerPC it's necessary to download *hfs.map*:

```
$ wget http://people.ubuntu.com/~cjwatson/hfs.map
```

Then build the actual ISO:

```
$ sudo mkisofs -o new_image.iso -chrp-boot \\
                    -U -part -hfs -T -r -l -J -A "application_id" \\
                    -sysid PPC -V "volid" -volset 4 -volset-size 1 \\
                    -volset-seqno 1 -hfs-volid "volume_name_hfs" \\
                    -hfs-bless extracted_cd/install \\
                    -map hfs.map -no-desktop -allow-multidot extracted_ppc_cd
```

IA64

Use the following command:

```
$ sudo mkisofs -r -V "Custom Ubuntu 6.04 Live CD ia64" \\
                    -o custom-dapper-live-ia64.iso -no-emul-boot \\
                    -J -b boot/boot.img -c boot/boot.catalog extracted_cd
```

Burn and Boot

You now have an ISO image of your customized Live CD, so burn it to a disc [Hack #33] and give it a try.

More Information and Scripts

The process of creating a customized Live CD is quite manual and laborious, but some of the steps above can be simplified using the *live_cd_tools* scripts that you can find online at http://wiki.ubuntu.com/LiveCDCustomizationHowTo. Note, however, that the process for building the Dapper Live CD is a bit different from the older process used by previous releases, such as Breezy, that used a compressed *loopback* filesystem instead of *squashfs*, so make sure you don't use scripts intended for the older process.

Hack 5. Install Ubuntu

Learn how to install Ubuntu on your computer.

If you've given Ubuntu a test-drive [Hack #1], or you're simply ready to dive into it sight unseen, all you need is an installation CD and a computer to install it on, and you can be up and running right away. There are a number of ways you can get an installation CD; if you've got broadband and a CD-R drive, you can probably get your hands on it in under an hour.

System Requirements

Ubuntu will run on just about any current personal computer. If you're using an Intel-compatible PC, it will probably "just work," since the kernel image that Ubuntu uses by default is optimized for the 80386, which means it will also be compatible with systems based on the 486, Pentium, and Pentium Pro, as well as the Pentium II, III, 4, and beyond, including all the other mainstream Intel-compatible CPUs such as the AMD Athlon and Sempron, as well as the Transmeta Crusoe and Efficeon. If your computer can run Windows 95 or later, it can probably run Ubuntu just fine. If you're running an AMD64 system, there is even a special version of Ubuntu you can download.

If you've got a G3, G4, or G5 Macintosh, you'll probably be able to run the PowerPC version of Ubuntu. If it can run Mac OS X, it should be able to run Ubuntu. Mac users should see "Install Ubuntu on a Mac" [Hack #8] for complete details.

Although you may have a CPU that's compatible with Ubuntu, you may run into some hardware that doesn't want to play along. Wireless network cards can be particularly tricky, but after you get Ubuntu up and running, there are some tricks [Hacks #41 and #42] you can use to get them working. However, because the Ubuntu installer tries to use the network, I strongly urge you to keep an Ethernet cable handy in case you need to plug your system into a wired network for the install. (Early on, the Ubuntu installer will report which network interfaces it was able to activate, so if you don't see your wireless network adapter listed, it's time to use that Ethernet cable.)

Disk space and memory are probably your most important considerations. If you are planning on running the GNOME (Ubuntu) [Hack #15] or KDE (Kubuntu) [Hack #16] desktop environment, your computer will benefit from plenty of RAM and disk space. Consider 2 GB of disk space and 256 MB of RAM to be a comfortable minimum. And with the price of disk space below $1 per GB and RAM being between $50 (desktop) and $100 (laptop) per GB, the more the merrier.

Preserving Your Existing Data

If you've already got Windows or Linux running on your system, and you want to keep it, you should check out "Dual-Boot Ubuntu and Windows" [Hack #6], which explains how to set up a dual-boot Ubuntu system. However, if you're just interested in archiving your existing installation so that you can pull files off of it at a later time [Hack #7], you have some choices:

New hard drive

Since the cost of storage is so low, you might want to pull and replace your current hard drive. Depending on the age of your current machine, that might give you a modest performance improvement, but it also gives you the opportunity to increase your disk space. Once you've done so, you can install your existing hard drive into an external enclosure so that you can access your old files (and use any free space for extra storage). Another option would be to simply buy an external drive and copy all your files onto it before you install Ubuntu.

Burn it to optical media

If your files can fit, burn them to CD or DVD for safekeeping. Linux will be able to read practically any format that you can put on an optical disc if you want to retrieve these files later.

Shrink your existing partition

If you want your data at your fingertips, then you should spend some time deleting everything you can live without from your old operating system: applications, data files, etc. If you're running Windows, make sure you disable hibernation in the control panel's power settings, because hibernation requires a file equal to the size of your installed RAM. Also, if you're not planning to boot the old operating system on a regular basis, disable the paging file. Do everything you can to free up disk space. Then defragment your drive and shrink the partition using a tool such as Partition Magic (http://www.symantec.com/partitionmagic/) or the Ubuntu installer's partitioning tool. If you want your old data at your fingertips and you still want to be able to boot up the old OS, be sure to check out "Dual-Boot Ubuntu and Windows" [Hack #6].

Move the data to another machine

If you've got another computer with lots of storage, copy the files across the network to a safe spot on that computer. Even if you have a wireless network, consider using a network cable for the transfer, since it will run a lot faster that way.

Getting the Installation CD

Before you install Ubuntu, you'll need to choose which flavor you want: Ubuntu, Kubuntu (Ubuntu with KDE as the default desktop), or Edubuntu (Ubuntu designed for young people). Although you can download a different CD-ROM for each one, you can install their core components [Hack #54] later; the packages *ubuntu-desktop*, *kubuntu-desktop*, and *edubuntu-desktop* can be installed at any time.

Another flavor of Ubuntu is Xubuntu (the package *xubuntu-desktop*), available at https://wiki.ubuntu.com/Xubuntu, which is a variant of Ubuntu that's optimized for older computers. The desktop system is more lightweight, so it's less demanding in terms of memory, CPU, and video-card resources.

Once you've decided which flavor of Ubuntu you want, go to the Ubuntu (http://www.ubuntu.com), Kubuntu (http://www.kubuntu.org), or Edubuntu (http://www.edubuntu.org) home page and follow the download link. Next, you'll need to choose which mirror site to install from (pick one that's close to you geographically), and then choose Intel x86 (for most PC compatibles), AMD64 (64-bit AMD-based PC compatibles), or PowerPC (for Macs, but if you choose this one, you should be in a different hack [Hack #8]).

On the download site, you'll find links that go directly to the CD image, an ISO file that you can burn to a CD-R using your favorite CD burning utility. Mac OS X includes Disk Utility in the */Applications/Utilities* folder. Linux usually includes the command-line *cdrecord* utility,

and in many cases, there will be a graphical frontend available [Hack #33]. Windows does not have its own CD burning utility, but most PCs come bundled with a CD burning application.

Be careful on Windows. I burned a few discs on my Dell 700m that wouldn't boot at all before I figured out what was happening. It turned out that the CD burning application that Dell so thoughtfully bundled with my computer wouldn't burn bootable ISOs without a paid upgrade. However, there are some free ISO burners for Windows. Microsoft includes a free ISO burner (CDBURN) in its Windows Server 2003 resource kit (http://www.microsoft.com/downloads/details.aspx?FamilyID=9d467a69-57ff-4ae7- 96ee-b18c4790cffd&DisplayLang=en), which, despite the name, runs on Windows XP. Alex Feinman's ISO Recorder (http://isorecorder.alexfeinman.com/isorecorder.htm) is free and works well.

Installing Ubuntu

Most computers will boot from the CD-ROM drive automatically, so put the Ubuntu CD into your computer, shut it down, and reboot. (If you try to insert the Ubuntu CD in the computer while it's booting, there's a good chance that whatever's on your hard drive will start booting before your computer even knows you've inserted a bootable CD, so press Ctrl-Alt-Delete after you insert the CD). If your computer refuses to boot from the CD, reboot it and enter the BIOS menu (usually by pressing F2, Del, or Escape as the computer is starting). Navigate through the BIOS menu and look for a set of boot options, and make sure that the CD-ROM is configured at a higher priority than your hard drive (it should be the first or, if you have a floppy drive, the second item in the list). If in doubt, consult the manual that came with your motherboard or your PC. If you don't have that manual, you should be able to download it in HTML or PDF form from the manufacturer's web site.

In very rare cases, you may have a computer that can't boot from CD-ROM. I had an ultraportable Sharp Actius MM10 that came with an external CD-ROM drive. It worked great until I tried to boot from a Linux install CD. It turned out that the CD-ROM drive was defective, and Sharp exchanged it for a working one. However, if you're in this situation, you may not have to wait for a new drive to arrive: instead, you could boot over the network [Hack #11] to install Ubuntu on your computer.

When you boot from the CD, you'll be offered several Install options: to the hard disk, OEM mode, or server. If you choose Install to the Hard Disk, you'll go through the standard Ubuntu install. OEM mode is good if you're setting up a computer that someone else will use, since it lets you set up all the hardware configuration and choose software packages, but lets the user do all the personalization (choosing a username, picking her preferred language, etc.). The server installation will set up a minimal Ubuntu system; if you want to add a desktop environment later, you can install *ubuntu-desktop*, *kubuntu-desktop*, *edubuntu-desktop*, or *xubuntu-desktop* when the need arises.

Before you start the installation, you can press F3 to choose the video mode to use, F4 to enable some assistive features, F5 to allow you to edit the boot-time kernel arguments and to enable some more options (including keyboard map and expert mode). Once you've made your choices (or left everything at the defaults), press Enter to start the installation. During the installation, you'll go through the following steps (in expert mode, you'll be taken to a main menu between each step):

Choose Language

In this step, you'll choose the default language for your system.

Choose Locale

This lets you select which locale or locales to install on your system. The *locale* is a feature that works in conjunction with the language to control such things as alphabetic sort order and the way dates and numbers are displayed.

Select a Keyboard Layout

This is where you choose the type of keyboard connected to your system.

Detect CD-ROM

By default, this step won't prompt you unless it has a problem finding your CD-ROM. In expert mode, you'll get to review and customize the installer's decision. You'll also be prompted to enable PCMCIA/PC Card services in expert mode during this stage.

Load Installer Components from CD

As with the previous step, the installer won't bug you unless it has a problem here. In expert mode, it will offer some extra components.

Detect Network Hardware

In this step, you get to choose which network adapter to use. In expert mode, it will offer some extra options.

Configure the Network

By default, the Ubuntu installer will attempt to configure the network using DHCP. If it fails, it will offer you some options to configure it differently. In expert mode, you'll be able to configure the network manually. Once the network is configured, you'll be prompted for a hostname (the default will be *ubuntu*, unless the installer finds an existing installation, in which case it will determine the hostname from that). In expert mode, you'll also be prompted for a domain name.

Choose a Mirror

In expert mode, this step lets you choose which mirror to use for additional components and package repositories. In regular mode, you'll be prompted to supply only an HTTP proxy, a server that makes requests to web servers on your behalf (usually found only on corporate or campus networks; leave it blank if you don't use one).

Detect Disks

In this step, the Ubuntu installer will detect hard disks. In expert mode, you will be prompted for some additional options, but normally this will run without any intervention.

Partition Disks

At this stage, the Ubuntu installer will offer some choices: resize your existing partition and use the free space, erase the entire disk, or manually edit the partition table. You'll notice that there are two ways you can erase the entire disk: just using the disk (usually something like "Erase entire disk: IDE1") or creating a logical volume ("Erase entire disk and use LVM"). Logical volumes offer advanced features, such as the ability to dynamically reallocate disk space between partitions.

If you want to keep your existing operating system, choose the resize option. If you want to wipe out the entire disk and put Ubuntu on it, choose one of the other options (the basic "Erase entire disk" option is a good choice if you are unsure).

Confirm Partitioning

This is the point of no return, so the Ubuntu installer reviews the partitioning choices you've made and asks you to confirm them. If you say Yes here, it will repartition your machine, possibly destroying existing data if you've chosen to remove or replace an existing partition.

Configure the Time Zone

In this step, you'll need to choose your time zone.

Configure the Clock

The Ubuntu installer needs to know how your clock is configured. Most computers have a clock that's set to the local time, but Ubuntu lets you configure it so it's set to Coordinated Universal Time (UTC). The choice here is not terribly important unless you plan to dual-boot between Ubuntu and Windows, in which case you should not set the clock to UTC.

Set Up Users and Passwords

At this point, you'll be able to add a user to the system. If you are in expert mode, you'll have some additional options, including the ability to create a password for the *root* user (by default, Ubuntu expects you to use the *sudo* command to use programs that require *root* privileges).

Install the Base System

This step will run without your intervention unless there is a problem. The installer will put a base installation of Ubuntu onto your system at this point. In expert mode, you'll have the opportunity to choose which kernel image to install.

Configure apt

This step is one of those points when it's a good idea to have a network connection, since the installer will try to contact the Ubuntu mirrors to verify that they are reachable. If it can't, it won't add them to the configuration file (*/etc/apt/sources.list*). In expert mode, you'll be asked some additional questions.

Select and Install Software

This step installs the rest of Ubuntu, including the GNOME (or KDE) desktop, and can take quite a while. Despite the name of the step, you won't be asked to choose which packages are installed. However, you will be prompted to select which video modes to use with the X server.

Copy Remaining Packages to Hard Disk

This step is offered in expert mode and gives you the option to copy 400 MB of packages to your hard drive, which eliminates the need to keep the CD-ROM handy.

Install the Boot Loader on a Hard Disk

This installs the GRUB boot loader on your hard drive. You'll be able to choose the LILO boot loader instead if you are in expert mode. Either LILO or GRUB is capable of booting Ubuntu, but GRUB is more flexible and easier to configure. In expert mode, you'll also be given the opportunity to select a password for your boot loader.

Finish the Installation

This step writes configurations to your new Ubuntu install and reboots you into your new Ubuntu system.

Once you've got Ubuntu up and running, you can log in as the user you created during setup, and start exploring your desktop, whether it be GNOME [Hack #15] or KDE [Hack #16].

Brian Jepson

Hack 6. Dual-Boot Ubuntu and Windows

If you're not ready to give Ubuntu total control over your computer, you can meet it half-way. Learn how to install Ubuntu so you can dual-boot with Windows, even if Windows already owns your entire hard drive.

"Install Ubuntu" [Hack #5] details how to install Ubuntu Linux on your machine as the primary operating system. But what if you're not ready to ditch Windows, or you've got a business requirement to run a certain Windows-only application? A possible solution for you might be to enable your system to dual-boot both Windows and Ubuntu. A dual-boot system has multiple hard disk partitions or hard disks, with each partition or disk containing a complete operating system. Typically, there is a boot loader installed on the first hard disk in the system that lets you choose which operating system to boot when you power on the system.

The Dapper Drake version of Ubuntu supports setting up a dual-boot environment from within the installer. Previous versions also had this capability; however, Dapper's installer automatically shrinks your current Windows partition and makes space available for the Ubuntu installation. Prior to this feature, you had to manually shrink your current Windows partition using tools like PartitionMagic or qtparted.

Preparation

There are just a couple of preparation steps that must be taken prior to setting up a dual-boot system:

- Your current Windows partition must be freshly defragmented to ensure that there is a large, contiguous block of free space available to dedicate to Ubuntu.

There are some files that the Windows defragmentation utility can't move, so you may want to try a third-party defragmentation utility, such as Executive Software's Diskeeper (http://www.diskeeper.com/defrag.asp). However, if it's your swap (paging) file that refuses to budge, and you have sufficient memory to run without one, try disabling it (right-click My Computer, choose Properties, select Advanced→Performance→Settings→Advanced→Change, and choose No Paging File), defragmenting your hard drive using the Windows disk defragmenter, and then re-enabling the paging file.

- You must back up any critical data you have on your Windows partition. The Ubuntu installer tries to resize your partition as safely as it can, but like any other disk utility, there is a slim chance of a loss of data. Play it safe and back up anything you can't live without.

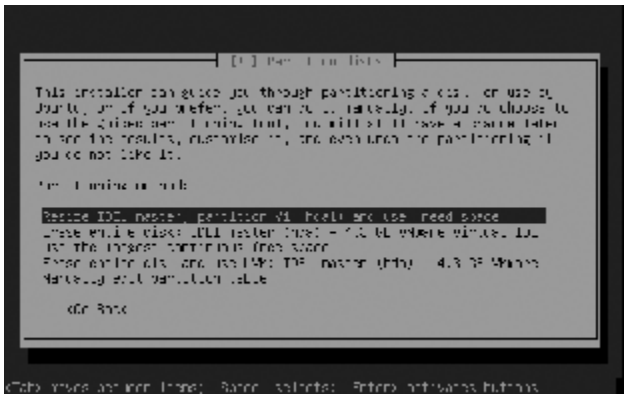
Installation

Let's get started on the dual-boot installation. (Something to remember is that the dual-boot installation is almost like a standard installation; the major difference lies in the method the partitioner uses to partition the hard disk.) First, boot from the CD, just like a standard standalone Ubuntu install. From the installer screen (see Figure 1-1), select Install to the hard disk, and press Enter. The installer will kick off and begin the installation.

Figure 1-1. The Ubuntu installer, beginning a dual-boot install

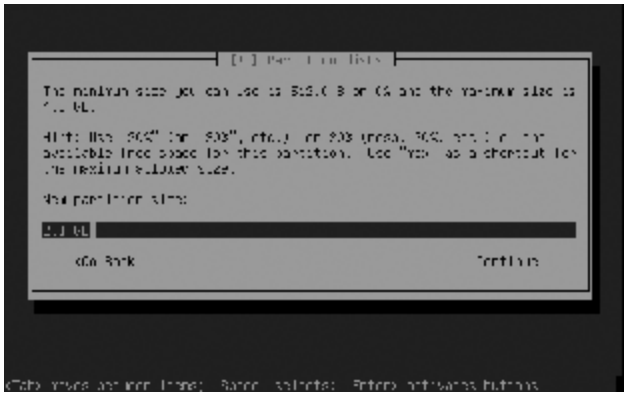
You'll follow the standard installation procedure [Hack #5], up to the point when the system will ask you how you want your disk partitioned. Rather than selecting "Erase entire disk," you'll select "Resize IDE1 master, partition #1 (hda1) and use freed space," as shown in Figure 1-2.

Figure 1-2. Resizing the disk



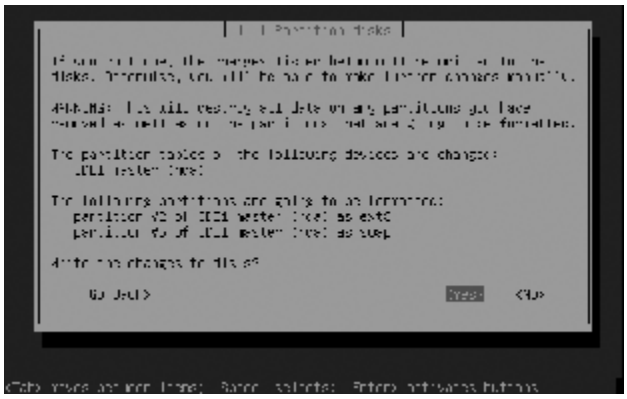
At this point, the partitioner will ask you how much space you wish to devote to Linux. Input your desired Linux partition size in either percent or gigabytes and select Continue (see Figure 1-3).

Figure 1-3. Assigning the free space to Linux



The partitioner will then ask you to confirm your decisions, and then it will write the changes to disk. If all looks good, select Yes to proceed, as shown in Figure 1-4.

Figure 1-4. Writing the partition changes to disk



After this, the installer will actually partition the disk and format your new Linux partition as an *ext3* filesystem, and then you'll be asked to enter your full name to create your Ubuntu account. From this point on, there is no difference between the dual-boot installation and a standard installation. The system will begin copying the binaries and other data from the CD-ROM, and at the end of the install, the GRUB boot loader will be written to the master boot record. The installer will prompt you to reboot, and you'll be able to select from Windows or Linux at boot time.

Hack 7. Move Your Windows Data to Ubuntu

Your files, bookmarks, and other settings are locked away in a Windows installation. Learn how to move them over to your new Ubuntu system.

So you're making the big move. You're ready to pack everything up and move from Windows to Ubuntu. The easy part is getting Ubuntu up and running. The trickier part is migrating all your data, which is spread out all over your Windows hard disk. Here's how to pack all your stuff up and make use of it on your new Ubuntu system.

Make Sure You Notify the Post Office

If you're switching from Outlook, you probably won't be able to directly import your mail settings into a Linux mail program. Your best bet is to install Thunderbird (http://www.mozilla.com/thunderbird/) on your Windows machine and import all your Outlook settings into Thunderbird. Once you've done that, you'll more easily be able to export your mail and contacts into formats that Linux mail programs can understand.

If your mail account is on an IMAP server, you won't need to worry about all this exporting and importing. Since IMAP keeps the mail on the server, all you need to do is configure your new mail client with your email server, login, and password information, and all your mail (your inbox and email folders) will appear on the

new system. Because IMAP keeps everything on the server, you can access the same email account from multiple servers, and you'll always have the same email messages on each computer. However, if you've moved any mail into local folders, you will need to export and import it.

Transfer Outlook into Thunderbird

Before you transfer your Outlook email into Thunderbird, first make sure that Outlook is set to be the default mail application (if you've been using it for your mail application, it probably is). Open the Control Panel and double-click on Internet Options. Go to the Programs tab and make sure that Microsoft Outlook (or Outlook Express, depending on which one you use) is specified as the email program, and then click OK.

Next, launch Mozilla Thunderbird. Select No when it asks whether you want to make Thunderbird your default email application. If this is the first time you've run it, it will prompt you to import your mail and settings.

If not, select Import from the Tools menu. Now you're ready to import your mail. When the Import dialog box appears, select Mail and click Next. Choose Outlook or Outlook Express and click Next. When it's done, click Finish. Your mail is now sitting in Thunderbird.

Transfer your Thunderbird mail to Ubuntu

If you transferred your mail from Outlook, it will be sitting in a Local Folder called Outlook Mail. You should have a few folders there, including Deleted Items, Drafts, Inbox, Outbox, and Sent Items. If you expected a whole bunch of mail to be imported from Outlook but found nothing, check to see whether you are using IMAP for your email (see the earlier note about transferring your IMAP account).

To grab the mail folders from Thunderbird, you'll need to locate your *profile folder*. First, shut down Thunderbird, and then open a Windows Command Prompt. Next, change to your Thunderbird directory with this command:

```
cd %APPDATA%\\Thunderbird
```

(`APPDATA` is a Windows environment variable that points to the currently logged-in user's *Application Data* directory). You'll find a *Profiles* directory in there, and in that directory, there should be a directory with a funny name, such as *pr4qpneu.default*. This is your Thunderbird profile directory. In this directory, look for the subdirectory *Mail\\Local Folders*, and then look for your mail folders, which may be in yet another subdirectory.

For example, this directory contains some empty folders (such as *Drafts*, *Sent*, and *Trash*):

```
C:\\... \\Local Folders> dir
 Volume in drive C has no label.
 Volume Serial Number is 864B-4AB0

 Directory of C:\\Documents and Settings\\bjepson\\Application
Data\\Thunderbird\\Profiles\\pr4qpneu.default\\Mail\\Local Folders

03/12/2006  04:24 PM    <DIR>          .
03/12/2006  04:24 PM    <DIR>          ..
10/04/2005  05:11 PM                 0 Drafts
03/12/2006  03:07 PM             1,445 Drafts.msf
10/04/2005  06:36 PM             1,236 Junk.msf
```

```
03/12/2006  04:24 PM                27 msgFilterRules.dat
03/12/2006  04:24 PM                 0 Outlook Mail
03/12/2006  05:17 PM             1,185 Outlook Mail.msf
03/12/2006  04:29 PM    <DIR>          Outlook Mail.sbd
10/04/2005  05:11 PM                 0 Sent
03/12/2006  03:07 PM             1,443 Sent.msf
10/04/2005  06:36 PM             1,320 Templates.msf
03/12/2006  04:24 PM                 0 Trash
03/12/2006  05:17 PM             1,448 Trash.msf
10/04/2005  05:11 PM                 0 Unsent Messages
03/12/2006  03:07 PM             1,797 Unsent Messages.msf
              13 File(s)          9,901 bytes
         3 Dir(s)  67,812,487,168 bytes free
```

But the imported Outlook folders are in the *Outlook Mail.sbd* subdirectory, including a 13 MB *Inbox* folder (these are the files you want to copy over to your Linux system):

```
C:\\... \\Local Folders> dir "Outlook Mail.sbd"
 Volume in drive C has no label.
 Volume Serial Number is 864B-4AB0

 Directory of C:\\Documents and Settings\\bjepson\\Application
Data\\Thunderbird\\Profiles\\pr4qpneu.default\\Mail\\Local
Folders\\Outlook Mail.sbd

03/12/2006  04:29 PM    <DIR>          .
03/12/2006  04:29 PM    <DIR>          ..
03/12/2006  04:24 PM                 0 Deleted Items
03/12/2006  04:24 PM             1,212 Deleted Items.msf
03/12/2006  04:24 PM                 0 Drafts
03/12/2006  04:24 PM             1,205 Drafts.msf
03/12/2006  04:24 PM        13,094,192 Inbox
03/12/2006  04:44 PM            79,534 Inbox.msf
03/12/2006  04:24 PM                 0 Outbox
03/12/2006  04:24 PM             1,205 Outbox.msf
03/12/2006  04:24 PM                 0 Sent Items
03/12/2006  04:24 PM             1,209 Sent Items.msf
              10 File(s)     13,178,557 bytes
               2 Dir(s)  67,812,483,072 bytes free
```

These folders use the Unix mbox format and can be imported into most Ubuntu email programs. For example, you can basically reverse the process to move the folder from Thunderbird on Windows to Thunderbird on Linux: find your *Local Folders* directory in your Ubuntu *~/.mozilla-thunderbird* directory, make sure that Thunderbird is not running, and copy the files into that directory. When you restart Thunderbird, the folders you copied should appear in your list of Local Folders.

In Evolution, you can select File→Import to bring up the Evolution Import Assistant. When prompted to choose the Importer Type, select Import a Single File and then specify the mbox file when prompted to choose a file. If you have trouble importing files into Evolution, you can copy them manually: exit out of Evolution, and copy the files to *~/.evolution/mail/local*.

If you have any problems, check the MozillaZine article on importing and exporting mail at http://kb.mozillazine.org/Importing_and_exporting_your_mail. However, considering how complicated it is to move mail from one system to another, you should consider using IMAP, which eliminates all these steps the next time you need to move from one machine to another; with IMAP, all your mail and folders are stored on the server. Contact your system administrator, Internet Service Provider, or mail provider to find out they support IMAP.

Are You Bringing the Browser?

You'll be happy to hear that exporting and importing your browser settings will be simple compared to migrating email. If you're using Internet Explorer, you'll first need to export your cookies and bookmarks and then copy them over to your Ubuntu system. To export them, start Internet Explorer, and select File→Import and Export. This starts the Import/Export Wizard. You'll have to run it twice: once for your cookies and once for your favorites (bookmarks).

If you're using Firefox on your old system, you can locate your profile folder and grab the *bookmarks.html* and *cookies.txt* files. To find your Firefox profile directory, open up a Windows Command Prompt and issue this command:

```
cd %APPDATA%\\Mozilla\\Firefox\\Profiles
```

As with Thunderbird, you'll find a strangely named directory, such as *9yk75acu.default*. Your cookies and bookmarks will be sitting in this directory.

Once you've grabbed your cookies and bookmarks, you can use your browser's import function to bring them over to Ubuntu. For example, with Firefox, you can import bookmarks by selecting Bookmarks→Manage Bookmarks, and then selecting File→Import. To copy your cookies over, you'll first need to close Firefox, then paste the contents of the exported cookies into your existing *cookies.txt* file in your Firefox profile directory on your Ubuntu system, located in *~/.mozilla/firefox*.

Cookies that you've exported from Internet Explorer are a bit more complex: you'll need to edit the *cookies.txt* file and put a period before every cookie that begins with a domain name. For more information, see http://kb.mozillazine.org/Import_cookies.

Your Stuff, Your Music

Unless you've decided to put your documents elsewhere, Windows makes moving them to Ubuntu quite easy. Windows uses the *My Documents* folder to organize your documents—including music, video, and others—and most applications respect this organization (for example, iTunes organizes its files under *My Documents**My Music**iTunes*). The actual location of the *My Documents* folder varies by Windows version. For example, on Windows XP, it is on the system drive (usually C:) in *DocumentsandSettings*\\ `username` *My Documents*.

Although it's easy to copy the files over, some files, such as multimedia files, will probably not play on Ubuntu without a little extra effort. See Chapter 3 for more information.

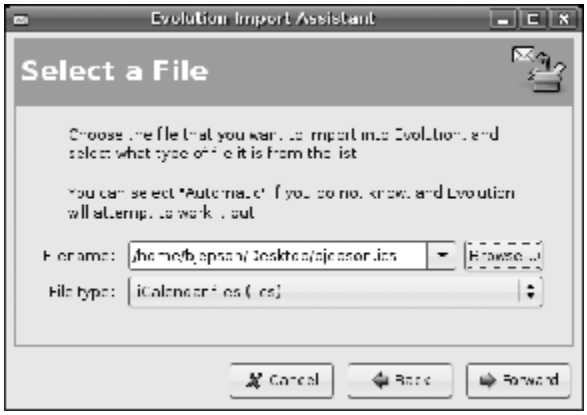
Your Little Black Book

To move your appointments and contacts over, it's best if you can get your data into the vCalendar or iCalendar format (for calendars) and the VCard format (for contacts). If you can do this, it will be simple to import into an application such as Evolution and Thunderbird.

Outlook will let you export items one at a time, but this is too tedious for importing many contacts or calendar items. The free OutPod (http://outpod.stoer.de/) is designed to export Outlook items to an iPod, but to accomplish this, it uses the vCalendar and iCalendar formats to store the data. To use it, simply run *OutPod.exe* and navigate to your list of calendar items or contacts (Outlook will probably ask for your permission to let OutPod access its data). Click in the list of contacts or calendars and press Ctrl-A to select all. Then, click OutPod's Outlook menu and choose "Save Selected Items in One File."

Copy the *.vcf* (VCard), *.ics* (iCalendar), or *.vcs* (VCalendar) file over to your Ubuntu system, and import it into Evolution or Thunderbird (addresses only). For Evolution, choose File→Import to bring up the Evolution Import Assistant, and choose to Import a Single File when prompted. Select the file you want to import (you may need to specify the file type), as shown in Figure 1-5, and then continue through the Import Assistant to import the data.

Figure 1-5. Importing an iCalendar file



Another utility you may find useful for exporting calendars from Outlook is the free Outlook to iCal Export Utility (http://outlook2ical.sourceforge.net).

If you're using Thunderbird on Windows, you can export contacts using the LDAP Data Interchange Format (*.ldif*). On your Windows machine, launch Thunderbird, select Tools→Address Book to bring up your list of contacts, and then select Tools→Export to export all contacts. When the Save As dialog appears, choose LDIF. You'll be able to import LDIF into Thunderbird and Evolution.

Am I Forgetting Anything?

As a last check, open up your Windows Control Panel, go to Add/Remove Programs, and review the list of programs. Do you see any that store their data in a nonstandard location? For example, anything with a keychain, such as GNU Privacy Guard or the PuTTY SSH suite, might be keeping some keys you can't live without.

GNU Privacy Guard keeps its files in the *gnupg* subdirectory of the user's *Application Data* folder. `APPDATA` is an environment variable that expands to the user's *Application Data* directory, so issuing the command:

```
dir %APPDATA%
```

in a Windows Command Prompt will reveal *gnupg*'s location. PuTTY keeps its keys wherever you decided to put them. If you've been using Pageant, select Add Key, and it should pop up with an Open File dialog set to the most recent directory you used.

If you're a Cygwin (http://www.cygwin.com) fan, don't forget that it keeps its home directories in a separate location (usually *C:\\cygwin\\home*). You may have all kinds of important documents and dotfiles kicking around there!

You Could Just Hire a Mover

There are a few third-party applications available to help move from Windows to Linux, but not all of them support Ubuntu. If you have a lot of machines to migrate, you might want to check one of these out:

- Desktop Migration Agent (http://www.alacos.com/linux.html)
- Progression Desktop (http://www.versora.com/)
- MoveOver (http://www.resolvo.com/products/moveover/index.htm)
- LSP (http://www.das.com.tw/elsp.htm)

Brian Jepson

Hack 8. Install Ubuntu on a Mac

Install Ubuntu and Mac OS X on the same machine for the best of both worlds.

Apple hardware has some quirks and differences compared to normal PC machines, but is generally of very high quality and can make a great Ubuntu machine. And with a little extra work, you can set it up to dual-boot for those times you still need to use Mac OS X.

First, get your hands on a copy of the PPC installer for Ubuntu. You can find instructions on downloading and burning Ubuntu disk images in “Install Ubuntu” [Hack #5].

Reinstall Mac OS X

If you want to run your computer dual-boot with both Mac OS X and Ubuntu installed at the same time, you will need to reinstall Mac OS X so that you can repartition your disk safely. If you intend to go for a straight Ubuntu system and don’t need Mac OS X anymore, you can skip ahead to “Install Ubuntu.”

After backing up any documents or data already installed, put the Mac OS X install CD into the CD-ROM drive and reboot. As the computer reboots hold down the C key to force it to boot from the CD and start the Mac OS X installer. Once the installer loads, you will be presented with a screen to select your preferred language. Don’t select a language yet; instead go to Installer→Open Disk Utility to open the Mac OS X Disk Utility. On the left of Disk Utility is a pane listing disks and volumes, so select the hard disk on which you intend to install Mac OS X and Linux. Next, click the Partition tab at the top of the right pane. Under Volume Scheme, click the drop-down menu labeled Current and select “2 Partitions.” The first partition is going to become your Ubuntu partition, and the second will be used by Mac OS X, so click the central divider between the partitions and drag it to adjust the relative partition sizes to suit your requirements, keeping in mind that Mac OS X requires a minimum of 1.5 GB to install.

Next, click the second partition and select “Mac OS Extended (Journaled)” as the format. The format of the first partition doesn’t matter because you’ll be replacing it anyway when Ubuntu is installed.

Once you are happy with the settings, click the Partition button near the bottom right and confirm that you want to go ahead. This step will delete all the files on the hard drive.

Once the partitioning is complete, select Disk Utility→Quit Disk Utility to return to the language-selection screen. Click Continue and follow the usual Mac OS X installation steps until you get to Select Destination, at which point you will have two partitions to choose from. Select the second and proceed with the rest of the Mac OS X installation.

Install Ubuntu

Put the Ubuntu PPC install disk in the CD-ROM drive and (re)start your computer, holding down the C key as the machine boots to force it to start up from the CD. Follow the usual installation prompts until you get to disk partitioning: here, select "Manually edit partition table" to view the partitions created by Mac OS X. This is where you'll probably get a bit of a shock, because you'll discover that Mac OS X didn't just create two partitions as it claimed—it probably created about ten. Most of them will be very small partitions that are used by the OS as part of the boot process, so you need to leave them alone, but you should also see the large partition you left aside for Ubuntu during the Mac OS X installation process. It will most likely be marked as an "hfs+" partition, so select it and press Enter. Arrow down to "Delete the partition" and press Enter, and you will then be returned to the partition table, which will now have a large area marked FREE SPACE.

AUs: Is it worth mentioning in the prev. paragraph what partition you should delete if you're just planning to have a straight Ubuntu system, as mentioned on the previous page in the first paragraph under "Reinstall Mac OS X"?

At this point, you could create the partitions you need manually, but the Mac OS bootloader has some unusual requirements, so it's simplest to let the installer figure things out for itself. Select "Guided partitioning," and this time select "Use the largest contiguous free space" to have three new partitions created for you: one for the bootloader, one as your root partition, and one as swap.

Select "Finish partitioning and write changes to disk," press Enter to proceed, and if the summary screen shows the changes you expected, select Yes. This is the point of no return when the disk is changed, so make sure you're happy with what is being done.

The installer will then create filesystems in the new partitions and continue with the regular Ubuntu installation with questions about your location and setting up an initial user.

Once the installer has finished, you will be asked to remove the install CD and restart.

When your Mac restarts, you will see a yaboot menu giving three options: press L or wait to boot Linux, press X to boot Mac OS X, or press C to boot from CD-ROM. The X option takes you straight to the regular Mac OS X startup process, while the L option takes you to another yaboot menu where you can type in boot options. Just press Enter to begin the Ubuntu startup process.

For the most part, everything should work smoothly. However, if you have an AirPort Extreme card, it may not be supported well by Ubuntu. One option is to simply use an Ethernet cable, or to install a PCI, PC Card, or USB Wi-Fi adapter that is supported by Ubuntu. You may also want to look into the Broadcom 43xx Linux Driver (http://bcm43xx.berlios.de/), which is an open source driver for the chipset used in most, if not all, AirPort Extreme cards.

Hack 9. Set Up Your Printer

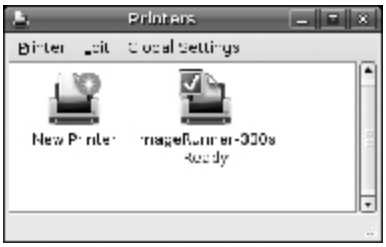
Get that printer connected, up, and running with Ubuntu.

Ubuntu uses CUPS, the Common Unix Printing System, to manager printers and print queues. CUPS can be configured using a variety of tools, including the GNOME CUPS Manager and its own built-in web interface that runs on port 631.

GNOME CUPS Manager

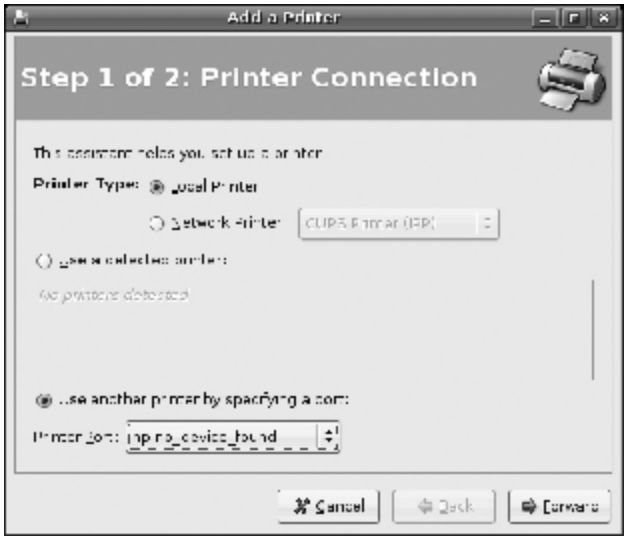
To launch GNOME CUPS Manager, select System→Administration→Printing, which will display a list of currently installed printers, shown in Figure 1-6, and give you the option of adding a new printer.

Figure 1-6. GNOME Cups Manager



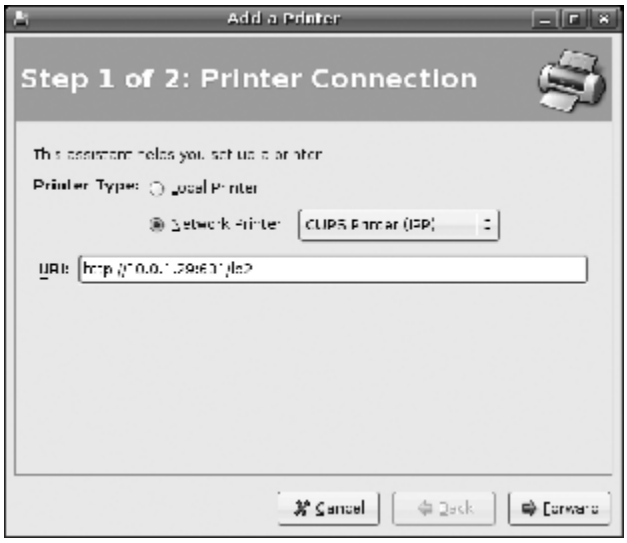
Double-click the New Printer icon to open the "Add a Printer" window. Here, you need to specify whether your printer is connected directly to your computer or is on your network (see Figure 1-7). If your printer is connected by USB, it's a good idea to use the instructions in "Mount Removable Devices with Persistent Names" [Hack #83] to assign a permanent name to your printer before going any further; otherwise, it will probably be assigned a different bus ID every time you plug it in, and you will be asked to configure it again each time!

Figure 1-7. Specifying local or network printer



If your printer is connected via the network, you will need to specify the protocol: IPP (Internet Printing Protocol), SMB (Windows printer sharing), LPD (Line Printer Daemon), or HP JetDirect. Each of those protocols in turn provides a number of configuration options to specify the printer identity. In Figure 1-8, the printer is connected to the second port on a network print server using IPP.

Figure 1-8. Configuring a network printer



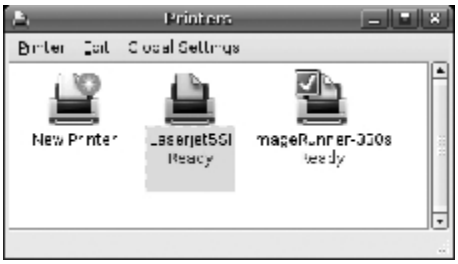
Specify the printer manufacturer and then select the model from the provided list, as shown in Figure 1-9.

Figure 1-9. Selecting a printer model

If your printer model isn't included in the list, you can try using one of the drivers listed under the Generic manufacturer, or if it's a PostScript printer, you can load the manufacturer's supplied PPD (PostScript Printer Description) file using the Install Driver button.

Apply the changes, and you'll return to the list of installed printers with your new printer listed, as shown in Figure 1-10.

Figure 1-10. Printer added



CUPS Web Interface

GNOME CUPS Manager provides quick access to basic options, but to really take control of CUPS, you should use the built-in web-management interface. Open your web browser and point it at http://localhost:631 to see a huge range of options.

Share Local Printers

You can allow other computers on your network to print to your locally connected printer by setting the `Shared` option.

Open */etc/cups/printers.conf* in a text editor to see the definition for your printers. Each printer is defined in a separate block, with configuration options applied in any order inside the block, and you should see an existing configuration option of:

```
Shared no
```

Change this to:

```
Shared yes
```

After you make changes to *printers.conf*, restart the CUPS scheduler:

```
$ sudo /etc/init.d/cupsys restart
```

Apply Print Quotas

CUPS implements a very simple quota system that allows you to restrict page count over a specified period, but it can only be applied per printer and not per user.

Open */etc/cups/printers.conf* in a text editor and add two options: `PageLimit`, which is the page count limit, and `QuotaPeriod`, which is the period in seconds for which you want it enforced. To apply a limit of 200 pages per week, you could set those options to:

```
QuotaPeriod 604800
PageLimit 200
```

One day is 86,400 seconds, a week is 604,800 seconds, and a month is 2,592,000 seconds.

You can also apply a `KLimit` option, which is the amount of data in kilobytes that the printer is allowed to receive. This can be useful if you want to prevent people sending extremely large print jobs to specific printers.

Remember to restart CUPS after making changes to *printers.conf*.

One possible hack to get around the per-printer limitation of quota settings is to configure the same printer multiple times, each with a different name and access controls but all communicating with the same device. The quota for each "printer" will then be tracked and enforced individually, so you could have each user connecting to a different virtual printer to enforce individual quotas.

Hack 10. Install Ubuntu on an External Drive

You can, in fact, install, boot, and run Ubuntu completely from a FireWire, USB, or other external drive, but it does require some special steps. This hack walks you through the process from start to finish.

In the process of working on this book, I realized one disadvantage to using a laptop as my primary computer: it is much more difficult to swap out hard drives for test systems. I wanted to set up an Ubuntu system so that I could test various hacks on a vanilla install, but I didn't necessarily want to repartition and install on my main laptop hard drive if I didn't have to. The solution was to install and run Ubuntu from an external USB drive I had; that way, the regular system stayed intact but I could boot Ubuntu whenever I wanted.

Unfortunately, this sort of install does not automatically work without some tweaking due to a few different reasons:

- By default, the *initrd* (initial ram disk) file that Ubuntu uses does not contain all of the drivers you need to boot from a removable drive. Your BIOS will find the drive fine (provided it supports booting from removable drives), but once the kernel loads, Linux won't be able to see or mount the drive to continue the boot process.
- Even if the *initrd* has the appropriate drivers, it takes a few seconds for the kernel to load these modules and detect your removable drive before it tries to use it. During this time, the system will likely try to boot and will not be able to find the removable drive because it hasn't finished configuring.
- The Ubuntu installer is very handy in that it tries to detect other OSes you might have installed on the system and will provide GRUB menu entries for each OS. Unfortunately, this means that it will set up any OS you have on the internal hard drive as being on the first BIOS drive, with the removable drive being second (or third or fourth if you have other drives on the system). When the BIOS boots from the removable drive, it will configure it as the first drive on the system, which will confuse GRUB.

In this hack, I discuss how to fix each of these problems so that you can install and boot Ubuntu from a removable drive.

Set Up the Partitions

The first step is to start the Ubuntu install process as you would with any other system, so read through "Install Ubuntu" [Hack #5] until you get to the section about partitioning a drive. When Ubuntu gets to the disk-partitioning tool, note that by default it will probably pick any internal IDE or SCSI drive currently in the system. If your system uses an IDE drive, you can choose your external drive by selecting the SCSI drive the system has detected. The line will probably refer to a disk called "SCSI (0,0,0) (sda)." If you already have a SCSI hard drive in the system, it will be a bit more difficult to locate the USB drive, but chances are it will be the last SCSI drive on the system.

Be absolutely sure you pick the correct drive in this phase, because Ubuntu will format and repartition the drive you choose and wipe out any data that might have been there. If you are uncertain which drive is the appropriate one, boot from the Ubuntu Live CD and confirm which device names (*sda*, *sdb*, etc.) it assigns the different drives on your system.

Install GRUB

Once you choose the correct drive to format, continue with the Ubuntu installation process up until it gets to the GRUB bootloader stage. Here, you will be asked whether you want to load GRUB to the MBR of the internal hard drive. You *don't* want to do this because it will overwrite any bootloader you are currently using on the system. Instead, say no and then specify */dev/sda* (or whatever Linux device was assigned to your removable drive) in the next screen that appears in order to install GRUB directly on the removable drive.

Use Chroot

Next, complete the Ubuntu installation up until when it prompts you to select Continue to reboot the system. Before you reboot, you will need to make some tweaks to the system. The Ubuntu installer actually provides a basic console you can use to run a few limited commands on the system. Hit Alt-F2 to switch to this console, and then hit Enter to activate it.

Now you need to prepare the removable disk so that you can chroot into it and change some files. The removable drive will actually be mounted under the */target* directory, and the first step is to mount the special */proc* filesystem within that drive:

```
# mount -t proc /target/proc
```

Now you can use the *chroot* tool to turn the */target* directory into the effective */* partition on the system. This way, you can run commands as though you had booted off of that drive:

```
# chroot /target
```

Tweak Initrd

Once inside the *chroot* environment, the first thing to do is to add the modules Linux uses to make your removable drive accessible to the *initrd*. The */etc/mkinitramfs/modules* file lets you configure extra modules to add to an *initrd*, so use your preferred console text editor to edit this file. If you don't have a preferred console text editor, just use *vim* (if you are unfamiliar with *vim*, check out "Edit Configuration Files" [Hack #74] for a *vim* primer):

```
# vim /etc/mkinitramfs/modules
```

Once this file is opened, move to the very bottom of the file and add the following lines and then save and close the file:

```
ehci-hcd
usb-storage
scsi_mod
sd_mod
```

If your removable drive is connected via IEEE1394, also add the following lines:

```
ieee1394
ohci1394
sbp2
```

and for any other devices, simply add the modules they need to this file.

With the correct modules configured, the next step is to set up a *initrd* so that it will wait a number of seconds before continuing to boot. That way, Linux has time to detect and configure the removable drive. Open */etc/mkinitramfs/initramfs.conf* in a text editor:

```
# vim /etc/mkinitramfs/initramfs.conf
```

Now add a new configuration option to the very top of the file so that Linux will wait for a few seconds before finishing the boot process:

```
WAIT=10
```

In my experience, 10 seconds is enough time for Linux to load my USB drive, but feel free to change this to a longer or shorter value if you need to. Save your changes and close the file.

Now you are ready to re-create an *initrd* file incorporating the new settings using the *mkinitramfs* tool:

```
# mkinitramfs -o /boot/initrd.img-
                    2.6.15-16-386
               /lib/modules/
                    2.6.15-16-386
```

Change the *initrd.img* and */lib/modules* paths to match the kernel version included in your Ubuntu install CD.

Update GRUB

The final step is to change a few settings in the GRUB configuration file. The Ubuntu installer sets up the external device as `(hd1)` (or second BIOS drive), but you need to change that to `(hd0)` because the drive will be the first BIOS drive in the system when the BIOS boots from it. Open the GRUB *menu.lst* file with a text editor:

```
# vim /boot/grub/menu.lst
```

and find the lines that refer to the GRUB root device. They will look something like the following:

```
## default grub root device
## e.g. groot=(hd0,0)
# groot=(hd1,0)
```

Change the last line to refer to `hd0` instead:

```
## default grub root device
## e.g. groot=(hd0,0)
# groot=(hd0,0)
```

Next, find the section in the file that refers to different Ubuntu kernels. It should look something like the following:

```
title       Ubuntu, kernel 2.6.15-16-386
root        (hd1,0)
kernel      /boot/vmlinuz-2.6.15-16-386 root=/dev/sda1 ro quiet splash
initrd      /boot/initrd.img-2.6.15-16-386
boot

title       Ubuntu, kernel 2.6.15-16-386 (recovery mode)
root        (hd1,0)
kernel      /boot/vmlinuz-2.6.15-16-386 root=/dev/sda1 ro single
initrd      /boot/initrd.img-2.6.15-16-386
boot

title       Ubuntu, memtest86+
root        (hd1,0)
kernel      /boot/memtest86+.bin
boot
```

Now change all of the references to `hd1` to `hd0`:

```
title       Ubuntu, kernel 2.6.15-16-386
root        (hd0,0)
kernel      /boot/vmlinuz-2.6.15-16-386 root=/dev/sda1 ro quiet splash
initrd      /boot/initrd.img-2.6.15-16-386
boot
```

```
title       Ubuntu, kernel 2.6.15-16-386 (recovery mode)
root        (hd0,0)
kernel      /boot/vmlinuz-2.6.15-16-386 root=/dev/sda1 ro single
initrd      /boot/initrd.img-2.6.15-16-386
boot

title       Ubuntu, memtest86+
root        (hd0,0)
kernel      /boot/memtest86+.bin
boot
```

If Ubuntu has detected and configured other OSes and you want the ability to choose them as well, simply repeat the same changes to the *root* config option for each OS—only change `hd0` to `hd1`. Then save your changes and close the file.

Now you can leave the chroot environment, so type `exit` in the console and then hit Alt-F1 to return to the main Ubuntu install console. Now you can select Continue to reboot the machine into your new install.

Keep in mind that most computers won't boot from a removable drive by default if a CD-ROM or other hard drive is present. Some BIOSes let you configure which device to boot from via a special key at boot time (such as F12). In other BIOSes, you may have to hit Esc, F2, or Del to enter the BIOS and configure the boot device order.

Hack 11. Install From a Network Boot Server

Boot your computer directly off a network server and install Ubuntu without using a CD.

Most modern computers have the ability to search the network for a boot server and load the operating system from it without using a local hard disk. This feature is typically used to boot thin clients that may not contain a hard disk at all, but you can also use it as a clever way to start the Ubuntu installation process without needing an install CD. This hack is perfect if you want to install Ubuntu onto a subnotebook with no CD-ROM drive or need to set up a large number of computers for a cluster, lab, or server farm.

Prepare The PXE Boot Server

The first step is to prepare the PXE boot server that will dish up the Ubuntu install image to your client. The easiest way to set this up is with an existing Linux server you have kicking around.

This boot server stores the install image and provides DHCP and TFTP (trivial FTP) services so that computers on the network can find and load the image when they start up. The whole process is triggered by the client connecting to the DHCP server and receiving special instructions telling it to fetch its boot image from the TFTP server instead of the local hard disk.

Configure DHCP

If you don't already have a DHCP server on your network, start by installing the *dhcp-server* package on the machine that will be your PXE Boot server:

```
$ sudo apt-get install dhcp-server
```

Then edit */etc/dhcp3/dhcpd/dhcpd.conf* and add a stanza similar to this:

```
host pxeinstall {
  hardware ethernet 00:00:00:00:00:00;
  filename "pxelinux.0";
}
```

Substitute the hardware MAC address of your client's Ethernet card in place of the string of zeros. Strictly speaking, you don't need the `hardware` line at all, but if you include it, your DHCP server will serve up the boot image only to that specific machine, so you won't need to worry about other random machines picking it up and reinstalling Ubuntu over their existing systems. On the other hand, if you're going to do installs on a lot of machines, you can just leave that line out, and every machine that netboots will be able to run the installer. Once you have updated the config restart the DHCP server:

```
$ sudo /etc/init.d/dhcpd restart
```

Configure TFTP

Install a TFTP server:

```
$ sudo apt-get install tftpd-hpa
```

Check */etc/inetd.conf* and make sure it has a line like this:

```
tftp    dgram   udp     wait    root  /usr/sbin/in.tftpd /usr/sbin/in.tftpd □
-s /var/lib/tftpboot
```

Next, restart *inetd*:

```
$ sudo /etc/init.d/inetd restart
```

Now you need to fetch the *netboot* install image (type the following all on one line):

```
$ sudo lftp -c "open http://archive.ubuntu.com/ubuntu/dists/dapper/main/installer-i386/current/images/; mirro
```

You will then have a directory called *netboot* with a number of files in it that need to be placed where the TFTP server can find them. The *inetd* config line listed earlier shows the path that *tftp-hpa* uses to store boot images—typically */var/lib/tftpboot*—so copy the files there and extract the boot image:

```
$ sudo cp -a netboot/* /var/lib/tftpboot
$ sudo cd /var/lib/tftpboot
$ sudo tar zxf netboot.tar.gz
```

You can even use a Windows machine as the boot server by installing a TFTP package such as Tftpd32 and placing the Ubuntu install image into the TFTP server root directory. Tftpd32 also includes a built-in DHCP server that you can use to specify the name of the install image. You can download Tftpd32 from http://tftpd32.jounin.net/.

Setting up a TFTP server is even easier in Mac OS X because *tftp* is already installed. Just run:

```
$ sudo mkdir -p /private/tftpboot; /sbin/service \\
                                              tftp start
```

and put your install image in place. You will, however, need to install and configure a DHCP server yourself unless you're running Mac OS X Server.

Boot the Client

In order to start the install process, your client machine needs to be able to perform a PXE network boot. Most modern machines can do this directly with a BIOS setting, although some older machines may need a special netboot floppy image to start the process.

Start up your client machine, use whatever key sequence is necessary to enter the BIOS setup menu, and locate the setting for boot devices. (The key sequence to enter the BIOS is usually something like F2 or ESC.) Set the first boot device to be PXE Boot, Network Boot, or the equivalent; save and exit the BIOS, and let the machine boot up again.

Some computers will display a boot menu when you press F12, so you can choose the boot device on the fly without having to modify your BIOS setting.

This time, you should see your machine report that it's looking for a DHCP server before it gets to the stage of trying to boot off the hard disk. After being given an IP address, it will report that it's searching for a PXE boot image. A couple of seconds later, you should see the Ubuntu installer splash screen, and after that the installation can proceed exactly as normal—except that all packages will be fetched directly from an Ubuntu mirror rather than a local CD-ROM.

Hack 12. Submit a Bug Report

All software has bugs, and Ubuntu is no exception. Here's how you can help improve Ubuntu by submitting a bug report.

It's an unfortunate rule of computing: all software has bugs. The Ubuntu developers and folks at Canonical have done their best to minimize the amount of bugs and their impact in the latest release of Ubuntu, but they can't catch everything. However, one of the major advantages of open source software is that you have an opportunity to help improve the software by filing a bug. The process of filing a bug is surprisingly easy and, despite the name, can be a rather fun and interactive process.

Getting Ready to File the Bug

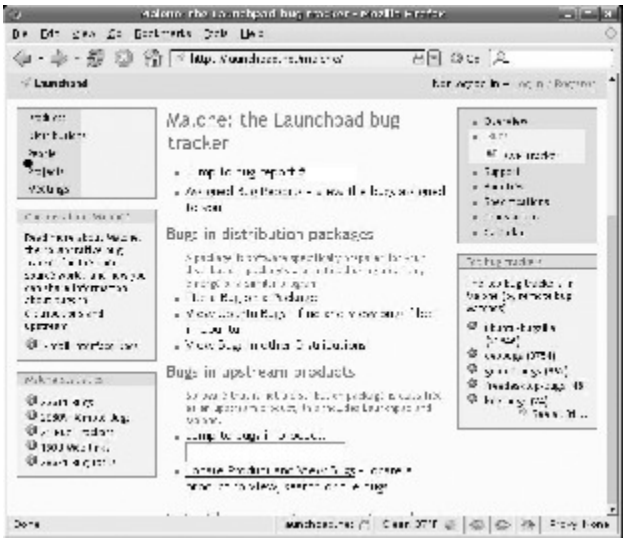
Before you actually go ahead and file a bug, you should run through a little checklist to better assist you in the process. The key thing to remember is that every piece of information you can embed in the bug report will help the people fixing your bug. These people may not have the same hardware as you, and there may be other difficulties in reproducing your bug, so every clue you can provide will help them in solving the mystery.

First, figure out what your problem is, in plain language. Ask yourself what's broken and what the proper behavior should be. Also, is there anything the software is doing that it shouldn't be doing? If you can capture logs or output from a terminal, save that information—you can attach it to the bug. If you've got knowledge on how to attach a debugger to your process, you may want to include output from that as well. Ensure you've got the package name of the piece of software you're having trouble with. Save all this information so you can have it handy when it's time to file the bug report.

Creating a Malone Account

Ubuntu's method of filing a bug report is via a web-enabled application called Malone (see Figure 1-11). Malone is part of Launchpad (https://launchpad.net/malone/), and the unique thing about Malone compared to other bugtrackers is that Malone tracks not only Ubuntu bugs, but upstream bugs as well as bugs in other distros. This helps to enable another benefit of open source software: the fact that "given enough eyes, all bugs are shallow". In a nutshell, this means that if one distro pinpoints and fixes a bug, all other distros that use Malone can see the fix, and everyone benefits.

Figure 1-11. Your first visit to Malone



On your first visit to Malone, you'll need to create an account for yourself so you can post to the bug database. Simply click on the Log In/ Register link in the upper-righthand corner and follow the instructions to register an account.

You can search the Malone bug database without creating an account, but posting new information to the database requires an account.

Searching for Your Bug

You've collected your data and created your account: now it's almost time to file your bug. There's just one thing left to do: search for your bug. Yes, that's right: you should always search the bug database for your bug (or one just like it) before you go ahead and create a new one. If you search for a bug with similar symptoms and criteria as your bug, there's a good chance you'll find a bug report already there. You can check and see if the behavior of the bug is the same, if the developers are stalled waiting for a piece of information, or if the bug's already been fixed and a patch is available.

Additionally, if you do see a bug report that resembles your bug, you can add your comments and information to that bug. That will help consolidate things and let the developers know that multiple people are experiencing that issue. Nothing is more frustrating for developers than duplicate bug reports (well, not entirely; poor bug reports are even worse!).

In Figure 1-12, I've searched for a *gnome-power-manager* issue that I happened to run into. As it happened, there was a pre-existing bug already opened, and the developer was awaiting information. I was able to add my own comments to the bug and give the developer some of the information he was looking for. That's open source in action!

Figure 1-12. Searching for a bug

Filing your Bug Report

If your bug isn't already in the bug database, then it's time to report it. Log in to Malone if you haven't already, and go to the "Report a bug about a package" page (*https://launchpad.net/malone/bugs/+package*). Select Ubuntu from the drop-down menu and enter the package name in the appropriate field. Next, enter the bug summary. You'll want to be as descriptive as you can be in one sentence. Don't say, "gnome-power-manager is broken" or, worse, "battery icon doesn't work." Either of those two summary lines will ensure that your bug is rejected. A more appropriate summary line would say, "gnome-power-manager icon does not show AC power status."

Next, fill out the description of the problem. Be as objective and descriptive as you can, and include all the notes I mentioned earlier. Take your time, and be sure to include every detail. I can't stress enough the importance of including not only what's wrong, but what the software is doing right. Include any debug information you may have gathered.

Once you've done that, you can submit your bug by clicking the Add button. There's a checkbox on the site to keep the contents of the bug confidential, so you'll want to check that box if there's a possible security risk exposed by this bug. However, 99.9 percent of the time the bug won't involve a security risk, and you can safely leave this box unchecked.

Now your bug's filed and in the system! Come back to your bug's page periodically to see if a developer has picked up the problem and made any progress on it.

Hack 13. Use the Command Line

Put your mouse down for a second, pop open a terminal window, and fall in love with the shell all over again.

If you are used to Windows or Mac desktops, the command line might seem like a foreign thing to you. Typing commands into a window might seem, well, arcane. But even though Linux has really progressed on the desktop, there's still a lot of power you can wield at the command line. If this is your first time with a terminal, this hack will guide you through some command-line basics.

Throughout this book, you'll find a number of places where you'll need to prefix commands with `sudo`. The sudo command [Hack #67] allows you to temporarily execute a command with different user privileges and is frequently used when you need to add or remove software [Hack #54] from the command line.

The first step is to launch a terminal. Click Applications→Accessories→Terminal to start the default Gnome Terminal program.

Navigate the Filesystem

Now that the terminal program is open, you can navigate the filesystem. By default, terminals will open into your home directory, so one thing you might want to do is see what files are currently in your home directory. The *ls* command displays all the files in the directory you specify (or in the current directory if you don't list a directory):

```
greenfly@ubuntu:~$ ls
Desktop
greenfly@ubuntu:~$ ls Desktop/
screenshot1.png  screenshot2.png
```

In the first command, I listed all of the files in my home directory. In this case only my *Desktop* directory existed. In the second example, I listed the contents of my *Desktop* directory, where I had two screenshot images.

To change to a different directory, use the *cd* command, followed by the directory to change to:

```
greenfly@ubuntu:~$ cd Desktop/
greenfly@ubuntu:~/Desktop$ ls
screenshot1.png  screenshot2.png
```

Notice that the terminal prompt changed in the second line to show that I am currently in the *Desktop* directory. You can also use the *pwd* command to see where you currently are:

```
greenfly@ubuntu:~/Desktop$ pwd
/home/greenfly/Desktop
```

The ~ symbol is shorthand in Linux for your user's home directory. If you type `cd ~` you will automatically change back to your home directory. It saves you from having to type out `cd /home/` *`username`*.

Rename and Delete Files and Directories

To create a directory from the command line, type the *mkdir* command followed by the name of the directory to create:

```
greenfly@ubuntu:~$ mkdir test
greenfly@ubuntu:~$ ls
Desktop  test
```

Use the *mv* command to move a file or directory to a different directory, or to rename it in its current directory. To rename the *test* directory I created to *testing* I would type:

```
greenfly@ubuntu:~$ mv test testing
greenfly@ubuntu:~$ ls
Desktop  testing
```

If I wanted to move the *testing* directory inside my *Desktop* directory, I would just specify the *Desktop* directory as the second argument:

```
greenfly@ubuntu:~$ mv testing Desktop/
greenfly@ubuntu:~$ ls Desktop/
screenshot1.png  screenshot2.png  testing
```

The *rm* command removes files, and *rmdir* removes directories. Just use the commands followed by the files or directories to remove, respectively:

```
greenfly@ubuntu:~$ rm Desktop/screenshot1.png Desktop/screenshot2.png
greenfly@ubuntu:~$ ls Desktop/
testing
greenfly@ubuntu:~$ rmdir Desktop/testing/
greenfly@ubuntu:~$ ls Desktop/
greenfly@ubuntu:~$
```

You can also remove a directory and all files and directories inside of it by running `rm -r`, followed by the name of the directory.

Be careful when you recursively delete a directory with this command that you do in fact want to remove all of the files within. Once removed via the command line, there's no trash bin to retrieve them from.

File Globs and Tab Completion

There are two major time-savers when dealing with long files on the command line: file globs and tab completion. *File globs* are symbols you can use as wildcards in the place of a filename. You can substitute the `?` symbol for any single character in a filename, and `*` for any number of characters in a filename. For instance, say we had three files: *foo*, *bar*, and *baz*. If I wanted to delete both *bar* and *baz*, I would type:

```
greenfly@ubuntu:$ rm ba?
```

The `?` matches both `r` and the `z` at the end of the filename. If I wanted to remove all files that started with the letter `b`, I would type:

```
greenfly@ubuntu:$ rm b*
```

Tab completion is another time-saver on the command line. If you start to type a command and then hit the Tab key, the shell will automatically attempt to complete the name of the command for you. In the case that more than one command matches what you have typed so far, hit Tab an extra time, and you will be shown all of the options that match:

```
greenfly@ubuntu:~$ gnome-cups-<Tab><Tab>
gnome-cups-add      gnome-cups-icon     gnome-cups-manager
```

Tab completion also works for files and directory names. Just type the first part of the filename and hit Tab, and the shell will fill out the rest for you.

Once you are finished with a terminal, you can close the window like any other window, or alternatively you can type `exit` on the command line.

Hack 14. Get Productive with Applications

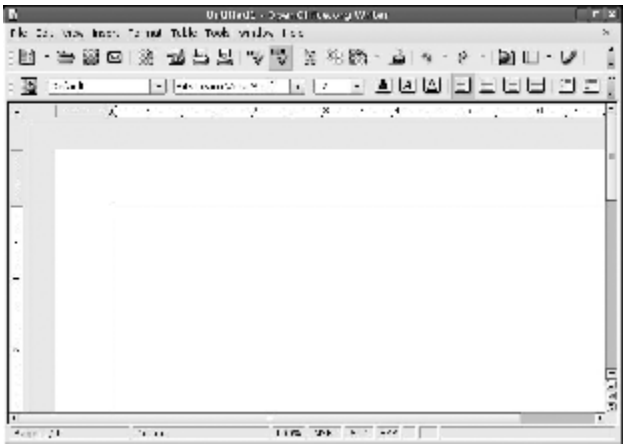
Even the coolest OS is useless without programs to run on it. Ubuntu ships with lots of built-in applications, including many Linux-based counterparts to some of the more common Windows applications.

Ubuntu Linux is great for lots of reasons, but one of its strengths is the amount and type of applications the operating system ships with. If you are a new Ubuntu user, you're probably familiar with Windows and the application suites on that OS. Here's an introduction to the Ubuntu and open source analogues to those Windows-based applications you may use all the time. Feel free to play with these applications; they all have excellent help-based documentation if you get stuck.

Office Suite

One of the most used Windows applications is Microsoft Office. Ubuntu ships with an office suite that's similar, called OpenOffice.org (sometimes called OOo). OpenOffice.org can even read and write Microsoft Office files, so you won't be left out of the loop when a friend or coworker emails you a document attachment. OpenOffice.org includes a word processor (Writer), shown in Figure 1-13; a spreadsheet (Calc); and a presentation tool (Impress). It also includes Math, a scientific formula editor; Draw, a flowchart and drawing program; and Base, a basic database. To access any of these programs, click on the Applications menu and select Office. All the OpenOffice.org applications are there. You can find out more about OpenOffice.org from its web site (http://www.openoffice.org).

Figure 1-13. OpenOffice.Org Writer



Graphics and Photo Editor

Adobe Photoshop is probably the most known photo and graphics editor. However, did you know that Ubuntu ships with a world-class graphics editor? This piece of software is called the GIMP (GNU Image Manipulation Program), and it's accessible from the Applications menu, in the graphics section. The GIMP (shown in Figure 1-14) includes many built-in filters for image manipulation and is scriptable using the programming language Python. It's so powerful, it's being used in some Hollywood studios! Surf over to the GIMP's web site (http://www.gimp.org) if you need more information.

Figure 1-14. The GIMP

Web Browsing

Web browsing in Ubuntu is handled by the super-popular Mozilla Firefox (http://www.mozilla.org/firefox) web browser (shown in Figure 1-15). Firefox comes in Windows and Macintosh versions, so you may have encountered it before. It has received much acclaim from the

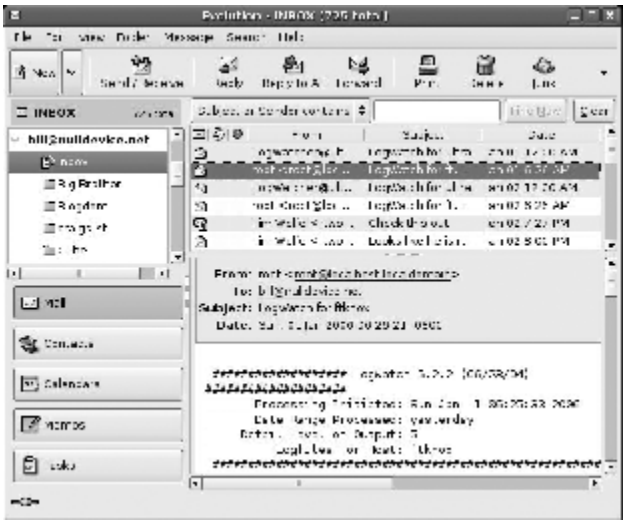
computer community in general—so much, in fact, that there's not much to add. To fire up Firefox, you can either click on the little world icon in the top panel, or you can run it via the Applications menu, in the Internet section.

Figure 1-15. Surfing with Firefox

Email

The default mail program in Ubuntu is Evolution (http://www.gnome.org/projects/evolution/). Evolution (shown in Figure 1-16) is much more than an email application, however. Like its counterpart, Microsoft Outlook, Evolution is designed to manage contacts and calendars in addition to email. It even includes an integrated junk-mail function to help you deal with ever-increasing spam issues. You can launch Evolution by clicking on the little envelope icon in the top panel, or by going to the Internet section of the Applications menu.

Figure 1-16. Reading mail with Evolution



Instant Messaging

Instant messaging has become a large part of the daily computing experience. Ubuntu's got you covered with the inclusion of Gaim (http://gaim.sourceforge.net), shown in Figure 1-17. Gaim is a cross-platform messaging client, able to communicate across the AOL, ICQ, MSN, Yahoo!, Google, and Jabber networks. It can even allow you to log into multiple IM networks at the same time, thanks to its multiple-account support.

Figure 1-17. Messaging via GAIM

ED will be supplying new image for Figure 1-17 that shows Gaim in action.

Media Player

Windows Media Player is the standard application for playing multimedia under Windows. Ubuntu ships with a similar program called Totem (http://www.gnome.org/projects/totem/), shown in Figure 1-18. Totem plays all manner of media files, music, and video files. It supports playlists and streaming media as well, so you can use it to listen to Internet radio stations.

Figure 1-18. Using Totem for media

Whatever need you've got, Ubuntu's probably got an application to fill it. If Ubuntu doesn't ship with it out of the box, it can probably be obtained via the Synaptic package manager or *apt-get*, thanks to the universe and multiverse repositories [Hack #60]. Have fun exploring the abilities of Ubuntu!

Chapter 2. The Linux Desktop

On its own, the X Window System (X11) isn't much of a friendly face. If you've ever run X11 without a window manager, you've no doubt seen it at its barest: a screen filled with a repeating crosshatch pattern, with an X for a mouse cursor. This simplicity does nothing to hint at X11's power, but once you've fired up GNOME, KDE, or any of many window managers available for Ubuntu, you start to see what it's all about.

This chapter takes you a little deeper into the GNOME and KDE environments, which are so much more than window managers. You'll also learn how to check out some more lightweight window managers in case you're after something simpler and less CPU-intensive.

Once you're settled into your desktop environment, you're going to want to get some work done. This chapter shows you how to install Java, which is needed by many applications, including some of the peer-to-peer (P2P) applications discussed herein. You'll also learn how to connect your handheld Palm or Pocket PC device to Ubuntu, work with remote file servers, and more.

Hack 15. Get Under the Hood of the GNOME Desktop

GNOME, the default Ubuntu desktop, is a powerful environment with a lot of features. Here is the information you need to quickly get up to speed on how to customize it.

Recently, the GNOME desktop seems to have lost some features. Looking around on mailing lists and reading people's blogs, you'll often find gripes about how some feature that was someone's personal favorite no longer exists. In reality, GNOME has far more features and configuration options available now than it ever has in the past—they're just hidden from sight, with users shown only the most commonly used options in the standard interface. This has the effect of making GNOME simpler and easier to use for the average person, but it also makes it a prime target for getting under the hood and tweaking the deskop to suit your own tastes if you're an advanced user and want everything to work just the way you like it.

Configuration Nirvana: the Configuration Editor

GNOME provides a central mechanism called GConf for storing user preferences on behalf of individual applications. Instead of writing out their own preferences files and then parsing them again to read values back in, applications can simply use the GConf API. This has a number of benefits, such as the ability to share preference settings among applications and to have preferences applied immediately to all running applications.

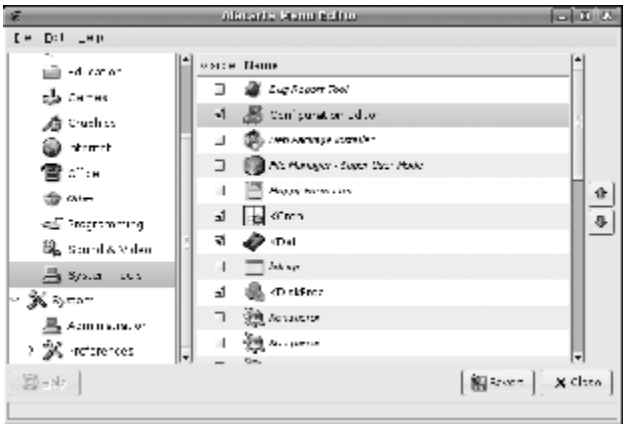
The GConf database is structured like a simple filesystem, containing keys organized into a tree hierarchy. Each key can be a directory that contains more keys, or it can have an actual value of its own. For example, the key */apps/nautilus/preferences* is a key that contains other keys (in a similar manner to a directory), and inside it is the */apps/nautilus/preferences/background_color* key with a default value of `#ffffff`. While keys are expressed as paths as in a filesystem, they don't actually exist on disk in that way: they are stored in an XML document, with the path representing the nested items within it.

GConf has several tools that you can use to directly browse, query, and manipulate the database using either a GUI or the command line. Configuration Editor provides a very nice graphical interface to the database, but it doesn't appear in Ubuntu's Applications menu by default, so you can launch it from the command line:

```
$ gconf-editor
```

Alternatively, you can edit the Applications menu and add it. Go to Applications→Accessories→Alacarte Menu Editor, select System Tools, and turn on Configuration Editor, as shown in Figure 2-1.

Figure 2-1. Edit the Applications menu



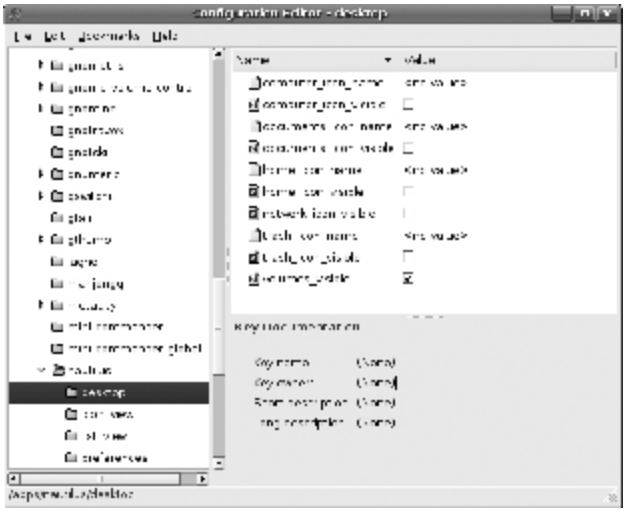
Once you close the Menu Editor and return to the Applications menu, you'll find a new entry for Configuration Editor under System Tools.

When you open Configuration Editor, you'll notice that there is no Save button and all changes you make are applied immediately—so be careful! Browse around to see what options you can set for various applications, including Nautilus (the GNOME file manager) and Metacity (the default window manager). To get started, here are some specific options you can tweak with the *gconf-editor*.

Icons on desktop

The default Ubuntu desktop configuration is totally empty: no trash can, no computer icon, no home icon. In Configuration Editor, go to */apps/nautilus/desktop* to see options to turn on or off a variety of desktop icons, as shown in Figure 2-2.

Figure 2-2. Enable desktop icons with gconf-editor



Empty trash without warnings

Emptying the trash generates an annoying confirmation message that has to be clicked, but you can suppress the confirmation by turning off */apps/nautilus/preferences/confirm_trash*.

And if you want to be able to bypass the Trash entirely and delete files by right-clicking on them, turn on *apps/nautilus/preferences/enable_delete* to add a "Delete" option to the right-click contextual menu. Now you can delete items immediately without sending them to the Trash first!

Open files with a single click

GNOME's default behavior is for a single-click to select files and a double-click to open them, but KDE and some other environments use a web-like "hot links" metaphor, in which files open on a single click. To enable the same behavior in GNOME, go to */apps/nautilus/preferences/click_policy* and replace the `double` value with `single`.

Scripting GConf

Automating changes to the GConf database is easy with tools such as *gconftool*, which is a command-line GConf client. Use it to read or set specific values in the database as well as explore its structure.

Recursively walk through parts of the database structure by specifying a starting point and using the `-R` (recursive read) option:

```
$ gconftool -R /apps/nautilus
```

Get the attribute type of a specific key using the `-T` (type) option:

```
$ gconftool -T /apps/nautilus/preferences/enable_delete
```

Read specific values by explicitly setting the key and using the `-g` (get) option:

```
$ gconftool -g /apps/nautilus/preferences/enable_delete
```

Write values by specifying the data type, using the `-s` (set) flag, the key, and the new value:

```
$ gconftool -t bool -s /apps/nautilus/preferences/enable_delete true
```

Writing values like this is a good way to demonstrate that changes really are immediate. Open up Configuration Editor and browse to */apps/nautilus/preferences/*. Then watch the "enable_delete" checkbox while you set and unset the value using *gconftool*.

Once you get started, you'll find that your imagination is the limit when it comes to scripting GConf! For example, you could write a script that pulls down a webcam image every ten minutes and then calls *gconftool* to set it as your desktop background. Make sure the image filename is different each time; otherwise, GConf won't see the change, and the background won't change:

```
$ gconftool -t string -s /desktop/gnome/background/picture_filename \\
             /home/jon/CamImages/Pic53332.jpg
```

Or you could set up a script to use XPlanet to generate an updated view of Earth and set it as your desktop background, and then call it from cron every 30 minutes. Install XPlanet:

```
$ sudo apt-get install xplanet xplanet-images
```

Then create a script to execute it. If you don't know the latitude and longitude of your city, you can probably find it (or a nearby city) in Wikipedia (http://www.wikipedia.org). Just adjust the following script to suit your location and screen resolution:

```
#!/bin/sh
rm -f /tmp/earth-*.jpg;
IMAGE=/tmp/earth-\Qdate +%s\Q.jpg;
nice xplanet -num_times 1 -output $IMAGE -geometry 1280x1024 \\
-longitude 96 -latitude 0;
gconftool -t string -s /desktop/gnome/background/picture_filename $IMAGE;
```

You can also use *gconftool* as a convenient way to add new URI handlers to GNOME. For example, to specify that you want *tel:* links to execute your Vonage client and initiate a call, you can run the following commands:

```
$ gconftool -t string -s /desktop/gnome/url-handlers/tel/command \\
            "bin/vonage-call %s"
$ gconftool -s /desktop/gnome/url-handlers/tel/needs_terminal false -t bool
$ gconftool -t bool -s /desktop/gnome/url-handlers/tel/enabled true
```

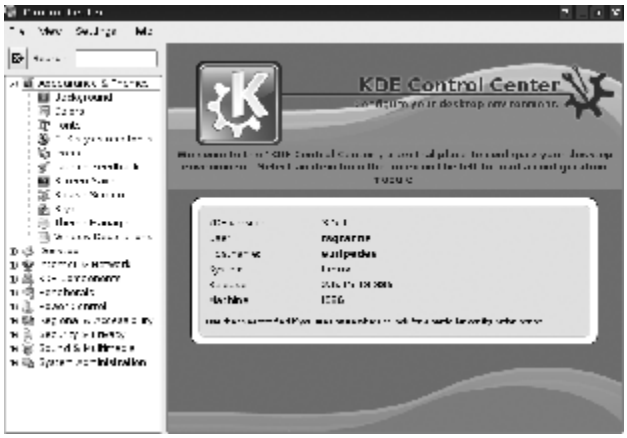
Hack 16. Tweak the KDE Desktop

Get up to speed with configuring KDE, the default desktop environment for the Kubuntu variant of Ubuntu.

While GNOME, the heart of Ubuntu, seems to be adopting an extremist policy of "simplify simplify simplify" that goes so far as to result in the father of Linux strongly criticizing it (see http://mail.gnome.org/archives/usability/2005-December/msg00021.html), KDE, the heart of Kubuntu, has sought to simplify without reducing features. Instead of simply hiding configuration options in the Windows Registry–like GConf or requiring that users know arcane key commands that serve to bring up important capabilities, both of which GNOME practices, KDE preserves the customizability and power that has garnered it fans all over the world, while streamlining options and increasing ease of use.

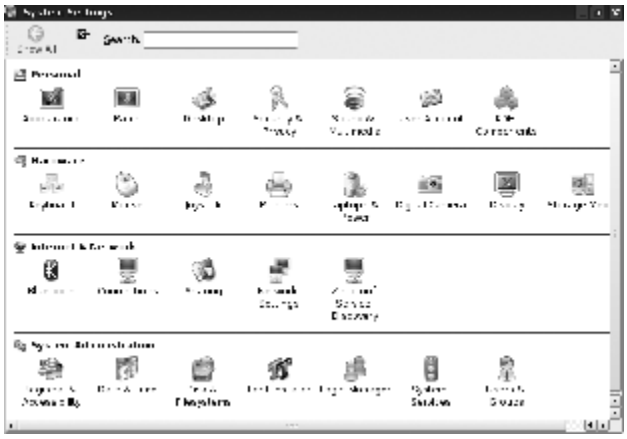
A prime example of this can be seen in KDE's evolution from the Control Center to System Settings. The Control Center allowed users to customize KDE in virtually infinite ways, but its layout was cluttered and confusing, as shown in Figure 2-3.

Figure 2-3. The old-fashioned KDE Control Center was pretty cluttered



The KDE developers responded to the criticisms they received and are transitioning to System Settings, which is found in Kubuntu (if, for some reason, it wasn't installed on your system, run the command `sudo apt-get install kde-systemsettings` to install it). As you can see in Figure 2-4, the layout is cleaner, easier to read, and more inviting. To start System Settings, click the K button and choose System Settings.

Figure 2-4. The new and improved KDE System Settings



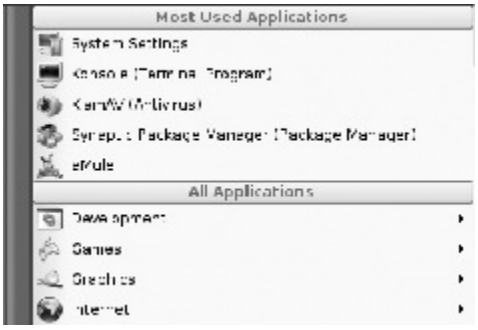
The infinite customizability is still present, but it's now more approachable for both experts and newbies. Instead of removing features or squirreling them away in byzantine ways, the KDE developers adopted an evolutionary path that improves usability without sacrificing power.

Right after I install Kubuntu, there are a few important changes I make using System Settings that I feel improve my use of KDE. Think of these as suggestions that you can try, but you really should explore the world of System Settings yourself to truly make your copy of Kubuntu your own.

Personal

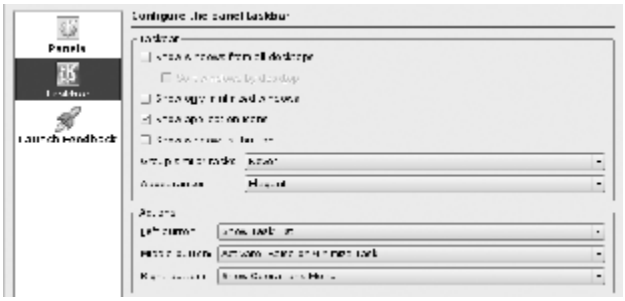
Click on the Panel button, and then on the Menus tab of Panels. Under QuickStart Menu Items, I like the fact that "Show the applications most frequently used" is selected, but in some cases, I've found that the number is set to 0. Change that to 5, and the five programs most frequently used in a session will show up at the top of the K menu, as you can see in Figure 2-5.

Figure 2-5. Show the five most used applications on the K menu



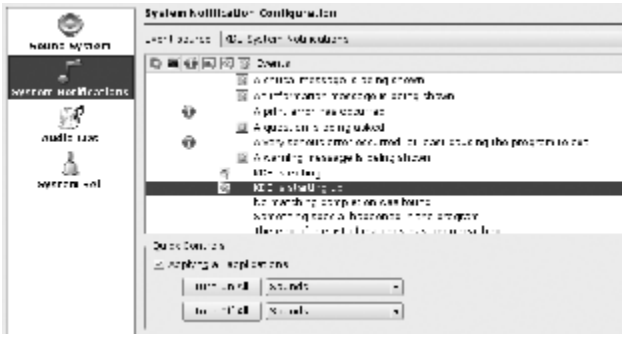
While also in Panel, select the Taskbar button. One of the so-called "innovations" that Windows XP introduced was *taskbar grouping*: if you have several windows from the same app open, they are grouped together as one button on the taskbar. To see the individual windows, select that taskbar button, holding down the mouse button, and a menu opens showing the various windows, which you can then pick from. Personally, I hate this feature. I want to see all open windows on a particular desktop, so I always change "Group similar tasks" from When Taskbar Full to Never. In addition, when I'm using multiple desktops, I don't want to see all the windows from all those desktops on the taskbar, which to me defeats the whole purpose of multiple desktops in the first place, so I uncheck "Show windows from all desktops" and "Sort windows by desktop." See Figure 2-6 for the changes I make.

Figure 2-6. Turn off taskbar grouping



Back in System Settings, move on to Sound & Multimedia→System Notifications. I hate it when my computer constantly beeps and boops at me; the noise that KDE makes every time a window opens or closes drives me especially batty. I check the box for "Apply to all applications" and then click the Turn Off All button next to Sounds, shown in Figure 2-7.

Figure 2-7. Turn off annoying system sounds



Still, I like the sound KDE makes when it's first loading and when it's shutting down, so I always go ahead and click in the Sound column (the fifth column) next to "KDE is exiting" and "KDE is starting up," so I can hear those tiny melodies.

Hardware

In an effort to be more "web-like," the default behavior in KDE to open files and folders is a single-click. While I love this behavior in Firefox, I don't like it so much in my file manager. To turn it off, select the Mouse button in the Hardware section and then, on the General tab, choose "Double-click to open files and folders."

System Administration

This one will help solve an annoyance that otherwise might cause U.S. users to smack their heads on their keyboards in frustration. In System Administration, choose "Regional and Accessibility," then "Country/Region and Language," and finally the "Other" tab. Change "Paper format" from A4 (which works great in Europe) to US Letter, the standard paper size in America. Now printing works again! And while you're there, change "Measure system" from Metric to Imperial, which is again what the U.S. uses. Of course, if your country's standards are different, here is where you change them.

When it comes to configuring KDE, I've just scratched the surface, but I hope this is enough to get your started, My advice is to click on every single item in System Settings and investigate what you find there. After you finish that, head over to http://www.kde-look.org, where you can download new wallpaper, themes, icons, styles, skins, screensavers, and splash screens to change how your copy of KDE appears. But KDE-Look.org isn't just about eye candy. You can also find service menus, software, improvements, and tips for changing how your KDE acts and behaves. It's a friendly, helpful community, and I know you'll find something fun and useful there. Linux is all about choice, and nowhere is that more obvious than when it comes to KDE and Kubuntu!

Scott Granneman

Hack 17. Switch to a Lighter Window Manager

GNOME and KDE are great, but they are a little heavy. If you're on an older system, or you just want a change of pace, you can use other window managers under Ubuntu, such as Fluxbox, XFCE, and Enlightenment.

If there's one thing that's great about Linux, it's choice. If you don't like a particular program, there's a pretty good chance that Linux has at least one other alternative. This even applies to your entire desktop environment. The desktop environment comprises a lot of different programs, such as a window manager (which handles drawing borders around your windows, moving them, and so forth), panels so you can launch programs, background-management programs, and more. The most popular of these desktop environments are GNOME and KDE. Ubuntu defaults to Gnome [Hack #15] as its desktop environment but also offers a Kubuntu alternative [Hack #16] that automatically defaults to KDE instead.

If you don't particularly like GNOME or KDE, you still have other options. Linux has a large number of window managers that you can use instead of GNOME or KDE, and all of the popular ones are available for Ubuntu. There are a number of reasons why you might want to give some of these window managers a try:

- Both Gnome and KDE need a fair amount of resources to run. Most of the alternative window managers require substantially fewer resources, so they might be attractive if you are using an older computer or if you just just want better performance out of your desktop.
- Alternative window managers often offer a totally different set of features and, in some cases, a different way to look at how to manage your windows. Some of these features include the ability to group windows into a single tabbed window (Fluxbox) or set up lots of fancy eye candy and control your windows' placement to a fine degree (Enlightenment).

If you want to stick with KDE or GNOME, there are some simple things you can do to lighten their resource usage. In KDE, run the program *kpersonalizer* (it's in the package of the same name) to reduce KDE's level of eye candy. In GNOME, use the Configuration Editor [Hack #15] to set */apps/metacity/general/reduced_resources* to `true`.

Even if you don't have a particular reason to try a different window manager, it doesn't hurt to install a few and see how they approach window management. You can easily switch back to your preferred desktop environment if you don't like them.

In this hack, I describe a few other window managers and how to install and use them in Ubuntu. There are hundreds of window managers I could cover, but here I will talk about three of the more popular alternatives to GNOME and KDE: XFCE, Fluxbox, and Enlightenment.

Generate Program Menus

The first step before you install a new window manager is to install and update a program to manage application menus, so that you can launch applications without the GNOME or KDE launchers. Use your preferred package installation tool and install the package called *menu*. Once the program is installed, open a terminal and update the current list of programs for this menu:

```
$ sudo update-menus
```

Change to Your New Window Manager

I will discuss how to install and use each of the different window managers, but since you will use the same method to change to each of them, I describe that first. Each of these window managers is integrated with the desktop manager Ubuntu uses (GDM by default, KDM for Kubuntu) and will add itself to the list of available sessions when you install it.

After you install a particular window manager, log out of your current desktop environment to get to the main login screen. Click on the Sessions button to see a list of available desktop environments and window managers, and select the window manager you'd like to try. After you log in, you will be presented with the choice to accept this choice permanently or just for this session. If you want to switch back, log out and then select your previous window manager from the list (GNOME under Ubuntu, KDE under Kubuntu).

Try XFCE

If you are interested in other window managers or desktop environments, probably one of the first to try is XFCE. XFCE (http://www.xfce.org) aims to be a lightweight desktop environment, so you will get many of the familiar features of a full desktop environment—such as a panel, desktop icons, and a taskbar—but with improved performance.

To install XFCE, use your preferred package-installation program to install the *xfce4* package. The desktop environment and many accompanying tools will be installed. XFCE has a number of other nonessential plug-ins and programs that you can install as well. Just use your package manager's search tool with the keyword `xfce` to show them all.

Once XFCE is installed, log out, choose the XFCE session, log in, and you will be presented with the default XFCE desktop (see Figure 2-8).

Figure 2-8. The default Ubuntu XFCE desktop

XFCE is organized into a panel at the bottom where you can launch common tools such as a terminal, XFCE's file manager *xffm*, a web browser, and other applications. To launch applications that aren't in the panel, right-click on the desktop to open the main menu. You can change a launcher's settings by right-clicking on it in the panel. You can also right-click on other parts of the panel to add new items, such as launchers, pagers, and other programs.

Along the top of the desktop is the taskbar, where you can see and switch between all open applications on the current desktop. Right-click on one of the applications in the taskbar to get extra options, such as the ability to maximize, close, and hide the program.

XFCE provides a graphical configuration tool you can access by clicking on the wrench icon in the panel. This program lets you configure anything from the desktop background to keybindings, screensaver settings, and the taskbar. Click the User Interface icon to open the theme manager, where you can configure the look and feel of XFCE.

To log out of XFCE, click the power icon on the panel, or right-click on the desktop and choose Quit. For more information about XFCE, visit the official page at http://www.xfce.org.

Use Fluxbox

Fluxbox (http://www.fluxbox.org) is another alternative window manager, which is popular for its speed and ability to group windows into tabs. While it is relatively lightweight, Fluxbox offers a number of features, including the ability to remember window placement, configurable panels, and add-ons that let you have icons on your desktop (*fbdesk*) and a pager (*fbpager*).

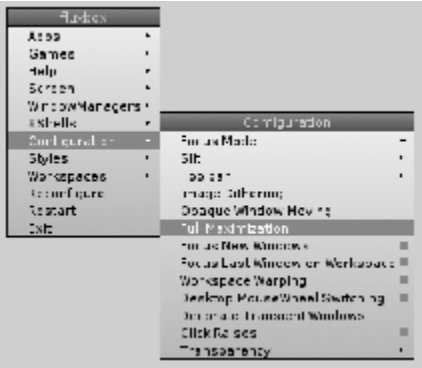
To install Fluxbox, select the *fluxbox* package in your preferred package management tool. After it is installed, log out of your current desktop environment, select the Fluxbox session, and log back in to see the default Fluxbox desktop, shown in Figure 2-9.

Figure 2-9. Default Ubuntu Fluxbox desktop

By default, the Fluxbox desktop is pretty bare; there is only a small panel along the bottom of the screen. This panel contains a taskbar to display all applications open on the current desktop and has the ability to switch to new desktops. Right-click on the panel to change panel-specific options.

Instead of an application menu in the panel, Fluxbox displays the menu whenever you right-click on the desktop. In addition to the standard application categories, such as Apps and Games, the main menu has a Configuration submenu where you can configure Fluxbox settings—including how it focuses windows and toolbar settings, as shown in Figure 2-10. There is also a Styles menu that lets you change the theme.

Figure 2-10. Fluxbox Configuration submenu



As I mentioned previously, Fluxbox allows you to group windows together with tabs. Middle-click on the titlebar of a window and then drag and drop it onto another window. Fluxbox will automatically tab the windows together so that you can click on one of the tabs on the titlebar to select a window. To remove a tab, middle-click on it in the titlebar and drag and drop it onto the desktop.

To exit Fluxbox, right-click on the desktop and select Exit. For more information on Fluxbox, visit the main project page at http://www.fluxbox.org.

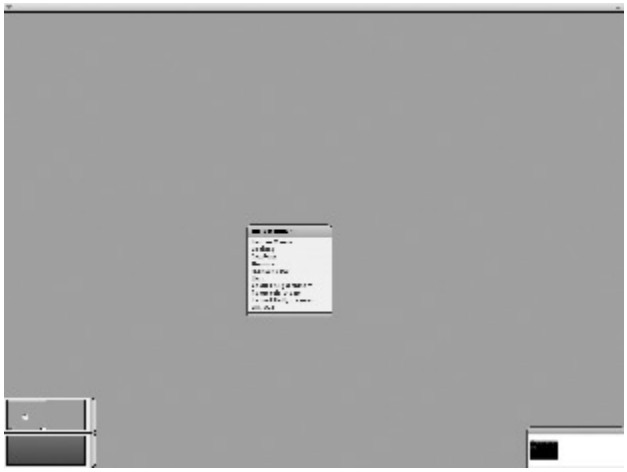
Seek Enlightenment

Enlightenment (http://www.enlightenment.org) has long been known as the window manager with all the eye candy. Back in the days of 100 MHz processors, it was also known as being slow. These days, Enlightenment offers the same eye candy with almost infinite configurability and, with modern computers, a really snappy response.

To install Enlightenment, install the *enlightenment* package with your preferred package manager. In addition, you may want to install the *e16keyedit* and *e16menuedit* tools, since they provide the GUI to edit your keybindings and main menu, respectively. After the *enlightenment* package is installed, log out of your current desktop, choose the Enlightenment session, and then log back in.

The default Enlightenment desktop is almost as bare as Fluxbox (see Figure 2-11). Along the top is a desktop dragbar, which you can use to switch desktops. If you are on any desktop other than the main root desktop, you can also drag the desktop dragbar down to reveal the desktop underneath. At the bottom left of the desktop is the Enlightenment pager. The pager displays a view of each desktop, along with any windows open on that desktop. You can drag and drop windows from one desktop to another by dragging them within the pager, or even by dragging them from the pager and dropping them on the current desktop. Along the bottom right of the desktop is the iconbox. The *iconbox* stores icons for any windows you *iconify* (minimize), instead of having them appear in a taskbar. Right-click on the iconbox to configure its settings, including its size and whether the background is transparent. To access the main application menu, middle-click on the desktop. Right-click on the desktop to configure Enlightenment settings, including the number of desktops, background settings, and so on. Right-click and select Special FX Settings to see the different types of eye candy you can configure.

Figure 2-11. Default Ubuntu Enlightenment desktop



Enlightenment also offers *advanced window memory*. Window memory allows you to remember settings about a particular window, such as its location, its size, which desktop it's on, and other settings. The next time you start the program, Enlightenment will remember and use any or all of the settings you told it to remember. This way, you can, for instance, always have your web browser open on a particular desktop. To configure which settings to remember, right-click on the titlebar for a window and select Remember.

Window grouping is another feature Enlightenment offers that many other window managers don't. To group windows, right-click the titlebar on the first window and select Window Groups→"Start a new group." Then right-click on the titlebar for any other windows you want to group with it and select Window Groups→"Add this Window to the Current Group." When windows are grouped, they can all be configured to mimic operations performed on any other window in the group. For instance, if you minimize one window in the group, the rest will minimize. If you move one window, the others will move along with it. To configure which behaviors all members of a group will adhere to, right-click on the titlebar for one of the windows and select Window Groups→"Configure this Window's Groups."

To exit Enlightenment, middle-click on the desktop and select Log Out. For more information about Enlightenment, middle-click on the desktop and select Help, or you can visit the main project page at http://www.enlightenment.org.

Other Window Managers

There are plenty of other window managers you can install under Ubuntu, such as Blackbox, Openbox, WindowMaker, Afterstep, and FVWM. To install any of these window managers, search for its name in your preferred package manager and then install the corresponding package. Most of the major window managers will add themselves to your sessions menu so you can easily select them when you log in.

Hack 18. Install Java

The modern web-browsing experience requires Java. Here's how to install Java and caffeinate your web browser.

Ubuntu's an amazing Linux distribution for lots of different reasons, but one of the things people find attractive about it is the fact that it ships with lots of different software, preconfigured and ready to use. However, one of the things that the Ubuntu developers can't include is the Sun Java Runtime Environment (JRE), because Sun charges to license it.

However, the Debian Java Maintenance team makes a nifty package called *java-package* that lets you take the Java installation bits from Sun and semi-automatically install them using the standard Debian tools. Since Ubuntu is based on Debian, it inherits this ability. The benefit of this procedure is that the *java-package* binaries also set up the Mozilla Java Plug-in for Firefox, and it takes only a few steps. You must have the universe and multiverse repositories enabled [Hack #60] in order to follow this procedure.

Getting the Java Runtime Environment

To get started, first go to http://java.sun.com/j2se/1.5.0/download.jsp and download the latest Java (JRE 5.0 Update 6, as of this writing). Ensure you get the "self-extracting file," not the RPM version. Once you've downloaded the JRE to your hard disk, make the file executable:

```
bill@defiant:~$ chmod +x jre-1_5_0_06-linux-i586.bin
```

Creating the Java package from the JRE

Next, install *java-package*, *java-common*, and *fakeroot*. Fakeroot is a tool that will allow a user without *root* permissions to create installable *.deb* archives:

```
bill@defiant:~$ sudo apt-get install fakeroot java-package java-common
```

Once you've installed those packages, you can move on to actually creating the Java installation package (you may be prompted to accept Sun's license for the JRE along the way):

```
bill@defiant:~$ fakeroot make-jpkg jre-1_5_0_06-linux-i586.bin
```

When this process is complete, you'll have a *.deb* package in that directory. The *.deb* will be called *sun-j2re* `version` *_i386.deb*.

Installing the new Java package

Once that's done, the JRE can be installed using the standard Debian/Ubuntu `dpkg` command:

```
bill@defiant:~$ sudo dpkg -i sun-j2re1.5_1.5.0+update06_i386.deb
```

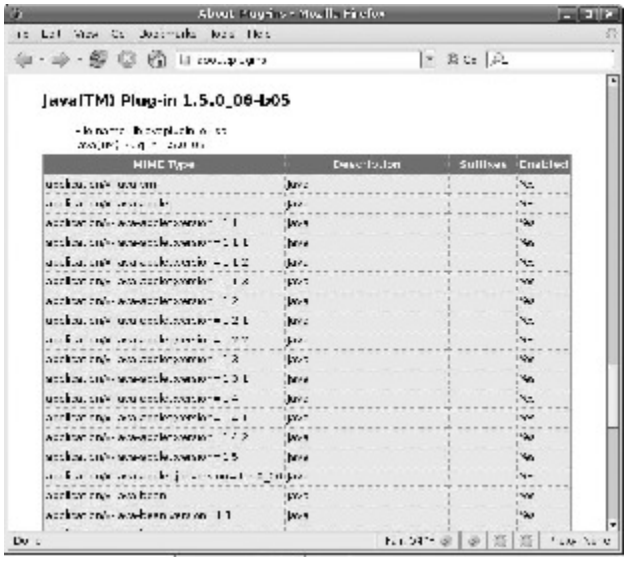
At this point, the JRE should be installed and configured. You can verify that it's installed by running the following command from a terminal window. You should get similar output.

```
bill@defiant:~$ java -version
java version "1.5.0_06"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_06-b05)
Java HotSpot(TM) Client VM (build 1.5.0_06-b05, mixed mode, sharing)
```

If you see an unexpected version of Java—in particular, one identified as "gij (GNU libgcj)"—then you probably have GNU Java installed. You can easily switch from one Java to another with the command `sudo update-alternatives --config java`, which will prompt you to choose which Java implementation to use. Pick the one in */usr/lib/j2re version* to use the JRE you just installed.

Firefox should also be Java-enabled at this point; to verify it, restart Firefox if you had it running during the Java installation and type `about:plugins` into the address bar. The About Plug-ins page will load, listing the Java plug-in, as shown in Figure 2-12.

Figure 2-12. The about:plugins screen in Firefox, showing the JRE plugin installed



You can now surf to Java-enabled web sites and use Java applications!

Hack 19. Search Your Computer

Have Beagle find and fetch your information faster than you thought possible.

One of Linux's weaker points as a desktop OS has been the lack of a search feature. Nautilus has had a "find this file" function for quite some time, but it's really not much more sophisticated than a GUI wrapper around the command-line *find* command. Both Nautilus and the *find*

command do similar things—they'll look at each and every file in the directory structure, trying to find a match for the criteria you've given them.

There is a better way to search a filesystem. It involves creating an index of all the files on that filesystem, which enables you to search the index much like you would a database. This is what Windows and Mac OS X do for their file-search capabilities, and now Linux has it too in the form of Beagle, a modular search engine that's written in Mono. It's easy to add Beagle to Ubuntu, and the usability benefits are tremendous.

Installing Beagle

In this hack, you'll be installing Beagle and a very cool search frontend known as *deskbar-applet*. *deskbar-applet* sits in your GNOME panel and enables all manner of search goodness for you. As with many optional goodies, you'll need to have the universe repository enabled [Hack #60] to install both of these packages. Now, open up a terminal [Hack #13] and install *beagle* and *deskbar-applet*:

```
bill@lexington:~$ sudo aptitude install beagle deskbar-applet
```

Starting Beagled

Once you've got *beagle* and *deskbar-applet* installed, you'll need to start *beagled* (the main engine and database) manually. From a terminal, you'll simply run *beagled*. It should start and detach from your terminal, and run in the background:

```
bill@lexington:~$ beagled
```

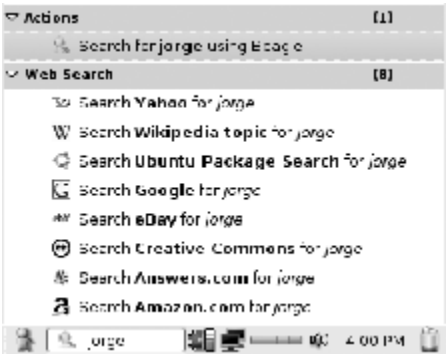
It will then begin the process of indexing your hard disk(s). This will take a while, depending on the amount and type of data you have. I have seen *beagled* take up to three hours to fully index a disk. If you're running *beagled* on a laptop, you may want to make sure it's plugged into AC power, because the high I/O from *beagled*'s initial indexing could drain your battery. While *beagled* is indexing, you can set up your GNOME desktop to automatically start *beagled* when you log in. Simply click on the System Menu, and select Preferences and then Sessions. Add *beagled* to your Startup Programs, and it will be ready to fetch stuff for you on your next login.

Using Beagle and deskbar-applet

Now that the Beagle daemon is running, it's time to add *deskbar-applet* to the mix. The deskbar applet is a GNOME applet, so add it to one of the GNOME panels by right-clicking on the panel and selecting "Add to Panel." Select Deskbar, click Add, and then close the window. You'll see the deskbar in your panel now.

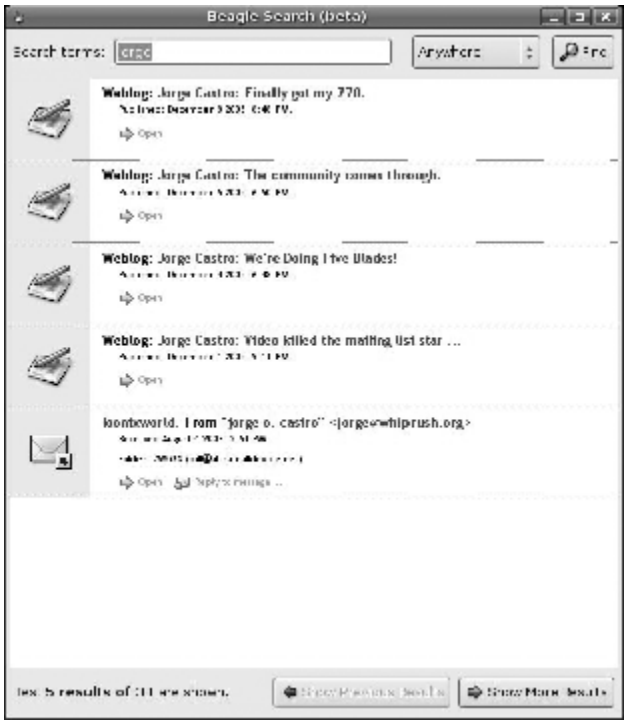
At this point, you can put criteria into the deskbar applet and go ahead and search for something (see Figure 2-13). If you put your search criteria in the applet and click the little magnifying glass, you can tell the deskbar applet to search any one of a number of databases, including Beagle, which is not enabled by default. (The default search can be changed in the preferences for *deskbar-applet*; simply right-click on the applet and select Preferences to adjust it).

Figure 2-13. Searching Beagle with deskbar



Once you click on "Search for ... using Beagle," Beagle takes over, digging through its index and fetching the proper results. By default, Beagle will search the files in your home directory, as well as metadata like email archives, instant messages, and blog posts. It doesn't matter where the data is; Beagle will fetch it for you (see Figure 2-14).

Figure 2-14. Beagle fetching my data for me



As you start to use Beagle, you'll gradually see the benefit of having an agent index all your data and metadata. It may be a clich\x8e , but over time, this will change the way you work.

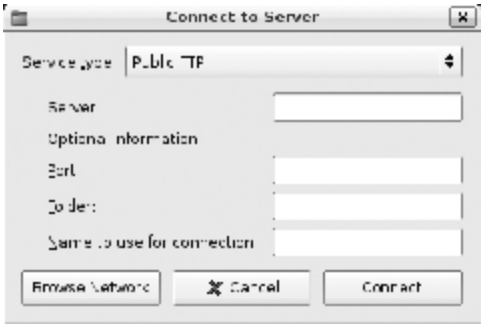
Hack 20. Access Remote Filesystems

Use integrated desktop tools to access a number of different remote network shares.

Let's face it: sometimes it's difficult to fit all of the files you need on a single computer. Whether they be Windows shares at your office, FTP servers somewhere on the Internet, or even machines on the network running SSH, you can access all of these servers and more from the Ubuntu desktop with a few clicks.

The key to connecting to remote file systems is the "Connect to Server" dialog window. Click Places→Connect to Server to see the default window, shown in Figure 2-15.

Figure 2-15. A sample "Connect to Server" window for FTP connections

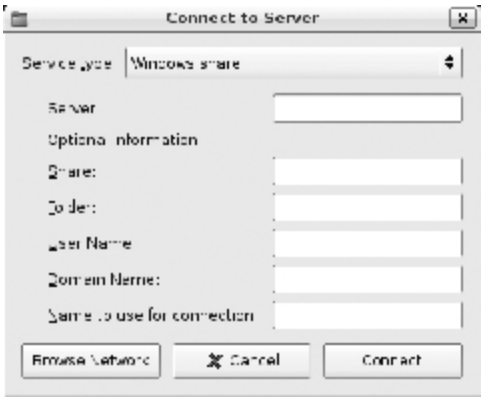


There are a number of different connection types the dialog supports. Figure 2-16 shows the options available from the drop-down menu. Apart from a few specific options, each of these connection types shares the same sorts of options. The top of the window requests the location of the server (a hostname or IP address), and then below that is a list of nonessential options you can configure. For instance, you can give each of your connections custom names so that they are easier to tell apart from each other.

Figure 2-16. The submenu shows the different types of remote servers you can connect to

Connecting to a Windows share is a good example of how to use the "Connect to Server" dialog. First, select "Windows share" from the drop-down menu. Then fill in the name of the server you want to connect to and, optionally, the name of the share you want to connect to (Figure 2-17). If your network requires authentication, you can also configure the username and domain name in this window. Once you have configured the share, click the Connect button. A new icon for this share will then appear on your desktop. Double-click that icon to open the Nautilus file browser to that share.

Figure 2-17. Sample "Windows share" dialog without any fields filled in

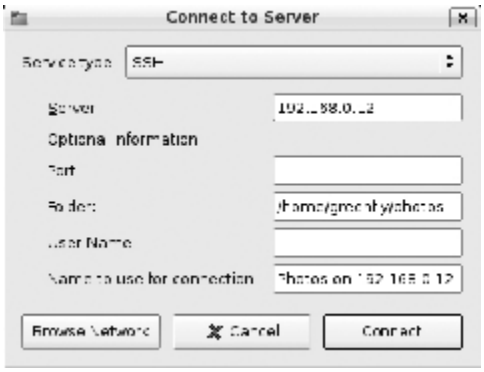


If you aren't quite sure about the settings for your Windows share, you can also click Browse Network to search the local network for any available Windows shares.

One of the more interesting abilities of the "Connect to Server" dialog is to connect to remote SSH servers and share files over SFTP. Not only does this mean that you don't have to configure any special file sharing on the remote machine, but any machine that runs SSH is now also a file share you can access. What's more, all of the communication is sent over an encrypted channel.

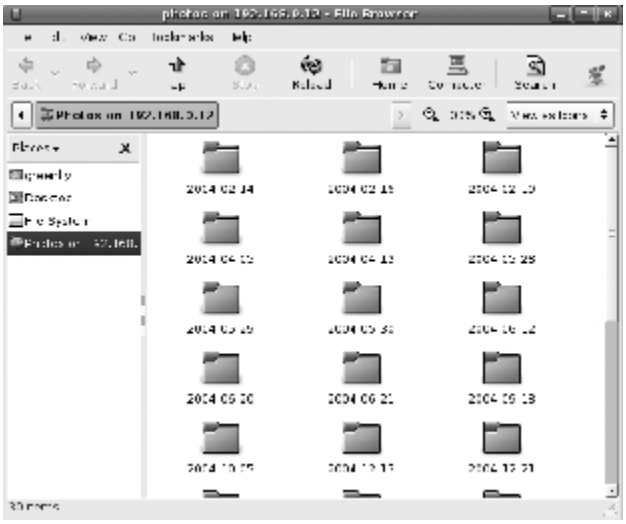
To connect to a remote SSH server, select SSH from the “Service type” drop-down menu. In Figure 2-18, I have an sample window filled out with information to connect to a server on my network. By default, the SSH connection will open into the / directory, but you can change that to any directory you wish in the Folder field. If I needed to connect to the remote server as a different user, I could also specify that in this window. Finally, I’ve decided to give this share a custom name that will appear both on its icon in the desktop and in the sidebar of the file browser.

Figure 2-18. You can even connect to remote servers strictly over SSH



Once I click Connect, a new icon appears on my desktop, and if I double-click it, I can access all of the photos on my remote server, as shown in Figure 2-19. I can then drag and drop them to and from my local machine as I do with any other directory.

Figure 2-19. Here are some of my files accessed over SSH



If you decide that you no longer want to access a particular share, just right-click on its icon on the desktop and select Unmount Volume. Otherwise, file shares will appear both in the sidebar of your file manager and in the Places→Network Servers window.

Hack 21. Tweak Your Desktop Like a Pro

Customize your desktop environment, and find a few things you never knew you needed, like a pop-up Quake-style terminal, automatic wallpaper shuffling, and dashboard functionality.

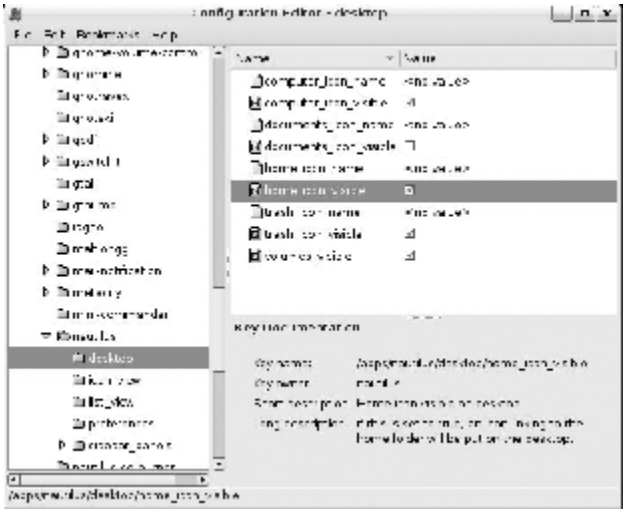
The default Ubuntu GNOME environment is very streamlined and easy to use. Due to its ease of use, however, some of the "power user" features aren't included in the base install. KDE can also benefit from the same tweaking. Here's how to get more usability and features from Ubuntu.

Get Icons on Your Desktop

The default Ubuntu Dapper Drake desktop has no icons on it—which gives the system a very clean and simple look. If you're coming from Windows, you may miss the My Computer icon and other desktop icons in that OS. It's relatively simple to add them to Ubuntu; it just requires the use of a configuration editor called gconf-editor [Hack #15].

If you hit Alt-F2, the system will pop up a Run Application dialog. Type `gconf-editor` in that dialog and click Run. The Gconf configuration program will start. Select "apps" from the left pane; then select "nautilus" and "desktop." In the right pane, you will see several options, like "computer_icon_name" and "computer_icon_visible." If you click on the checkbox next to the "*name*_icon_visible" option (see Figure 2-20), Nautilus will spontaneously add that icon to your desktop. In this fashion, you can add icons for your computer, home directory, network places, documents, and wastebasket.

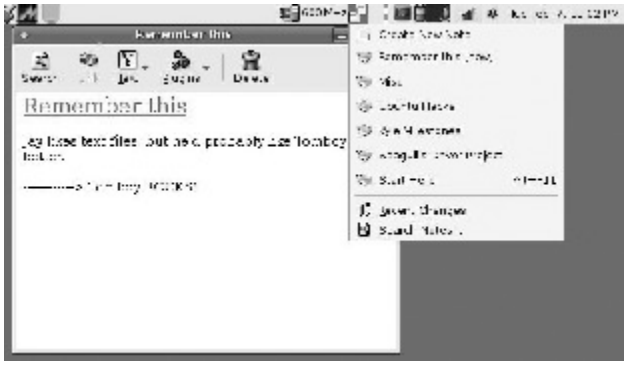
Figure 2-20. GConf showing the Nautilus icon-configuration options



Get Organized with Tomboy

Tomboy, shown in Figure 2-21, is a tiny application that sits in your panel and acts as an always-on mini Wiki. It's great for making quick notes to yourself without worrying about saving myriad text files or waiting for an editor to start. Much like a Wiki, you can cross-reference the Tomboy notes you make using Wiki-like annotations. Not only that, but the notes are quickly searchable using Tomboy's built-in search capabilities.

Figure 2-21. Tomboy notes



To install Tomboy, just *apt-get* it from a terminal:

```
bill@defiant:~$ sudo apt-get install tomboy
```

That will fetch and install Tomboy and its dependencies. Once that's completed, you can add Tomboy to your panel by right-clicking on the panel, selecting "Add to Panel," and then choosing Tomboy Notes from the dialog.

Automatic Wallpaper Switching

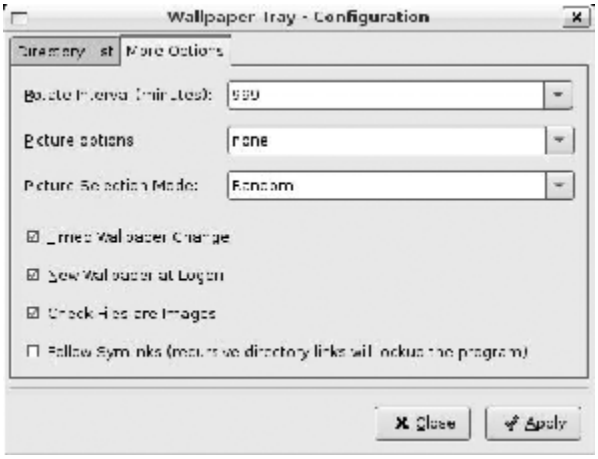
Something GNOME doesn't provide for is a way to automatically rotate your desktop wallpaper. Luckily, there's a little application written by someone known only as "Earthworm" called *wp_tray* that will sit in your notification area and rotate your wallpaper based on whatever scheme you wish. The source code for this application is at http://planetearthworm.com/projects/wp_tray/files/wp_tray-0.4.6.tar.gz, but I've built it for Ubuntu and made it available at http://wildbill.nulldevice.net/ubuntu/wp-tray_0.4.6-1_i386.deb. Download the *.deb* for *wp_tray* and install it using the following command:

```
bill@defiant:~$ sudo dpkg -i wp-tray_0.4.6-1_i386.deb
```

Once *wp_tray* is installed, add it to your startup programs so it starts when you log in: click on the System Menu, then Preferences, then Sessions, and add *wp_tray* to the list of Startup Programs. Log out and log in again, and you will be able to right-click on the applet (see Figure 2-22) and configure it to your wishes (Figure 2-23).

Figure 2-22. The interface to wp_tray

Figure 2-23. wp_tray's configuration screen



Getting a Pull-Down "Quake" Terminal

There's a very useful KDE application called *yakuake* that takes the standard KDE Konsole and changes it to a drop-down, on-demand terminal over your desktop and applications. (It's called a "Quake" terminal because of its resemblence to the console that drops down in the Quake series of games.) This is a very handy little application. To install, simply use *apt-get* to install both *konsole* and *yakuake*:

```
bill@lexington:~$ sudo apt-get install konsole yakuake
```

Add *yakuake* to your startup session using the same instructions for the *wp_tray* applet. Log out and log in again, and then you can hit F12 to cause *yakuake* to drop down over your desktop as shown in Figure 2-24.

Figure 2-24. yakuake: the drop-down konsole

Hack 22. Sync Your Palm PDA

Your Palm OS handheld can join in on the Ubuntu fun. Learn how to install applications on your Palm or Treo, keep it in sync with Evolution, and back it up.

Getting a Palm OS PDA to synchronize with Linux has usually involved some amount of effort, and some pain installing and configuring the necessary software. The folks working on Ubuntu have made it easy, however. Ubuntu includes all the software necessary to synchronize your Palm with Evolution and do almost everything you've done under Windows.

Configuring Palm Synchronization

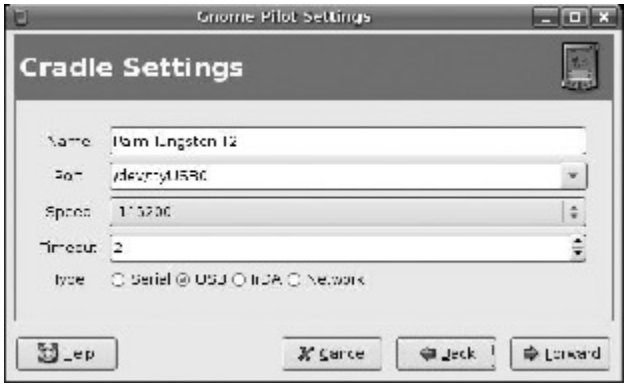
Since the Ubuntu and Evolution developers included the *gnome-pilot* package in Ubuntu, there's no software that needs to be installed. Everything you need is on your system; it just needs to be configured to sync with your Palm.

To begin the configuration process, start Evolution, click on the Edit menu, and select Synchronization Options. The *gnome-pilot* splash screen will appear (see Figure 2-25). Click on Forward to proceed.

Figure 2-25. The gnome-pilot startup dialog

Next, *gnome-pilot* displays the Cradle Settings dialog, shown in Figure 2-26. Put values corresponding to your Palm and your system in this dialog box. For instance, USB-equipped Palms will probably sync using port */dev/ttyUSB0* and a speed of 115200, and they will require the USB radio button to be selected. Older, serial Palms will probably need the port set to */dev/ttyS0* and a speed of 57600, and they will need the Serial radio button selected. Click on Forward to continue.

Figure 2-26. The Cradle Settings dialog



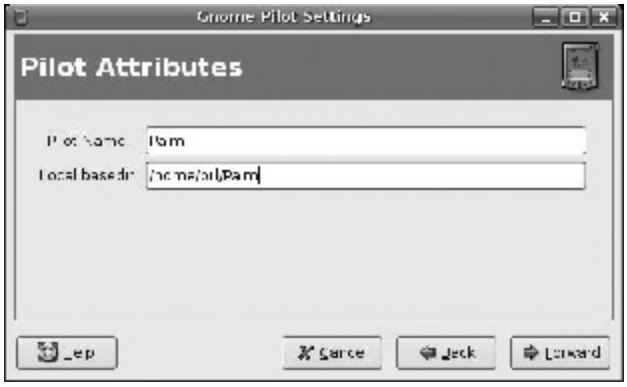
Now it's time to identify the Palm (see Figure 2-27). If you've synchronized your Palm with another PC or operating system before, select "Yes, I've used sync software with this pilot before." If you have never synchronized your Palm, select "No, I've never used sync software with this pilot before." If you select No, ensure that your User Name is set to something you'd like the Palm to have embedded in it. The ID string doesn't require any changes or editing. Click on Forward to move to the next step (initial sync).

At this point, you'll be prompted to press the Hotsync button on your Palm. Press it, and you should hear a couple of quick beeps from the Palm. Click Forward to move on to the next step.

Figure 2-27. Identifying the Palm

The next screen (see Figure 2-28) asks you to enter a descriptive name for your Palm and a directory path. It defaults to MyPilot; you can leave it at the default or change it as I have.

Figure 2-28. Setting the Pilot attributes



You're now done configuring the Palm sync method. Click on the Apply button (see Figure 2-29) to commit the configuration.

Figure 2-29. Palm sync configured

Your sync may not initiate automatically. You may have to mount the *usbfs* by entering `sudo mount -t usbfs none /proc/bus/usb/` at a terminal window. This is to work around a change the Dapper developers made.

Configuring the Sync Conduits

The only thing that remains is to configure the individual synchronization conduits. In Evolution, go back to the Edit menu and select Synchronization Options. The Pilot Settings dialog is where you can enable and disable the Evolution conduits, and set up the synchronization methods and rules. To edit conduits, click on the Conduits tab, select a conduit, and click Enable or Disable.

You can set the default sync action as well as a one-time action that will fire just on the next sync. You can choose to add private records to your sync and decide which Evolution database you'd like to sync to. Figure 2-30 shows an example of the Address Book conduit.

Figure 2-30. Address Book sync options

Add the Pilot Applet

To make sure the GNOME Pilot daemon (*gpilotd*) is started when you log in, you should add the Pilot Applet to your panel. Right-click on the panel and choose Add To Panel. Locate Pilot Applet under Utilities, and add this to the panel. You can click the applet to bring up the conduit settings and right-click it to get a menu with several options, shown in Figure 2-31.

Figure 2-31. Pilot Applet option menu

At this point, your Palm is ready to sync with Evolution. Simply hook the Palm up and initiate a hotsync, and it'll sync with Linux just like it did under Windows.

Hack 23. Sync Your Pocket PC

Just because your Pocket PC is powered by Windows doesn't mean you can't sync it with Ubuntu.

Inside the box of your Pocket PC, you'll find a USB cable that's compatible with nearly every computer on the planet. What you won't find is software that's compatible with anything other than Windows. To sync your Pocket PC with Ubuntu, you'll need some additional software. Fortunately, most of this software is available in Ubuntu's *universe* repository.

ActiveSync, the software that comes with your Pocket PC, takes care of synchronizing calendars and contacts, and also installing applications in the Pocket PC. With the *ipaq* USB-serial module, the SynCE suite of tools, and Multisync, you can do all of this on your Linux system.

Connecting the Pocket PC

The first order of business is to figure out which interface your Pocket PC uses. If you're using a recent Pocket PC, it will probably look like an iPAQ as far as Linux is concerned. Before you plug your Pocket PC in, run `sudo modprobe ipaq` to load the iPAQ USB-to-serial driver. Plug your Pocket PC in and examine the output of *dmesg* to see whether it was detected and which device represents it. In the following example, a Pocket PC was detected on */dev/ttyUSB0*:

```
bjepson@ubuntu:~$ sudo modprobe ipaq
Password:
bjepson@ubuntu:~$ dmesg | tail
[ 1720.274390] usbcore: registered new driver usbserial
[ 1720.285461] drivers/usb/serial/usb-serial.c: USB Serial support registered for generic
[ 1720.317318] usbcore: registered new driver usbserial_generic
[ 1720.319073] drivers/usb/serial/usb-serial.c: USB Serial Driver core
[ 1720.415421] drivers/usb/serial/usb-serial.c: USB Serial support registered for PocketPC PDA
[ 1720.448394] drivers/usb/serial/ipaq.c: USB PocketPC PDA driver v0.5
[ 1720.457095] usbcore: registered new driver ipaq
[ 1738.233238] usb 1-1: new full speed USB device using ohci_hcd and address 2
```

```
[ 1738.796279] ipaq 1-1:1.0: PocketPC PDA converter detected
[ 1738.803167] usb 1-1: PocketPC PDA converter now attached to ttyUSB0
```

Unfortunately, this probably won't work with a Pocket PC that's too new. At the time of this writing, SynCE did not yet support the most recent Pocket PC operating system, Windows Mobile 5.0.

The next thing you'll need to do is install some packages, but before you do so, make sure you've enabled the *universe* repository in */etc/apt/sources.list*. Then, run `sudo apt-get update` to get the latest packages and install *synce*, *dccm*, and some supporting tools:

```
bjepson@ubuntu:~$ sudo apt-get install synce-dccm synce-serial librra0-tools
```

When *dpkg* hits the *synce-serial* package, it will ask you for a few things: the serial interface used by your Pocket PC, the IP addresses for the PPP connection used with your Pocket PC, and the IP address of your DNS server. You need to provide the DNS server address, but you can accept the defaults for the rest (although, you should check the serial interface against what you found earlier in the output of *dmesg*). Although you specified the serial port during the setup, I've found that you need to do it once more after installation, so run this command:

```
bjepson@ubuntu:~$ sudo synce-serial-config /dev/ttyUSB0

You can now run synce-serial-start to start a serial connection.
```

Testing the Connection

Now it's time to give it a try. Make sure your Pocket PC is connected to your computer, and start the *dccm* daemon. Without arguments, it will run in the background. Since this is its first time out, I suggest starting it in the foreground (`-f`) with debugging enabled (`-d 4`):

```
bjepson@ubuntu:~$ dccm -d 4 -f
dccm[4599]: Running in foreground
dccm[4599]: Listening for connections on port 5679
```

Open another terminal and start up SynCE:

```
bjepson@ubuntoo2:~$ sudo synce-serial-start

synce-serial-start is now waiting for your device to connect
```

If you see the error "synce-serial-abort was unable find a running SynCE connection. Will do nothing," then the Pocket PC has probably gone to sleep while you were setting things up. If it did, it will disconnect from the machine, and */dev/ttyUSB0* will disappear. If you see "PocketPC PDA converter now disconnected from ttyUSB0" in the output of *dmesg*, unplug it and plug it back in, and try *synce-serial-start* again.

Once you have the connection up and running, the Pocket PC should stay awake.

Watch the shell where you started *dccm* and wait a while; it can take anywhere from 30 seconds to a minute for the Pocket PC and *dccm* to start chatting. Once they begin talking to one another, you'll see something like this:

```
dccm[4941]: Connection from 192.168.131.201 accepted
info package (104 bytes):
  0000: 28 00 00 00 04 14 00 00  (.......
  0008: 11 0a 00 00 00 00 00 00  ........
  0010: 2a 1e c3 37 00 00 00 00  *..7....
  0018: 28 00 00 00 3c 00 00 00  (.......
  0020: 5a 00 00 00 00 00 00 00  Z.......
  0028: 50 00 6f 00 63 00 6b 00  P.o.c.k.
  0030: 65 00 74 00 5f 00 50 00  e.t._.P.
  0038: 43 00 00 00 50 00 6f 00  C...P.o.
  0040: 63 00 6b 00 65 00 74 00  c.k.e.t.
  0048: 50 00 43 00 00 00 53 00  P.C...S.
  0050: 53 00 44 00 4b 00 00 00  S.D.K...
  0058: 00 00 50 00 57 00 31 00  ..P.W.1.
  0060: 30 00 42 00 31 00 00 00  0.B.1...
  0068:
dccm[4941]: Talking to 'Pocket_PC', a PocketPC device of type PW10B1
```

To disconnect the Pocket PC, first run the command `killall -HUP dccm`. If all went well, you should see "Connection interrupted" and "Connection from 192.168.131.201 closed" in the *dccm* session. You can leave *dccm* running. To reconnect, run `sudo synce-serial-start` again.

Establish a Partnership

Before you can sync to the Pocket PC, you'll need to set up a partnership. Pocket PCs can be partnered with at most two computers, so if you've already configured your Pocket PC with a Windows computer running ActiveSync, you'll have one slot left. The *synce-matchmaker* program will pick an empty slot and use it:

```
bjepson@ubuntoo2:~$ synce-matchmaker create
[rra_matchmaker_create_partnership:356] Partnership slot 1 is empty on device
[rra_matchmaker_create_partnership:356] Partnership slot 2 is empty on device
Partnership creation succeeded. Using partnership index 1.
```

If you don't have a free slot, you'll need to pick one to remove. See the *synce-matchmaker* manpage for details.

You need to run *synce-matchmaker* only once to pair it with your computer.

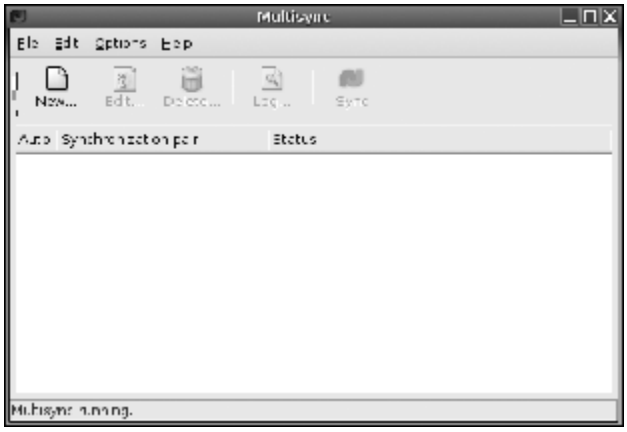
Synchronize with Evolution

To synchronize your Pocket PC with Evolution, you'll need to install Multisync (http://multisync.sourceforge.net), a modular synchronization package that lets you create pairs of repositories that are kept in sync. To install Multisync and the SynCE plug-in, issue the command `sudo apt-get install libmultisync-plugin-all synce-multisync-plugin`, which will also install a collection of plug-ins, including the Evolution plug-in and the backup plug-in. You'll use the backup plugin to verify that Multisync is running correctly before you try to synchronize it with SynCE.

At the time of this writing, most of the pieces required to sync a Pocket PC with KDE were present in Ubuntu, but they did not work well together. In particular, the Raki panel applet would segfault when it tried to set up a partnership, and the *syncekonnector*, required for syncing PIM information with KDE, did not build against the versions of various libraries included with Dapper. However, for those inclined to compile a handful of libraries from scratch, there is a HOWTO for Debian users at http://sourceforge.net/mailarchive/forum.php?thread_id=9562091&forum_id=15200.

Don't look for Multisync in Evolution: you won't find it under Edit→Synchronization Options or Edit→Plugins. You'll need to launch the *multisync* application under X11 to work with it (you can either launch it from a shell or load it from Applications→Accessories→Multisync). When you launch Multisync, its main window appears, as shown in Figure 2-32.

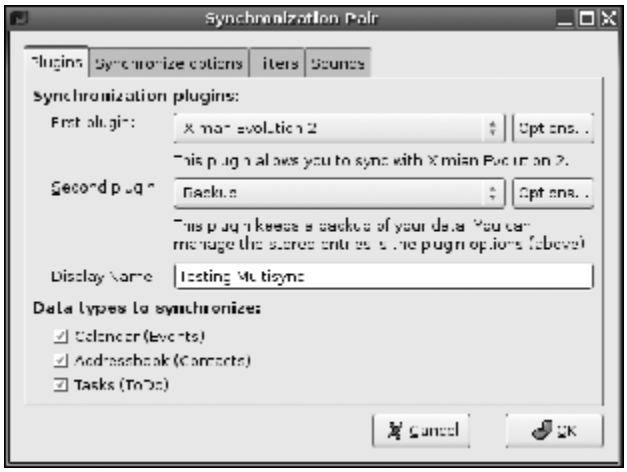
Figure 2-32. Launching Multisync for the first time



Testing Multisync

To make sure that Multisync is working properly, you should configure the backup plug-in and do a test sync with it. Click the New button, and the Synchronization Pair dialog will appear, as shown in Figure 2-33.

Figure 2-33. Testing Multisync with the backup plug-in



Specify Ximian Evolution 2 as the first plug-in and Backup as the second (see Figure 2-33). Give this pair a name, and then click and set the Options for each plug-in. For Evolution, you need to specify which calendar, address book, and task categories to sync. For Backup, you need to specify a backup directory (this appears on the Options tab of the Backup plug-in options dialog). You can then dismiss the Synchronization Pair dialog by clicking OK, which returns you to the main Multisync window. Click once to select the pair you just created, and then click the Sync button. The sync should go quickly, depending on how many contacts, tasks, and appointments you have. Once it's finished, open a shell or Nautilus window and navigate to whichever backup directory you specified in the Backup options. You should see some new files there:

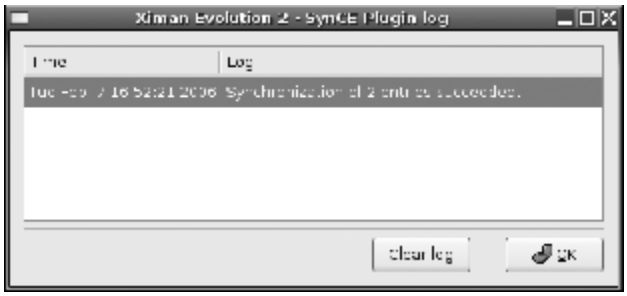
```
bjepson@ubuntoo2:~$ cd backup/
bjepson@ubuntoo2:~/backup$ ls -l
total 8
-rw-r--r-- 1 bjepson bjepson  26 2006-02-07 16:14 backup_entries
-rw-r--r-- 1 bjepson bjepson 422 2006-02-07 16:14 multisync1139346850-0
```

Syncing to the Pocket PC

Return to the Multisync main window and click Add again. This time, set up a Synchronization Pair between Evolution and the SynCE plug-in. Just as you did with the earlier pair, click Options (next to Evolution) to configure which categories to synchronize (the SynCE plug-in has no options). Click OK to dismiss the dialog and return to the Multisync main window. Click to select the Evolution/SynCE pair and click the Sync button.

If all went well, your Pocket PC and Evolution will be in sync. Click the Log button to see what happened. In Figure 2-34, you can see that two entries were copied successfully from the Pocket PC to Evolution.

Figure 2-34. The Multisync log after synchronizing with a Pocket PC



Each time you want to sync, you'll need to start *dccm* (or keep it running in the background), plug in the Pocket PC, and run `sudo synce-serial-start`. Then you can launch Multisync and synchronize. To disconnect, run `killall -HUP dccm` and unplug the Pocket PC.

Install Pocket PC Software

Pocket PC software usually comes in one of two forms: a self-installing *.exe* file designed to be run under Windows, or a *.cab* (cabinet) file that contains ready-to-run binaries for your Pocket PC. If you're stuck with an *.exe* file, no need to worry. The *orange* package is available in the universe repository [Hack #60] and will extract the *.cab* files for you:

```
$ orange PocketVCSSetup.exe
squeezing out: /home/bjepson/pocketvcs.arm.cab
squeezing out: /home/bjepson/pocketvcs.mips.cab
squeezing out: /home/bjepson/pocketvcs.sh3.cab
-------
3 files
```

Once you have the *.cab* file (either downloaded from a web site or extracted using *orange*), you can install it onto your Pocket PC. Most modern Pocket PCs are of the ARM variety, but you can use *synce-pstatus* to find out for sure:

```
$ synce-pstatus | grep ^Processor
Processor architecture: 5 (ARM)
Processor type:         2577 (StrongARM)
```

Then you can use *synce-install-cab* to install the *.cab* file for the appropriate architecture:

```
$ synce-install-cab pocketvcs.arm.cab
Copying file 'pocketvcs.arm.cab' to device...
File copy of 1660980 bytes took 0 minutes and 6 seconds, that's 276830 bytes/s.
Installing 'pocketvcs.arm.cab'...
```

After the file is copied over, you will see an installer run on your Pocket PC. You may be prompted for additional information. You can use the *synce-list-programs* and *synce-remove-program* utilities to list and remove programs you've installed on the Pocket PC.

Accessing the Pocket PC's Filesystem

You can work with files on the Pocket PC using the utilities listed in Table 2-1.

Table Utilities for working with the Pocket PC filesystem

Utility	Description
synce-pcp	Copy a file
synce-pls	List files
synce-pmkdir	Create a directory
synce-pmv	Move/rename a file
synce-prmdir	Remove a directory

For example, the PocketVCS application isn't much good without some games. As you can see, it doesn't come with any:

```
$ synce-pls "/My Documents/PocketVCS/"
AC--------      444646  Tue 11 Mar 2003 04:19:22 PM EST  PocketVCS.pro
```

To actually manipulate a file (copy, remove, etc.), you need to prefix it on the remote file system with a colon (`:`). Here's how to copy the free Oystron (http://www.io.com/~nickb/atari/oystron.html) over to the Pocket PC:

```
$ synce-pcp OYSTR29.BIN ":/My Documents/PocketVCS/OYSTR29.BIN"
File copy took less than one second!
```

As you can see, the file is over there and ready to play:

```
$ synce-pls "/My Documents/PocketVCS/"
AC--------        4096  Thu 01 Jul 2004 04:56:26 AM EDT  OYSTR29.BIN
AC--------      444646  Tue 11 Mar 2003 04:19:22 PM EST  PocketVCS.pro
```

Troubleshooting

If you have trouble with your Pocket PC, make sure that the *ipaq* module was loaded correctly and examine the output of *dmesg* to ensure that the Pocket PC was detected and the serial link was established.

If your Pocket PC and Evolution don't seem to be talking, go back to the terminal that's running *dccm* and make sure that it has recognized your Pocket PC and established a connection to it. If they aren't talking, try stopping the connection and reconnecting. If `killall -HUP dccm` doesn't reset the connection, try `sudo synce-serial-abort`. Pocket PCs are notorious for getting confused about things, so if you find that it's impossible to connect after that point, you might want to reset your Pocket PC. Heck, you might even need to reboot your computer if things get really confused.

If you still have problems, visit the SynCE web site (http://synce.sourceforge.net) for documentation and other troubleshooting information.

Brian Jepson

Hack 24. Customize the Right-Click Contextual Menu

Write your own scripts to perform custom actions when you right-click on a file, folder, or the desktop, and add templates to make document creation quick and painless.

Right-clicking on items on the desktop or in the Nautilus file browser pulls up a contextual menu that allows you to perform operations directly on the item. But you're not limited to just the default options: you can add template documents and scripts to the menu for easy access with a single click.

Easy-Access Templates

Right-clicking on the desktop or in the background of a Nautilus window brings up a Create Document menu item, which normally includes only an "Empty File" item. If you select Empty File, a new file called *new file* will be created, and you can rename it to anything you like. However, the new file is just a totally empty file. Creating a document this way is essentially the same as running:

```
$ touch "new file"
```

and about as useful.

However, it's easy to add your own templates to the menu. Create a directory called *Templates* in your home directory:

```
$ mkdir ~/Templates
```

Any document you place in that directory will now be available through the Create Document contextual menu. If the menu becomes large, you can group items into submenus by placing them in subdirectories within *Templates*.

HTML developers can put a file called *HTMLFile.html* in this directory (it will appear as "HTML File" on the Create Document menu) and fill it with the skeleton of an HTML file. If you create a lot of corporate documents using OpenOffice.org templates, copy the templates in, and you'll be able to start a new document in any location by just right-clicking and selecting the template.

If you do not see your new templates appear in the menu right away, just log out of GNOME and log back in again.

Custom Scripts

You can also execute custom scripts directly from the contextual menu by placing your scripts into a special directory located inside your home directory: *.gnome2/nautilus-scripts/*. Any script you place in that location can be accessed by right-clicking on a file or window and selecting it from the Scripts submenu. (The Scripts menu will not be available unless you have some scripts installed.)

When a script is executed via the contextual menu, it is passed a number of environment variables and usually a number of arguments so it can optionally act on the files you have selected. If you execute the script from the context of a local folder on your computer, it will be passed

the names of all selected files as arguments. If you execute the script from the context of a remote folder [Hack #20], such as a Nautilus window showing web or FTP content, no arguments will be passed to it.

Four environment variables are also set, which you can access from within the script:

`NAUTILUS_SCRIPT_SELECTED_FILE_PATHS`
: Newline-delimited paths for selected files if they are local

`NAUTILUS_SCRIPT_SELECTED_URIS`
: Newline-delimited URIs for selected files

`NAUTILUS_SCRIPT_CURRENT_URI`
: URI for current location

`NAUTILUS_SCRIPT_WINDOW_GEOMETRY`
: Position and size of the current window

There are even packages for various prewritten script collections, such as the Nautilus Subversion Management Scripts and Nautilus Audio Convert packages, which allow you to perform Subversion actions and convert audio file formats by right-clicking on files:

```
$ sudo apt-get install nautilus-script-collection-svn \\
              nautilus-script-audio-convert
```

Not seeing the Scripts menu when you right-click? That may be because Nautilus doesn't think you have any scripts. To give it a heads-up, select Go→Location, type into the location bar, and press Enter. Next time you right-click on a file or directory, you should see the Scripts menu.

As a simple example, you could place the following into *~/.gnome2/nautilus-scripts/Terminal* to give you easy access to a terminal from the contextual menu:

```
#!/bin/sh
gnome-terminal
```

This will open a terminal with the current directory set to the directory that encloses whatever you right-clicked on. So, if you right-clicked on the icon for *~/foo*, you'd get a terminal whose current working directory was ~. But suppose you right-click on a directory? The following Terminal script will check each item in `NAUTILUS_SCRIPT_SELECTED_FILE_PATHS`, and if it finds a directory, it will `cd` to it and open the terminal there. Otherwise, it will just open the terminal in the directory that contains the item you clicked on:

```
#!/bin/sh
for d in $NAUTILUS_SCRIPT_SELECTED_FILE_PATHS; do
 if [ -d $d ]; then
    cd $d
    gnome-terminal
    exit
  fi
done
gnome-terminal
```

More complex uses could be to encrypt a selected file using GPG, set a selected image as the desktop background, or send a selected file as an email attachment. For a collection of a variety of scripts specifically designed for use in the contextual menu, visit http://g-scripts.sourceforge.net.

Hack 25. Download and Share Files with the Best P2P Software

File sharing is here to stay, and Ubuntu provides some powerful tools that enable users to join the revolution.

Peer-to-peer is huge and getting bigger all the time. Linux users don't have to be left out of all the excitement. In fact, we have a huge variety of P2P apps and networks from which to choose. In this hack, I'm going to show you how to install several different P2P apps; using them, however, will be up to you. And be sensible about what you share, OK? I don't want any large organizations whose job is protecting dying cartels suing you.

BitTorrent

When it comes to P2P, first and foremost on any serious Linux user's machine is BitTorrent, the fabulous technology developed by Bram Cohen that makes downloading ISOs and other huge collections of files easy, fast, and stable.

Don't know how BitTorrent works or what makes it special? See Wikipedia's article at http://en.wikipedia.org/wiki/Bittorrent. Also at Wikipedia, there is an excellent comparison of the various BitTorrent clients (http://en.wikipedia.org/wiki/Comparison_of_BitTorrent_clients).

There are oodles of BitTorrent apps, and they can all be classified as either command-line or GUI-based. You should know about both, since they each have their purposes. If I'm looking for something quick and dirty, I use the command line; if I want a lot more info and control, I use a GUI.

Command Line

You probably already have BitTorrent installed on your copy of Ubuntu, which the following command will confirm:

```
$ whereis bittorrent
bittorrent: /usr/share/bittorrent
```

If you don't have it on your computer, run the following:

```
$ sudo apt-get install bittorrent
```

Accept any dependencies if they're requested, and now you're ready to roll.

To test the software, try downloading a Linux ISO image. Open Firefox and head over to http://linuxtracker.org, which tracks many different Linux distros, all available via BitTorrent. Find a distro that intrigues you, and click on the little floppy-disk icon. When Firefox asks you where to save the *.torrent* file, pick a location on your hard drive and save it (Firefox may just go ahead and drop it in the *Desktop* directory). In my case, I'll use *~/iso*, a directory in my home for ISO images. Close Firefox, open your terminal, `cd` to the directory containing the *.torrent*, and then use the text-mode BitTorrent client (*btdownloadcurses*) to download the file:

```
$ cd ~/iso
$ ls
KANOTIX-2006-CeBIT-RC3.iso.torrent
$ btdownloadcurses --responsefile KANOTIX-2006-CeBIT-RC3.iso.torrent
```

Once BitTorrent starts up, you'll see information about the progress of your download (and upload, since BitTorrent also shares whatever you download with other users; if you want to limit upload speed, use the `--max_upload_rate` *SPEED* option on the command line, replacing *SPEED* with the speed in kilobytes per second):

```
| file:      KANOTIX-2006-CeBIT-RC3.iso
| size:      694,765,568 (662.58 MiB)
| dest:      /home/scott/iso/KANOTIX-2006-CeBIT-RC3.iso
| progress: ##_____________________________________________ _
| status:    finishing in 1:09:25 (3.5%)
| dl speed: 193.6 KB/s
| ul speed: 0.5 KB/s
| sharing:  0.007  (0.2 MB up / 28.5 MB down)
| seeds:    13 seen now, plus 0.882 distributed copies
| peers:    1 seen now, 85.7% done at 56.2 kB/s
```

If you instead have a URL instead of a torrent file, use a slightly different command:

```
$ btdownloadcurses --url "http://linuxtracker.org/download. □
            php?id=1624&name=KANOTIX-2006-CeBIT-RC3.iso.torrent"
```

The same progress information will appear, allowing you to follow along with the progress of BitTorrent. Just leave your terminal open until your download finishes. In order to give something back to your fellow BitTorrent users, it's a good idea to leave everything running until your upload is equal to your download. When you're ready to stop BitTorrent, just press Q to quit.

For more information about using BitTorrent on the command line, use `man bittorrent`. There are actually a lot of different options you can use to customize BitTorrent exactly to your needs.

GUI

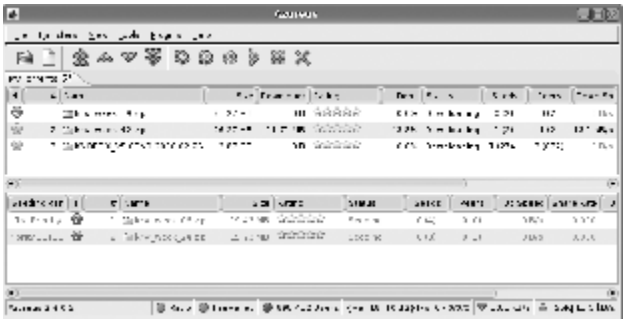
There are lots of good BitTorrent GUIs out there, but Azureus is probably the best—with the most features, constant updates, and a dedicated development team. Before you can use the program, however, you need to install Java [Hack #18], since Azureus requires it. (Sun's licensing, however, makes it impossible for Ubuntu to include the official Java Runtime Environment in its repositories.)

After you've installed Java, download Azureus from http://azureus.sourceforge.net/. Extract the file somewhere, such as */opt*:

```
$ cd /opt/
$ sudo tar xvfj /home/bjepson/Desktop/Azureus_2.4.0.2_linux.tar.bz2
```

You can now run Azureus by executing `/opt/azureus/azureus`. As you can see in Figure 2-35, it's a program that gives you an enormous amount of data about the files you're downloading and sharing.

Figure 2-35. Azureus is a full-featured, powerful P2P app



For more on this great app, check out an article I wrote for *Linux Magazine* titled "Azureus: A Better Way to BitTorrent" (http://www.linux-mag.com/content/view/1923/43/). It'll give you some tips when it comes to using Azureus that you should find useful.

aMule and eDonkey2000

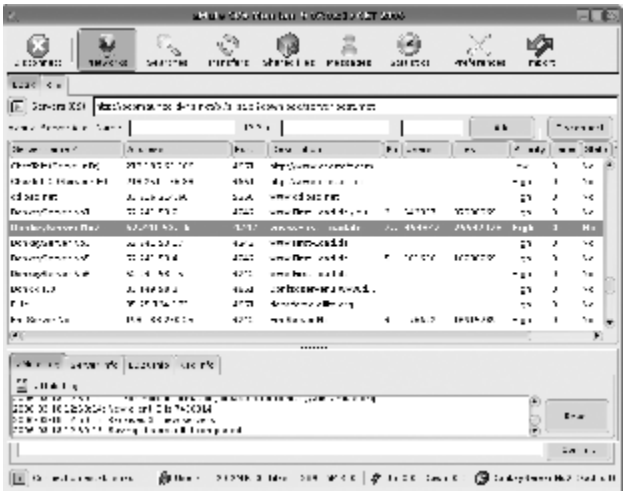
aMule is a P2P client for the eDonkey2000 (or ed2k) network, which, according to Wikipedia, is the most widely used P2P network in the world (see http://en.wikipedia.org/wiki/EDonkey_network for that statistic, as well as some other important information you should read if you plan to use the ed2k network). You can install aMule using *apt-get*, since it is part of Ubuntu's *universe* repository.

Make sure that the universe repository [Hack #60] is enabled, and then enter the following:

```
$ sudo apt-get install amule
```

You'll be asked if you want to install some other packages that aMule needs, so go ahead and say yes. When *apt* finishes, you can start aMule by going to the K menu→Internet→aMule (if you use KDE) or Applications→Internet→aMule (if you use GNOME). The program will start up, as you can see in Figure 2-36.

Figure 2-36. Searching for Linux goodies using aMule



aMule is running, but there's a lot more to do. To configure and use aMule, start with the "aMule wiki" (http://www.amule.org/wiki/index.php/Main_Page), especially the Getting Started page (http://www.amule.org/wiki/index.php/Getting_Started). Another great source of help is the aMule forums (http://www.amule.org/amule/index.php). Try out aMule: you really can find just about anything there.

LimeWire and Gnutella

One of the first solutions to the centralized-server problem posed by Napster was the Gnutella network (for more info, see http://en.wikipedia.org/wiki/Napster and http://en.wikipedia.org/wiki/Gnutella). Today, there are many Gnutella clients available for Linux users, but one of the best is undoubtedly LimeWire. There are two versions of LimeWire: the pay version (which is $18.00 for six months of updates) and the free version, which will constantly bother you to upgrade to the pay version but is otherwise adware- and spyware-free.

To get the free version, point your web browser to http://www.limewire.com/LimeWireSoftLinux, and you should be prompted to accept a download. It's an RPM file, which normally won't work on a Debian-based distro like Ubuntu, but don't worry. You're going to use a wonderful program called *alien* to convert the RPM into something you can use on your distro. First you'll need to install *alien*:

```
$ sudo apt-get install alien
```

You may have to approve a few additional packages that satisfy dependencies, so go ahead and do so. Once *alien* is in place, you can use it to transform the RPM into a DEB suitable for installing on your Ubuntu box:

```
$ sudo alien LimeWireLinux.rpm
limewire-free_4.10.9-1_i386.deb generated
```

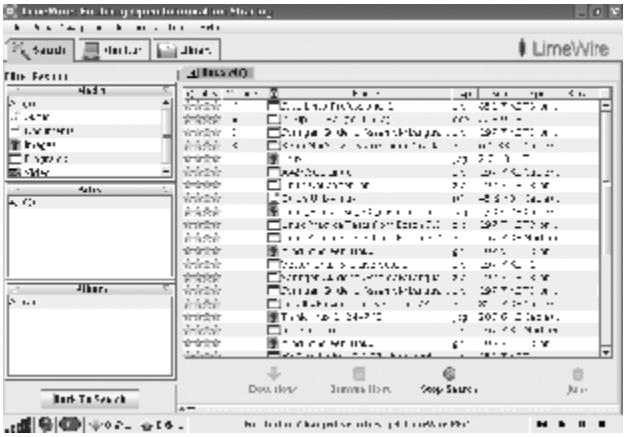
Now you can install the LimeWire DEB:

```
$ sudo dpkg -i limewire-free_4.10.9-1_i386.deb
```

As a bonus, you now have a DEB that you can use on other Ubuntu machines under your control, or you can pass it along to friends.

To use LimeWire, go to the K menu→Internet→LimeWire (for KDE) or Applications→Internet→LimeWire (for GNOME). LimeWire will open and ask you a few questions to get started, and then you can begin using it to search for goodies, as you can see in Figure 2-37. (Regrettably, some of the search results appear to be unauthorized copies of books and other material.)

Figure 2-37. LimeWire in action



LimeWire is a powerful program that connects to a large and popular network, so you should be able to have a lot of fun with it.

Other P2P Apps

I've shown you programs for many of the popular P2P networks, but there are many others out there, and it seems like new networks and programs pop up all the time. A great way to keep up-to-date with this growing technology is through the Wikipedia article "Comparison of file sharing applications" (http://en.wikipedia.org/wiki/Comparison_of_P2P_applications). If you see one that looks interesting, do a Google search for that app or network, followed by the word "Ubuntu"; so, for instance, if a new P2P app appeared named "arglebargle," you'd query Google for *arglebargle ubuntu* to see if there was any Ubuntu-specific info available. Just remember: be careful with what you share!

Scott Granneman

Hack 26. Make your own PDFs

Are you used to using the "Print to PDF" feature in Adobe Acrobat? Here's how you can make your own PDF files using a similar technique.

A very handy feature that's included in the Mac OS X operating system is the ability to "print" a PDF file from any application. Windows can also do this, via Adobe Acrobat. However, with the addition of a single package and a little bit of tweaking, you can get the same capability for free on Ubuntu Linux.

Installing CUPS-PDF

The key to getting PDF printing enabled is in the package *cups-pdf*, which is in the universe repository [Hack #60]. Use *apt-get* from a terminal window to install the *cups-pdf* package:

```
bill@lexington:~$ sudo apt-get install cups-pdf
Password:
Reading package lists... Done
Building dependency tree... Done
The following NEW packages will be installed
  cups-pdf
0 upgraded, 1 newly installed, 0 to remove and 34 not upgraded.
Need to get 23.4kB of archives.
After unpacking 147kB of additional disk space will be used.
Get: 1 http://us.archive.ubuntu.com dapper/universe cups-pdf 2.0.3-1 [23.4kB]
Fetched 23.4kB in 0s (31.2kB/s)
Selecting previously deselected package cups-pdf.
(Reading database ... 105713 files and directories currently installed.)
Unpacking cups-pdf (from .../cups-pdf_2.0.3-1_i386.deb) ...
Setting up cups-pdf (2.0.3-1) ...
 * Stopping Common Unix Printing System: cupsd                 [ ok ]
 * Starting Common Unix Printing System: cupsd    ...done.
```

After the installation of *cups-pdf* is complete, the CUPS configuration file requires a small edit to enable PDF printing. From a terminal window, run:

```
bill@lexington:~$ sudo gedit /etc/cups/cupsd.conf
```

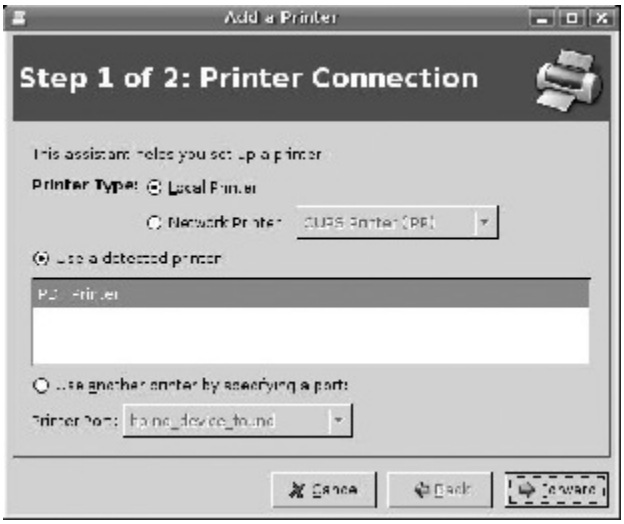
Find the line that says `RunAsUser Yes` and change it to `RunAsUser No`, and then save the file and exit *gedit*. Next, you'll need to restart CUPS to make the configuration change effective:

```
bill@lexington:~$ sudo /etc/init.d/cupsys restart
 * Stopping Common Unix Printing System: cupsd     [ ok ]
 * Starting Common Unix Printing System: cupsd    ...done.
```

Configuring CUPS for the PDF printer

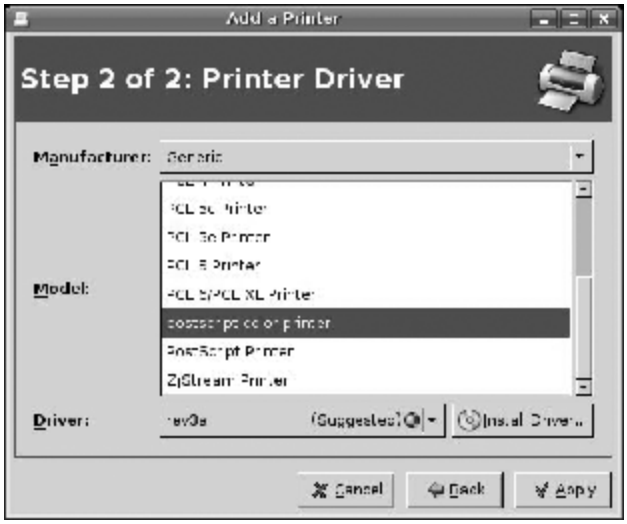
Now you must tell CUPS to use the newly installed *cups-pdf* package. Click on the System menu; select Administration and then Printing. Double-click on New Printer to start the "Add a Printer" wizard. Ensure that Local Printer is selected under the printer type and that "Use a Detected Printer" is selected, with PDF Printer highlighted in the list below (see Figure 2-38). Click on Forward to proceed.

Figure 2-38. Adding the PDF printer



On the next screen, select Generic under the Manufacturer field (see Figure 2-39). Pull down the Model pull-down and select "postscript color printer." Click on Apply to commit the changes.

Figure 2-39. Specifying the postscript driver



Using your new PDF printer

Using the new PDF printer is simple: from any application, use that application's native printing facility and select "postscript-color-printer" as your printer. The printer driver will automatically create a PDF and put it in a PDF subdirectory within your home directory. The filename will probably be something like *job_1-untitled_document.pdf*. That's it, it's just that easy! Who needs Acrobat?

Hack 27. Blog with Ubuntu

Blogging is all the rage, and you can update your blog from Ubuntu using the Drivel blog client.

Blogging has become a very popular activity as of late; there's even been an O'Reilly book about it (*Essential Blogging* by Cory Doctorow et al.). Lots of people have LiveJournals or maintain their own blog servers. Most blogs have their own web-based administrative interfaces, but if you're not online, you can't update your blog. That's where a blog client comes in: it allows you to write blog posts offline and upload them when you're ready. There is a very good blog client called Drivel in the *universe* repository, which is very easy to configure and use.

Installing Drivel

Thanks to *apt-get*, Drivel is extremely easy to install (you need to have the universe repository enabled [Hack #60]). Simply run the following command from a terminal window:

```
bill@lexington:~$ sudo apt-get install drivel
Reading package lists... Done
Building dependency tree... Done
The following NEW packages will be installed
  drivel
0 upgraded, 1 newly installed, 0 to remove and 34 not upgraded.
Need to get 353kB of archives.
After unpacking 1487kB of additional disk space will be used.
Get: 1 http://us.archive.ubuntu.com dapper/universe drivel 2.0.2-5 [353kB]
Fetched 353kB in 1s (191kB/s)
Selecting previously deselected package drivel.
(Reading database ... 105636 files and directories currently installed.)
Unpacking drivel (from .../drivel_2.0.2-5_i386.deb) ...
Setting up drivel (2.0.2-5) ...
```

There will be a new menu entry for Drivel created in the Applications menu in the Internet section. Simply click on that entry to start Drivel.

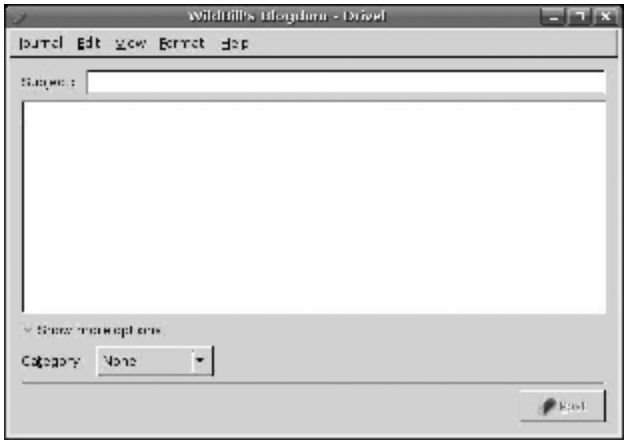
Configuring and Using Drivel

Upon starting Drivel for the first time, you'll be presented with the main Drivel dialog box, which is where you'll need to configure the program. Enter your username, password, and server address into the appropriate fields in the dialog, and select your journal type from the drop-down menu. (If you're using a Movable Type or Movable Type–compatible blog such as Wordpress, make sure to put the path to your *mt-xmlrpc.cgi* file in the Server Address field.) Click on Log In (see Figure 2-40), and Drivel will hook up to your blog and pull down an index of your latest blog posts.

Figure 2-40. Drivel's splash and login screen

Now Drivel's ready for you to make your first post (see Figure 2-41)! Fill in a subject line for your post, and type some content in the big body field. When you click Post, Drivel will upload the post to your blog, where it'll be made available to the whole world. The "Show more options" arrow will bring down a pull-down dialog where you can select which category you want your post to be categorized in, if your blog supports categories.

Figure 2-41. Drivel's main window



Drivel also supports different fonts and font effects via the Format menu. It will also automatically construct image links for you, although you'll have to upload the image (in the correct size) to your blog outside of Drivel. Drivel also makes it easy to call up old blog entries and edit them if necessary; simply click on the Journal menu and select "Recent Entries" to see the last few entries you've written.

In all, Drivel's one of those small applications that does one thing, and does it well. Give it a test-drive and see if it can't help you manage your blog.

Chapter 3. Multimedia

A modern computer can be a nifty little entertainment center; it's got fast video, stereo speakers, probably plenty of disk space for movies and videos. So, how come Ubuntu doesn't know what to do when you insert a DVD or double-click on a video or audio file?

The fundamental problem is that there are some critical pieces to the multimedia puzzle that Ubuntu can't distribute in all jurisdictions, so you need to go out and get those pieces yourself. Fortunately, Ubuntu makes it easy for you to install those bits once you do obtain them. In this chapter, you'll learn how to get Ubuntu to support more multimedia formats, play DVDs, and even let you buy music online.

For care and feeding of your multimedia collection, you'll need to eventually work with optical discs. Whether you want to take a CD you purchased and rip it down into MP3 files, create a personalized remix CD, or burn a movie to DVD, the hacks in this chapter have you covered.

Hack 28. Install Multimedia Plug-ins

Music and video files come in a wide variety of exotic formats. Learn how to locate and install the plug-ins you need to view or listen to them.

On some Linux distributions, getting all of your multimedia files to play seems like it requires all sorts of command-line voodoo. One thing that sets Ubuntu apart is just how easy it is to grab all of the packages you need to play multimedia content. In this hack, we will walk you through the necessary steps so that once you are finished, Ubuntu will happily play just about any media file you throw at it.

Tweak Your Repository List

Many of the multimedia packages you need to install do not reside in the default Ubuntu repository. To get these packages, you will need to add the *universe* and *multiverse* repositories to Ubuntu. If you haven't done so yet, check out "Modify the List of Package Repositories" [Hack #60] for information about these repositories and how to add them. Once you have added the repositories, be sure to update your list of packages within your package-management tool before proceeding.

Install General-Purpose Libraries and Tools

There are a number of basic libraries and multimedia tools you need to install to get the best support for your multimedia files. These include libraries for MP3 and Ogg Vorbis playback, as well as media players and other tools. If you use Synaptic, select *totem-xine* from the GNOME Desktop Environment (*universe*) category, *vorbis-tools* from the Multimedia category, *sox* from the Multimedia (*universe*) category, *faad* and *lame* from the Multimedia (*multiverse*) category, *imagemagick* from the Graphics category, and finally *ffmpeg* and *mjpegtools* from the Graphics (*universe*) category. Or, if you use *apt-get*, type

```
$ sudo apt-get install totem-xine vorbis-tools sox faad lame \\
              imagemagick ffmpeg mjpegtools
```

Install Gstreamer Libraries

Gstreamer is a new plug-in–based approach to multimedia playback. Gnome uses Gstreamer for much of its own multimedia playback, and while it is still under heavy development, you will still want to install a number of Gstreamer libraries for GNOME. If you use Synaptic as your package management tool, select the *gstreamer0.8-plugins-multiverse* package from the Libraries (*multiverse*) category, *gstreamer0.8-ffmpeg*, *gstreamer0.8-mad*, and *gstreamer0.8-plugins* from the Libraries (*universe*) category, and *gstreamer0.8-lame* from the Multimedia (*multiverse*) category, and apply your changes. Alternatively, if you use *apt-get*, type:

```
$ sudo apt-get install gstreamer0.8-plugins-multiverse \\
              gstreamer0.8-ffmpeg gstreamer0.8-mad gstreamer0.8-plugins \\
              gstreamer0.8-lame
```

If version 0.8 of these Gstreamer libraries is not available, search your package repository to see if a new version is available (for example, if you are using *apt*, run the command `apt-cache search gstreamer`).

Once you have installed all the Gstreamer libraries, open a terminal and type:

```
$ gst-register-0.8
```

to register all of the Gstreamer plug-ins on your system.

Install Codecs of Ambiguous Legality

There are a number of multimedia formats that are encumbered by special licenses that require the user to leverage Windows codec libraries on their Linux system to play back the file. Some of these include QuickTime and Windows Media formats. In certain countries, it may be illegal to play files via these codecs, so open up your checkbook, call up your lawyer, and have a chat before proceeding. Then, open a terminal window and grab a copy of the *w32codecs .deb* file from ftp://ftp.nerim.net/debian-marillat/pool/main/w/w32codecs/ and install the *.deb*:

```
$ wget ftp://ftp.nerim.net/debian-marillat/pool/main/w/w32codecs/w32codecs_ □

            20050412-0.0_i386.deb
$ sudo dpkg -i w32codecs_20050412-0.0_i386.deb
```

If for some reason that site isn't available, you can also track down the *w32codecs* packages from the official MPlayer page at http://mplayerhq.hu.

For information on setting up your computer to play encrypted DVDs, read "Play DVDs" [Hack #30].

Hack 29. Watch Videos

While the default Totem video player in Ubuntu is great, it's hard to beat MPlayer in terms of flexibility, configurability, and features.

Every once in a while, a tool comes along in Linux that impresses you in almost every respect with its flexibility. MPlayer is one of those tools. When it comes to video and audio playback, think of MPlayer as your universal translator. It can play basically any audio or video format you throw at it (provided it has the libraries available), in just about any container you throw at it. For instance, it can play DVDs from the disc, an image of the disc, or even just the VOBs from the disc. Of course, depending on your taste, there is one downside: by default, MPlayer is a command-line program. There is a graphical frontend for MPlayer for those interested, called *gmplayer*, or you can just use the default Ubuntu video player, Totem. This hack discusses the basics of how to play multimedia files with MPlayer from the command line.

Install MPlayer

The first step to using MPlayer is to install it and the codecs it needs. If you haven't already followed the steps in "Install Multimedia Plug-ins" [Hack #28], do that first to get all of the codecs you'll need. Next, use your preferred packaging tool and install the *mplayer* package that matches your CPU architecture. If you use *apt-get*, type:

```
$ sudo apt-get install mplayer-
                       686
```

Replace *686* with `386`, `586`, `k6`, `k7`, `g4`, `g5`, etc., depending on your processor. To see a list of the different processor options, type:

```
$ apt-cache search mplayer
```

Use MPlayer

With MPlayer installed, basic file playback is as simple as opening a terminal and typing:

```
$ mplayer
                    file.avi
```

The console will immediately fill with a lot of different output. This can be useful because MPlayer is telling you what information it can figure out about the file you passed to it, along with information about how it will try to play it. MPlayer should also display the video in a new window and immediately start playback. Back in the console, you will see output scroll by as MPlayer updates you on which frame is playing and how far along MPlayer is in the video.

MPlayer provides an extensive list of key bindings so that you can control playback. The manpage lists all of these options; Table 3-1 lists some of the more commonly used ones.

Table Mplayer keybindings

Keys	Function
Left and right arrows	Seek backward/forward 10 seconds
Up and down arrows	Seek backward/forward 1 minute
Page Up and Page Down	Seek backward/forward 10 minutes
< and >	Move backward/forward in playlist
p, Space	Pause movie (pressing again unpauses)
q, Esc	Stop playing and quit
+ and –	Adjust audio delay by +/- 0.1 seconds
/,9 and *,0	Decrease/increase volume
m	Mute sound
f	Toggle full-screen
t	Toggle stay-on-top

Most of these key bindings are pretty self-explanatory, but the + and - options to adjust the audio delay are worth further discussion. Sometimes when you create your own videos or convert videos between formats, the audio and video fall out of sync. This can be very frustrating when you are watching a movie, but with MPlayer, you can tweak the audio with the + and - keys. Just hit one of the keys a few times and see whether you have improved or worsened the sync problems, and then adjust until the video and audio is completely in sync.

The full-screen key binding (f) won't necessarily scale the video to fill up the entire screen. Whether the video scales depends on the video output option you select for MPlayer.

MPlayer is truly a universal multimedia playback tool, and in the next sections, I'll list some examples for playing back specific video types. For most video files, it is sufficient to simply pass the filename as an argument to *mplayer*, but for special videos such as DVDs, VCDs, and filestreams, things are done slightly differently.

DVD playback

MPlayer has good support for DVD playback; however, one thing it does not support is DVD menus. When you play a DVD with MPlayer, it skips the menu system and everything else up-front and goes right to the movie, which can actually be a feature if you don't want to sit

through the numerous ads and FBI warnings some DVDs have. Most DVDs have a main feature—the movie you purchased—plus several lesser features, such as behind-the-scenes footage or scenes that were cut. In the case of episodic discs like TV box sets, each episode is a different feature. Each of these features is a title, and you can select which title to play when you run *mplayer*. To start playback of the first title on a DVD, type:

```
$ mplayer dvd://
                1
```

To play other titles, replace `1` with the number of the title you want to play. If you want to play a range of titles, you can specify the range on the command line. For instance, to play titles three through six, type:

```
$ mplayer dvd://3-6
```

You can also specify individual chapters (scenes) or a range of chapters with the `-chapter` argument. To play chapters four through eight on title one, type:

```
$ mplayer dvd://1 -chapter 4-8
```

MPlayer will attempt to play from */dev/dvd*, but if that device doesn't exist, or you want to point it to a different device, use the `-dvd-device` argument. The following command will play back from */dev/hdc*:

```
$ mplayer dvd://1 -dvd-device /dev/hdc
```

You can even use the `-dvd-device` argument to play back directly from a DVD image somewhere on your filesystem:

```
$ mplayer dvd://1 -dvd-device
                /path/to/dvd.iso
```

It is even possible to use a directory full of VOB files:

```
$ mplayer dvd://1 -dvd-device
                /path/to/directory/
```

You may also specify language and subtitle options directly from the command line. The `-alang` option controls the audio language option and can accept multiple languages separated by commas. In that case, MPlayer will try the first language and fall back on the next language if the first isn't available. For instance, to play a movie in Japanese and fall back to English if Japanese isn't available, type:

```
$ mplayer dvd://1 -alang ja,en
```

The `-slang` option controls which language's subtitles are shown. To show the English subtitles on the above example, type:

```
$ mplayer dvd://1 -alang ja,en -slang en
```

(S)VCD playback

(S)VCD playback in MPlayer is much like DVD playback. Just use `vcd://` instead of `dvd://` in the command line, with the track to play as an argument. So, to play track one of a VCD, type:

```
$ mplayer vcd://1
```

MPlayer can even play the *.bin* files from (S)VCDs. You don't even need to pass any special options; just point *mplayer* to the *.bin* file to start playback.

Streaming playback

MPlayer supports playback from a number of different audio and video streams. Just pass the URL on the command line:

```
$ mplayer
                http://example.com/stream.avi
$ mplayer
                rtsp://example.com/stream
```

Troubleshooting

There are a number of reasons MPlayer may not output your video correctly. If MPlayer has trouble identifying your video, all the video codecs *mplayer* requires may not be installed on your system. "Install Multimedia Plug-ins" [Hack #28] explains how to find and install the various video and audio codecs you need under Linux.

If MPlayer plays the video, but the video output looks strange, you can't see it at all, or playback is very jerky, it's possible that MPlayer is configured to use the wrong video output option for your system. Try passing `-vo x11` as an argument to *mplayer* on the command line and see if that lets you at least view the video.

Another reason for jerky video is simply that a system is too slow to play the video well. In this case, MPlayer will warn you in its output that your system is too slow to play the video and will recommend that you add the `-framedrop` option. This option tells MPlayer to drop video frames if the video can't keep up with the audio on the system.

Hack 30. Play DVDs

Install libraries that allow you to play encrypted DVDs under Ubuntu.

Out of the box, Ubuntu will probably not be able to play most of the DVDs that you own. This isn't because of an oversight on the part of the Ubuntu developers; it's simply because most DVDs you might buy are encrypted with a system called CSS (Content Scrambling System). While video-player packages such as *totem-gstreamer*, *totem-xine*, *xine*, *mplayer*, and *vlc* can all play unencrypted DVDs, to play CSS-encrypted DVDs, you will have to actually circumvent the encryption scheme. (Note that in certain countries circumventing CSS is not legal, so here is a good place to stop reading and phone up your attorney before proceeding.)

Ubuntu actually makes this process very simple. The first step is to install one of the aforementioned video-player packages along with the *libdvdread3* package, if they aren't installed, so use your preferred package-installation tool to do so (see Chapter 6 for different ways to install packages).

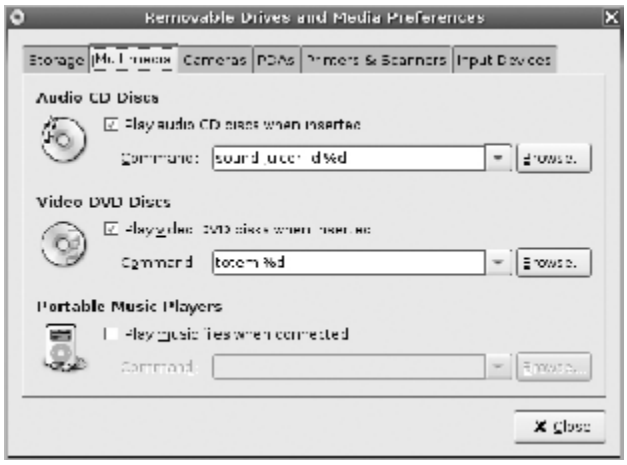
Which video-player package should you install? This is mostly a matter of preference, but the *totem* video player, particularly the *totem-xine* package, is a good one to try first, since it won't require too many extra packages to work and will integrate into the default Gnome desktop environment well.

After you install *libdvdread3*, you must run the script it provides to download and install the *libdvdcss2* libraries you need. Open a terminal and run the following script:

```
$ sudo /usr/share/doc/libdvdread3/examples/install-css.sh
Password:
--20:19:23--  http://www.dtek.chalmers.se/groups/dvd/deb/libdvdcss2_1.2.5-1_i386.deb
       => \Q/tmp/libdvdcss.deb'
Resolving www.dtek.chalmers.se... 129.16.30.198
Connecting to www.dtek.chalmers.se|129.16.30.198|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25,178 (25K) [text/plain]

100%[====================================>] 25,178     55.66K/s

20:19:25 (55.54 KB/s) - \Q/tmp/libdvdcss.deb' saved [25178/25178]

(Reading database ... 59605 files and directories currently installed.)
Preparing to replace libdvdcss2 1.2.5-1 (using /tmp/libdvdcss.deb) ...
Unpacking replacement libdvdcss2 ...
Setting up libdvdcss2 (1.2.5-1) ...
```

That's all. Now to play a DVD, just insert it into your computer's DVD player. Ubuntu is configured by default to automatically open DVD video with the Totem media player. If you want to toggle that setting, click System→Preferences→Removable Drives and Media, click the Multimedia tab in the window that appears, and check or uncheck the checkbox next to "Play video DVD disks when inserted" (see Figure 3-1). From this window, you can also change the default program used to open DVDs. Just change the default video player from *totem* to your program of choice.

Figure 3-1. The Removable Drives and Media Configuration Window



To start Totem manually, click Applications→Sound & Video→Movie Player, then click Movie→Play Disc to open your movie. Totem provides a simple interface to navigate through a movie—with standard Play/Pause, previous track, and next track buttons on the main window, along with a sliding bar you can use to quickly skip through a movie. When you are finished watching the movie, click Movie→Eject or hit Ctrl-E from within Totem.

Hack 31. Buy Songs at the iTunes Music Store

Yes, it's possible to access, buy, and download music from iTMS on Linux.

If you're not averse to spending a little money for a fantastically useful app and to running a proprietary, nonfree program (but then again, if you were, why are you using the DRM-laden iTunes Music Store?), CrossOver Office from CodeWeavers is a great way to get your iTMS fix. As a bonus, you'll also be able to run other Windows-based, proprietary programs, such as Microsoft Word, Excel, PowerPoint, and Internet Explorer, as well as multimedia apps like QuickTime and Shockwave. The advantage of CrossOver is that it makes it really easy to run these Windows programs in Linux, and you won't be skirting the terms of their software licenses. The disadvantage is that CrossOver costs $40, which is, IMHO, a pittance for all that it offers you. If you're not sure if you want to buy the program, you can try it for free for 30 days, which should be plenty of time for you to kick the tires and discover how useful this program really is.

To start with CrossOver, head over to http://www.codeweavers.com. CrossOver Office is featured prominently on the home page, with a big red Try Now button that you should press. Follow the instructions on the page and download the software via FTP, BitTorrent, or HTTP.

After downloading the CrossOver shell script, you need to make it executable and then install it. Both actions are performed as a normal user, without *sudo*:

```
$ chmod 744 install-crossover-standard-demo-5.0.1.sh
$ ./install-crossover-standard-demo-5.0.1.sh
```

A GUI installer will open. Click OK to accept the quite-reasonable licensing terms (basically that you won't copy the software and let everyone in the world use it) and then you'll be asked where you wish to install the software. By default, CrossOver will install at */home/*`user/`

cxoffice, but I don't like cluttering up my home directory, so I always change that to */home/scott/.cxoffice*. (There's a dot in front of *cxoffice*, which makes it a hidden directory.) Once that's in place, click Begin Install.

The program installs, and immediately the Introduction Wizard opens so that you can tell CrossOver about your Internet connectivity. Click Next, and you'll be asked if you use an HTTP proxy. Most likely you don't, so click Finish; if you do, enter the correct information and then click Finish.

You're back at the main Setup screen, but now there's a new button: Install Windows Software. Click it, and the Installation Wizard opens, displaying a list of supported Windows software, including various Microsoft, Adobe, Intuit, and other companies' apps. Scroll down until you see iTunes, and then click Next.

If it's not clear by now, you are expected to own a license for the software you install, or install freeware like Shockwave and QuickTime. CodeWeavers is a legit company and is not going to provide you with copies of software you're supposed to purchase.

The first screen of the install tells you that CrossOver is going to download and use iTunes 4.9.0, and that you should avoid upgrading. Listen to the folks at CodeWeavers! They test this stuff, and they know what they're talking about.

You're given a choice between the Express and Advanced install; go with the Express unless you absolutely know what you're doing and need the Advanced option. Click Next. CrossOver will download and install iTunes for you, and the whole time Apple's program will think it's running on Windows instead of Linux. I'm not going to walk you through an installation of iTunes, since presumably you're already familiar with how to install (and use) it. It's mostly just Next, Next, Next, anyway.

When CrossOver finishes, you'll reach the end of the Installation Wizard. If you're ready to quit CrossOver, click Finish. Before doing so, you ought to check the box next to "Remove installer files" so you can keep your hard drive tidy. If you had a problem with the installation and you want to help CodeWeavers diagnose the issue, check the box next to "Package log for CodeWeavers support", which saves a logfile detailing the process, so that tech support has some useful info. If you want to place more software on your system, check the box next to "Install more Windows software," which starts the wizard over again. Finally, "View installed associations" shows you what file types are now linked to the program you just installed. It may be handy to see, so feel free to check it.

Now that iTunes is installed on your Ubuntu machine, you'll find a shortcut for it on your desktop, as well as on your K menu (if you're using KDE) or Applications menu (if you're using GNOME). Start the program up, and you can buy as much DRM-laden music as you'd like.

I'm not really a fan of iTunes, or DRM in general. For my reasons why, see an article titled "The Big DRM Mistake" that I wrote for SecurityFocus, available at http://www.securityfocus.com/columnists/390.

If you ever want to uninstall iTunes, go to your K menu or Applications menu, and then to CrossOver→Configuration. Select iTunes, and simply press Repair/Remove.

I've used CrossOver to run a wide variety of software; in fact, I've written several books using the VBA-laden Microsoft Word templates that publishers love so much, and they've run beautifully under CrossOver. You'll find that CrossOver is an incredibly invaluable tool that can already run a lot of important Windows apps, including iTunes, with the list growing all the time. Try it out for 30 days and see what you think.

Scott Granneman

Hack 32. Get a Grip on CD Ripping

Use the Grip program to automate ripping audio CDs into music files.

The command line is definitely a powerful tool, particularly for automation, but it can also make doing a task like ripping a CD more trouble than it's worth, especially if you plan on tagging the resulting audio with metadata such as ID3 tags. While there are several frontends for command-line tools, Grip, in my opinion, is an excellent example of a GUI frontend that balances the power and configurability of the command line with the ease of use of a GUI interface. After you get to the end of this hack, your CD-ripping process will be so automated that once you start, you won't even have to pick up a mouse.

Install Grip

It is simple to install Grip: just install the *grip* package using your preferred package manager. Grip is a frontend in that for the most part, it uses other command-line utilities behind the scenes to do all of the work and simply provides an easy-to-use interface to configure the commands it passes down to those tools. Because it is a frontend, Grip can make use of many different command-line CD-ripping and audio-encoding programs, and, as such, it supports ripping to a number of popular formats including MP3, Ogg Vorbis, FLAC, or even a custom encoder of your choosing. This also means that to take advantage of those tools, you will need to already have them installed, but Ubuntu's package manager will take care of the major dependencies.

Configure Ubuntu to Default to Grip

By default Ubuntu uses a program called Sound Juicer to play audio CDs. This program is fine but lacks a lot of the configurability and power of Grip.

You could manually launch Grip each time you want to use it, but you can make Ubuntu launch it for you whenever it detects that an audio CD has been inserted. To do so, click System→Preferences→Removable Drives and Media to open the "Removable Drives and Media Preferences" window. Then, where it says Command under "Audio CD Discs" (on the Multimedia tab), replace `sound-juicer` (and its arguments, if any) with `grip`.

You may have to log out of and back in to GNOME for this change to take effect.

Configure Grip

Before you rip your first CD, you need to configure Grip. First, launch Grip from Applications→Sound & Audio→Grip. Grip's main interface is broken into a number of tabs:

Tracks

This tab displays the current list of tracks for a CD that has been inserted into the CD player and allows you to check which of the tracks you want to rip. Grip also functions as a CD player, so you can select a particular track from this tab and click the play button at the bottom of the window to play the track.

Rip

Here, you can see the current progress of any ripping and encoding you have scheduled and start or abort the ripping process.

Config

Under Config, you will find a number of subtabs that configure how Grip rips and encodes a CD.

Status

Look here for constantly updated text output for any jobs that have been done. You can look here to see any error messages or other output.

Help

This tab provides buttons to launch help for different categories, including how to play CDs, rip CDs, and configure Grip.

About

Here, you'll find information about Grip itself, including the version and a link to the official web site.

To configure Grip, click the Config tab to reveal a number of subtabs that configure different Grip settings. The tabs you are interested in are CD, Rip, Encode, ID3, and DiscDB.

Configure CD options

The first tab, CD, lets you configure your CD device. For the most part, the default settings will work, but for the purposes of automated ripping, make sure that the "Auto-play on disc insert" option is off. To test whether Grip has the correct CD-ROM device, insert an audio CD and see whether Grip sees and can play it. If not, make sure */dev/cdrom* is pointing to your correct CD-ROM device, often */dev/hdc* or */dev/scd0*.

Configure ripping options

The Rip configuration tab is where things start to get interesting. Because so many people are used to automated programs that turn their CDs into MP3s, they often don't realize that ripping is a two-stage process: first ripping the tracks from the CD into WAV files, and then encoding the tracks to MP3, Ogg, or whatever other format you wish. This tab controls the ripping stage of the process, and most of the options are pretty self-explanatory.

The first subtab, Ripper, lets you configure which CD-ripping program to use. Grip now includes its own version of *cdparanoia* by default, and I recommend that you use it unless you have a good reason not to. *cdparanoia* by default rips more slowly than most other ripping programs (on most of my CD-ROM drives it rips at 2x), but what it loses in speed, it more than makes up for in accuracy. *cdparanoia* is slow because it is particularly thorough about getting every bit it can from the CD. If you rip with faster but less thorough ripping programs, you may notice pops or gaps in your tracks. Even on many of my scratched-up CDs, *cdparanoia* has been able to recover the track.

Once you choose your CD-ripping program, you can configure it further in the Ripper tab. In the case of *cdparanoia*, you can disable a number of its default options, including what it calls "paranoia" and "extra paranoia"—how thorough it is about reading the CD. I recommend you leave these and the scratch-detection and repair options alone. The primary option in this subtab you should be interested in configuring is the "Rip file format" option. Here, you can tell Grip where to put and how to name the WAV files it rips. Grip uses a number of variables that correspond to metadata from the CD. Table 3-2 lists a number of the common variables and what they represent.

Table Grip naming variables

Variable	What it represents
`%A`	The artist name for the disc

Variable	What it represents
`%a`	The artist name for the track (useful for multi-artist CDs)
`%y`	The year of the disc
`%d`	The name of the disc
`%t`	The track number, zero-filled so that 3 would be shown as 03
`%n`	The name of the track
`%x`	The encoded file extension (*mp3* for MP3 files, *ogg* for OGG files, and *wav* for WAV files)

For example, if you stored your MP3s in a directory called *mp3* in your home directory, you might put in the "Rip file format" field:

```
~/mp3/%A/%y-%d/%t-%n.%x
```

Decoded, that line would turn Track 10 of the *London Calling* CD by The Clash, called "The Guns of Brixton," into the file *~/mp3/the_clash/1979-london_calling/10-the_guns_of_brixton.wav*.

You can use whatever naming scheme you wish for your audio files. I prefer this method because it organizes all my CDs by artist, then by each album sorted by date, then by each track. This way, no matter what audio player I use, by default the tracks are all in order.

With this subtab configured, click the Options subtab to get to other ripping options. There are a few options here that I like to enable, particularly "Auto-rip on insert" and "Auto-eject after rip." With these options enabled, when Grip is running it will automatically start ripping a CD for you when you insert it and then eject it when it's done. This means that once the rest of Grip is configured, you can set up a stack of CDs at your desk, start Grip, insert the first CD, and then minimize it and do something else. When you see that the CD has ejected, you can just replace it with another CD to rip and go on with what you were doing. Grip will handle the rest. It doesn't get much more automated than that!

Configure encoding options

The next main configuration tab is Encode. This tab lets you configure what kind of audio files Grip will encode the WAV files into. The first option, Encoder, lets you choose what encoding program to use. What you choose here depends heavily on what encoding programs you have installed and what kind of audio files you want. For instance, if you want to make MP3s, you will likely choose LAME, mp3encode, or your favorite MP3 encoder. I usually stick with LAME because it is fast and produces decent-quality MP3s. If you want to create Ogg Vorbis files, choose "oggenc." If you want to create FLAC files (a lossless audio codec so there is no degradation in quality), choose "flac." After you have chosen the encoder to use, make sure the encoder executable path points to the location of your encoder (the default should be fine here). The defaults for the next two options, the encoder command line and file extension, should be fine unless you choose to use a special encoder not directly supported by Grip. The next field, "Encode file format," takes the same type of information as the "Ripo file format" field in the Rip tab. In fact, if you make sure to use `%x` as the file extension, you can likely just directly copy and paste from this.

In the Options subtab for Encode, you can configure some specific options for your encoder. Probably the most important option here is the "Encoding bitrate" option, which determines the bitrate at which to encode your audio file if you use a lossy encoding such as MP3 or Ogg. What you put here is largely a matter of taste, although the higher the number, the larger your resulting file will be. In the case of MP3s, some people can't tell the difference between 128 and 256 kilobits per second. For other people, the distinction is great. I usually use 192 or 256 kilobits per second here, but you may want to experiment with the output for your audio files and determine what bitrate is best for you; your choice may vary depending upon the type of music you are encoding. In this tab, you also have the option to create an *.m3u* playlist file for each CD you rip and choose where to put it. Generally, the only other option I enable in this subtab is "Delete .wav after encoding." (*.wav* files are quite large, and once Grip has encoded them to MP3 or another format, there's no reason to keep the *.wav* around.)

Configure ID3 options

The next major configuration tab, ID3, lets you control whether Grip inserts ID3 tags into your audio files. Generally, this is a good thing, and you will want all these options enabled unless you want to go back later and manually set ID3 tags—an often tedious process.

Configure DiscDB options

The final configuration tab you might want to configure is the DiscDB tab. This tab lets you configure the primary and secondary CD database servers that Grip will query when you insert a CD. These servers contain information on many CDs, based on CD signatures. When you insert a CD, Grip will query this database and retrieve artist, disc, and track information for the CD so that it can automatically fill out the ID3 tags and name the files appropriately. I would recommend sticking with the default servers listed here unless you know what you are doing. Make sure that the "Perform disc lookups automatically" option is enabled here.

Rip a CD

Once you have configured Grip, the process to rip a CD is rather simple: just insert the CD into the CD-ROM drive. Grip will automatically scan the CD, retrieve the track information from your CD database servers, select all tracks for ripping, and start the ripping and encoding process. You can click on the Rip tab to monitor the progress of both the ripping and encoding to see how far along in the process you are. If for some reason you want to stop the ripping process, click the "Abort Rip and Encode" button.

If you notice that the track information that Grip retrieved is wrong, or if Grip was unable to retrieve the track information at all, abort the ripping and encoding process and then click the pencil icon at the bottom of the window. This will expand the window and provide a number of fields that you can use to fill out artist, title, genre, track name, and other CD information. Choose different tracks from the Tracks tab to change specific information for that track. Once you have finished your changes, you can click the envelope icon to submit your changes to your configured CD database so that the information is available to the next person who rips the CD. Once the changes have been made, select all of the tracks in the Tracks tab, and then go to the Rip tab and click Rip+Encode to restart the ripping process.

As I mentioned before, the nice thing about configuring Grip in this way is that you can let it run virtually unattended and just feed it new CDs until all your CDs are ripped—which certainly beats typing long commands and editing ID3 tags by hand!

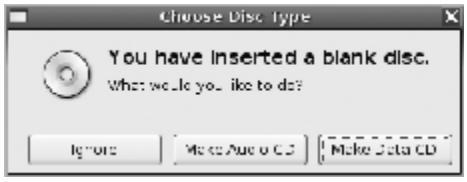
Hack 33. Burn CDs and DVDs

A few click-and-drag operations are all that separates your data from a CD or DVD.

While USB drives seem to be all the rage for transporting data, it's still hard to beat the price of blank CDs or DVDs. Whether you want to create an archive of some files "just in case," or you want to take some larger files with you to another computer, under Ubuntu the process to make a data CD or DVD only takes a couple of clicks.

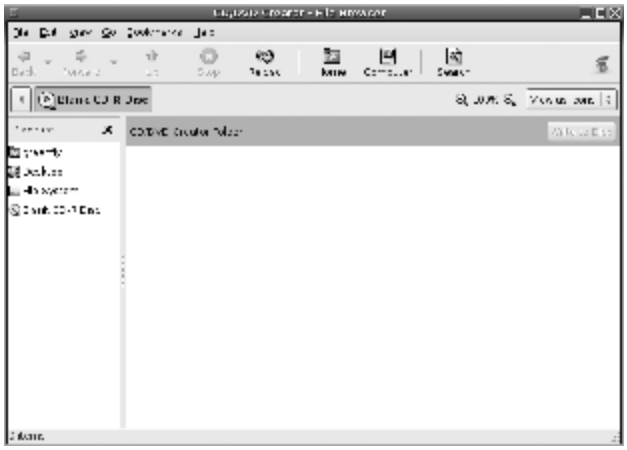
To copy data to a CD or DVD, first you need a blank CD or DVD. Insert your blank media into your CD/DVD burner and wait a moment as Ubuntu detects that you have inserted blank media into the drive. Figure 3-2 shows you the default prompt that Ubuntu will present you, where you will be given the choice to make an audio CD [Hack #34], make a data CD, or just ignore the CD. Choose "Make Data CD." Ubuntu will now open a special Nautilus window devoted to Data CD and DVD creation.

Figure 3-2. The prompt that greets you when you insert a blank CD or DVD



The CD/DVD Creator Folder is pretty basic, as Figure 3-3 illustrates. Essentially, you have an empty directory into which you can drag and drop files you would like to burn. Probably the easiest thing to do is to click the File menu at the top of the screen and open a new Nautilus window, and then drag files or directories from it into the CD/DVD Creator Folder.

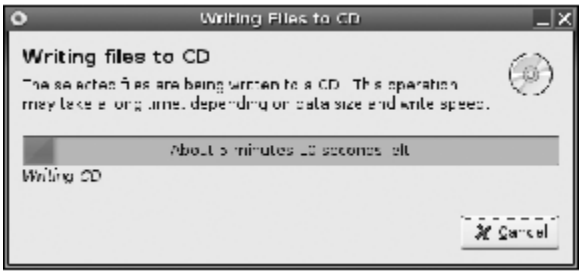
Figure 3-3. The default Nautilus CD/DVD burning window



Once you have collected all of the files you wish to burn into the folder, click the Write to Disc button at the top of the window. A new window will appear that lists some basic CD burning options in a number of drop-down menus. From here you can choose between multiple CD burners if your computer has them, assign a name to the disc, view the current size of the disc, and pick the write speed to use.

When you are ready to burn the CD, click the Write button and the CD burning process will begin. A simple status bar will appear so you can keep track of your disc's progress (see Figure 3-4).

Figure 3-4. The Nautilus Writing Files to CD dialog



If you find the options in the default CD/DVD creator limiting, I recommend checking out the *k3b* package. It isn't installed by default, so you will need to install it with your preferred package manager. K3B offers a number of advanced features and is featured along with the Serpentine audio CD creator in "Automate Audio CD Burning" [Hack #34].

Hack 34. Automate Audio CD Burning

Use Serpentine and K3b to burn your own custom audio CD in just a few clicks.

Even with the advent of high-tech hard drives, flash-memory-based media players, and sophisticated peer-to-peer file-sharing software, sometimes the simplest way to take your music with you is on a good old-fashioned CD. After all, many home and car stereos still don't support the playback of MP3s or other audio formats, so if you mix your favorite tracks and burn them to a CD, you can play them just about anywhere. Plus, if your CD breaks, it's quick, easy, and cheap to replace with another burned copy.

Creating a CD used to be a bit of a dark science under Linux and required you to use a number of command-line tools to convert audio files into WAVs if they weren't already WAVs. Then you had to execute another script to burn them onto a CD and, of course, hope that you calculated the song length correctly so your music would all fit on the CD. With Ubuntu, those days are over. Ubuntu offers a number of options for burning audio CDs, but this hack will cover the built-in option Serpentine, and K3b, a very powerful and very user-friendly graphical CD- and DVD-burning tool. Both tools accomplish all of the steps that you would normally have to do on the command line, all within a nice simple interface.

Serpentine is installed by default on your Ubuntu system, so you will only need to get K3b (and its MP3 decoding library, if you want to burn MP3s to disc). To do so, just install the *k3b* and *libk3b2-mp3* packages using your preferred package manager (for example, the command `sudo apt-get install k3b libk3b2-mp3` will do the trick). Ubuntu will also need to pull down a number of supporting libraries for K3b, so be patient as everything downloads and installs.

Use Serpentine

To use Serpentine, just insert a blank CD into your CD burner. Ubuntu will pop up a dialog window asking you whether you'd like to burn a data CD, burn an audio CD, or just ignore the CD (see Figure 3-2 in "Burn CDs and DVDs" [Hack #33]). Choose Make Audio CD.

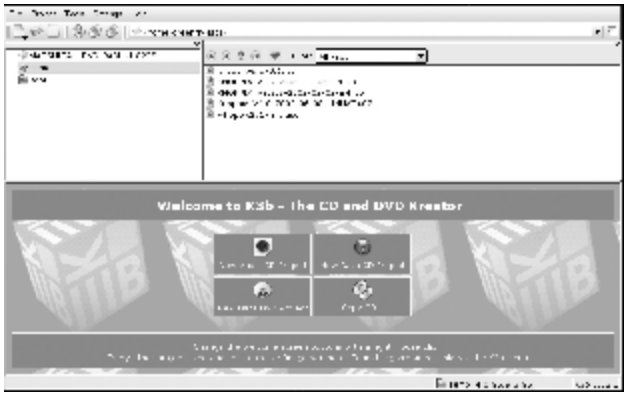
As Figure 3-5 shows, Serpentine's interface is very basic. To add tracks to the CD, either click the Add button to locate audio files on your file system or just drag and drop the tracks from your file manager. Serpentine will calculate the remaining time and display it on the main window. Once you are ready to burn the CD, click "Write to Disc," and Serpentine will start the process of converting and burning your tracks to the CD.

Figure 3-5. The default Serpentine window

Use K3b

To use K3b, either launch it from your application menu or type `k3b` in a terminal. The screen that initially appears offers some quick links to start common CD-burning projects (Figure 3-6), so click New Audio Project or select File→New→ New Project→New Audio CD Project. The split-screen interface shows you a view of your file system on the top of the window and a view of your CD project at the bottom. To add tracks to your CD, browse through the top interface to the WAV, MP3, and Ogg Vorbis audio files you want to add. You can also drag tracks from a CD you have inserted.

Figure 3-6. Main K3b window

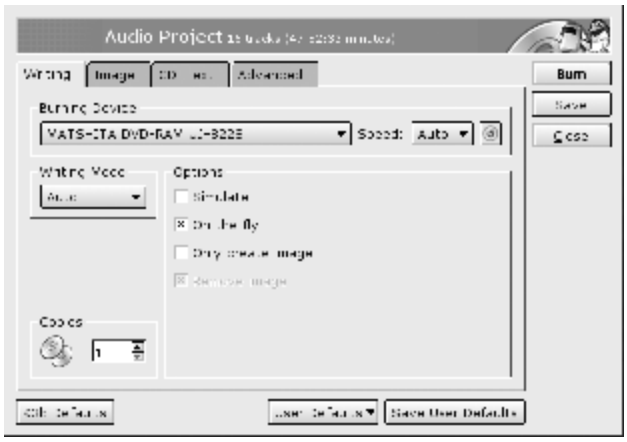


As you add tracks, K3b automatically figures out how much space you have left on the CD and displays it in a progress bar at the bottom of the window. K3b will also read any ID3 tags your music files may have and use the information to label the tracks on the CD, so that the artist

and title will show up on CD players that support CD-Text. Right-click on a track and choose Properties to edit any of the text fields, control what portion of the track to burn to CD, and change the gap length between tracks. If you want to save the project so you can burn another CD with the same settings at a future date, click File→Save As to store the project information to your disk.

Once you have arranged your new CD how you want it, click the Burn button at the bottom-righthand corner of the screen. The dialog that appears allows you to tweak any last-minute settings for your CD burner, including write speed, write mode, and which CD burner to use (see Figure 3-7). Unless you have specific needs, the default settings should work just fine. Once you have inserted the CD and are ready to burn, click the final Burn button in this dialog to start the burning process, or the Simulate button to simulate the process without actually writing to the CD. A new status window will appear and give you information on the progress of your burning session, including elapsed time, current track being burned, and other status information.

Figure 3-7. K3b Burn Dialog



While you can cancel the process at this point by clicking the Cancel button, doing so will almost certainly render your writeable CD useless, or, if it's a rewriteable CD, you will need to blank the CD before writing to it again (in K3b, select Tools→Erase CD-RW).

Hack 35. Rip and Encode DVDs

The acidrip utility gives you access to many of the common mencoder functions so you can use a GUI to rip and encode a DVD to a file.

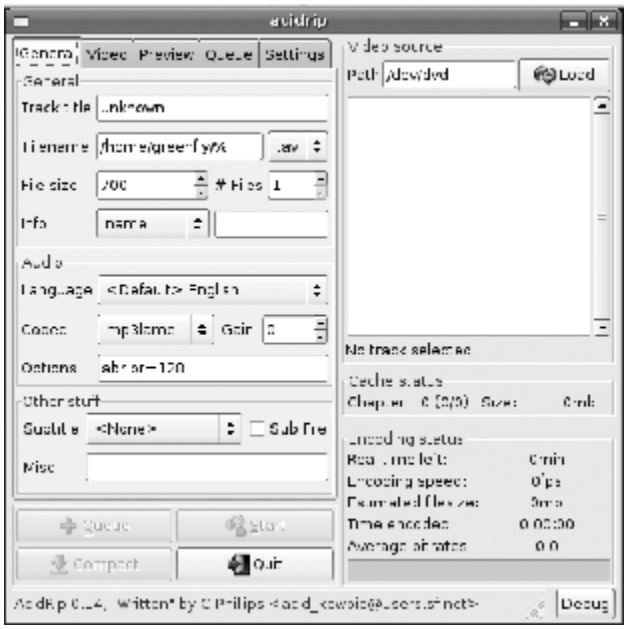
As with many things under Linux, there are a number of ways to rip and encode a DVD. For instance, some people use the *mencoder* tool (part of the *mplayer* suite) with two-pass encoding to turn VOB files they have extracted from a DVD into an MPEG4 *.avi* file. This method works great; however, some people are turned off by the thought of doing encoding entirely from the command line. If you want to use *mencoder* to encode a DVD but would rather have a GUI, the *acidrip* utility provides you with a GUI interface for most of the major mencoder options you might want.

Before you install *acidrip*, follow the instructions in "Install Multimedia Plug-ins" [Hack #28] and "Play DVDs" [Hack #30] to make sure that you have all of the multimedia plug-ins you'll need. After that, use your preferred package manager to install the *acidrip* package (it's in the Multiverse subcategory); if you use apt-get, type:

```
$ sudo apt-get install acidrip
```

With *acidrip* installed, click Applications→Sound and Video→Acidrip, or type `acidrip` from the command line to start the program. The default screen that greets you (see Figure 3-8) can be a bit intimidating at first, because it has so many options. For basic ripping, however, there are only a few options you need to worry about, and the nice thing is that *acidrip* will remember your settings for next time. That means that after you have it set up, you can rip multiple DVDs with minimal effort.

Figure 3-8. The default acidrip window



Load Your DVD

To rip a DVD, first locate the section on the right side of the window labeled "Video source" and type the path to your DVD in the Path field. If you are ripping directly from a DVD, insert the DVD and type the path to your DVD device (such as */dev/dvd*). If you have already ripped the DVD to the hard drive, type in the path to the directory that contains the DVD's *VIDEO_TS* directory. Click Load, and *acidrip* scans the DVD and displays each title with its playback time. Use the playback time to identify the main title you want to encode (generally the title with the longest playback time, and often the first title on the DVD) and select it.

If you are unsure of which title is the correct title, select the title and then click the Preview tab. Click the Preview button in that tab, and *acidrip* will use *mplayer* to play that track inside the window. Deselect the "Embed window" checkbox if you want to watch the video in its own window. If you enable any crop settings, you can also preview those here.

Configure Settings

Now click on the General tab on the left side of the window and fill in the "Track title" if *acidrip* didn't automatically detect it for you. By default, *acidrip* uses this field for the final filename. In the Filename field, type in the path where you want *acidrip* to put the final encoded video. The `%T` in this field is a variable that gets replaced with the contents of the "Track title" field. In the drop-down menu next to the Filename, you can choose whether to give the final video an *.avi* or *.mpg* extension. Underneath that field, you can configure the final file size for the video and whether to split it across multiple files. For instance, if you wanted to fit the video across two CDs, you would set "File size" to 700 and "# Files" to 2.

Those settings should be fine for typical DVD rippings, but *acidrip* provides plenty of other options so that you can configure mencoder for special cases. The General tab also lets you configure what codec to use for the audio track, along with whether to include a subtitle in the final video.

The Video tab gives the experienced *mencoder* user access to more advanced options. Here, you can configure which codec to use for the output file and can set the bitrate for the final file by hand. You can also crop the final video or scale it to a different width and height. If you want to add any special pre- or post-filters to *mencoder*, you can also configure that here.

I noticed that my version of *acidrip* had the `pp=de` pre-filter enabled by default, which caused an error in *mencoder*. I simply disabled this filter, and *acidrip* worked fine.

The Settings tab lets you configure general *acidrip* options, including what program to use for *mencoder*, *mplayer*, and *lsdvd* (useful if there is more than one instance of these programs in your path) and what directory to use to cache a DVD (if you have that option enabled). From this tab, you can also tell *acidrip* to automatically shut down the computer after it is finished ripping.

Queue and Rip

After you have configured your settings the way you want them, click the Queue button at the bottom of the window to add the job to the queue. The Queue tab displays any queued jobs you have set. This tab can be handy if you want to learn more about the *mencoder* processes *acidrip* uses, because it displays the full commands *mencoder* will run. This can also be good for debugging purposes if *acidrip* fails to start ripping. Copy and paste the *mencoder* lines from the queue to the command line and make note of any errors *mencoder* outputs. You can also queue more than one job, so you can line up a bunch of jobs and leave them running overnight.

Once you finish configuring your job and queue it, click the Start button to start the encoding process. *acidrip* shrinks down to a smaller window and displays its progress, including time left in the current process, encoding speed, and estimated file size. Click the Full View button to go back to the full-sized window. When *acidrip* finishes, it goes back to full view, and you will be able to check out your new video files in the directory you specified.

As mentioned earlier, the nice thing about using *acidrip* over *mencoder* is that you can tweak *acidrip* with your favorite *mencoder* settings a single time and then just concentrate on adding encoding jobs. This helps to eliminate the problem of trying to remember which options to use each time; plus it makes it easier to queue up multiple jobs, one after another.

Hack 36. Create a Video DVD

Use the tovid scripts to automate the conversion of many video formats to DVD.

Before DVD burners and media were relatively inexpensive, creating your own video DVDs was a daunting prospect. Even today, depending on which tools you use, it can still be a daunting prospect under Linux. However, with the tovid set of scripts (http://tovid.org), you can easily convert just about any video into a DVD-compatible format.

So you have a video (or a number of videos) that you want to convert into a DVD. The first step is to convert that video into a format compatible with the DVD standard. Although you could use *mencoder* or *transcode* directly to perform this conversion, the number of options involved can quickly get complicated and confusing. Luckily, a great tool, tovid, has been created to solve this problem. The tovid suite is a series of scripts that automate the process of converting a video into a VCD. The scripts involved have basic, easy-to-understand arguments and, because the output shows you the commands that are being executed, you can also use them to learn more about the underlying process.

Install tovid

To install tovid, download the latest release from the official project page at http://tovid.org. The main tool in the suite is also called *tovid* and uses the *mplayer*, *mjpegtools*, *ffmpeg*, *mkisofs*, *dvdauthor*, *transcode*, *vcdimager*, and *normalize-audio* packages to perform the video conversion, so you will need to have these packages installed beforehand (you'll need the universe and multiverse repositories enabled [Hack #60]). For example:

```
$ sudo apt-get install mplayer mjpegtools ffmpeg mkisofs \\
              dvdauthor normalize-audio transcode vcdimager
```

Once these requirements are met, download the latest release and untar it:

```
$ tar -xzvf tovid-0.25.tar.gz
```

Now enter the *tovid* source directory that *tar* created and then run the *configure* script inside. This script automates the process of installing tovid on your system, and once it completes, you are ready to start:

```
greenfly@ubuntu:~$ cd tovid-0.25
greenfly@ubuntu:~tovid-0.25$ ./configure
```

The *configure* script will confirm that you have all the required dependencies installed. If you are missing a dependency, check out "Search for Packages from the Command Line" [Hack #58] for information on how to track down the exact package name you need. Once *configure* has detected all the dependencies it needs, run the *setup.sh* script:

```
greenfly@ubuntu:~tovid-0.25$ sudo ./setup.sh
```

Convert the Video

With tovid installed, now it's time to convert the video. The *tovid* arguments are pretty basic. The only wrinkle is that you need to decide whether to use NTSC or PAL formats and which aspect ratio to use for the video so tovid knows how to properly resize the video. Whether to use NTSC or PAL formats depends on where you live (or, more specifically, what your TV uses). If you live in the United States, use NTSC. If you live in Europe or Japan, use PAL.

tovid supports full-screen (4:3), wide-screen (16:9), and theatrical widescreen (2.35:1) aspect ratios through the `-full`, `-wide`, and `-panavision` options, respectively. Generally speaking, if you are creating a DVD of a home video or TV show, you will probably use `-full` (which is what *tovid* uses by default if you don't specify the option). If the video source is from a movie, you will use `-wide` or `-panavision`, depending on how wide the video is. If you are unsure, run the *idvid* utility that comes with the tovid suite on the video file to output the width and height of the video, and then divide the width by the height:

```
$ idvid  sample.avi
----------------------------------
idvid video identification script
Version 0.25
Written in 2004 by Eric Pierce
http://tovid.sourceforge.net/
----------------------------------
Gathering video information. This may take several minutes,
so please be patient...
==================================================
File: sample.avi
Width: 512 pixels
Height: 384 pixels
...
$
```

In this example, the video aspect ratio is 512/384, or 4:3.

With the aspect ratio chosen, run *tovid* with the `-dvd` option to create the new DVD-compatible MPEG2 file. *tovid* also takes as arguments `-in` followed by the input file, and `-out` followed by the name to give the output file (without any file extensions):

```
$ tovid -dvd -ntsc -full -in sample.avi -out output
Probing video for information. This may take several minutes...
Input file is 512 x 384 at 23.976 fps.
Reported running time is 1267 seconds.
Source is not 29.970 fps. Adjusting to 29.970 fps.
Scaling and/or padding with letterbox bars
Scaling 512 x 384 directly to 720 x 480
The encoding process is estimated to require 886 MB of disk space.
You currently have 21396 MB available in this directory.
==========================================================
Testing mplayer stability with -vc dummy option:
Test succeeded!
Creating WAV of audio stream with the following command:
mplayer -quiet -vo null -ao pcm "sample.avi" -vc dummy -ao pcm:file=stream.
wav
```

```
=========================================================
=========================================================
Encoding WAV to ac3 format with the following command:
ffmpeg -i stream.wav -ab 224 -ar 48000 -ac 2 -acodec ac3 -y "output.ac3"
Audio encoding finished successfully
=========================================================
Creating and encoding video stream using the following commands:
nice -n 0 mplayer -benchmark -nosound -noframedrop -noautosub -vo yuv4mpeg -
vf-add pp=hb/vb/dr/al:f -vf-add hqdn3d -vf-add scale=720:480 "sample.avi"
cat stream.yuv | yuvfps -r 30000:1001 -n -v 0 | nice -n 0 mpeg2enc -M 2 -a 2
-f 8 -b 8000 -g 4 -G 11 -D 10 -F 4 -v 0 -n n -4 2 -2 1 -q 5 --keep-hf -o
"output.m2v"
```

If you have more than one video you would like to convert, you can use the *tovid-batch* command instead. *tovid-batch* takes the same arguments as *tovid*, except that you use `-infiles` instead of `-in` and you don't specify an output filename; *tovid-batch* will determine the output filename based on the input filename. So if you had a directory of full-screen *.avi* files you wanted to convert to DVD, run this:

```
$ tovid-batch -dvd -full -ntsc -infiles *.avi
```

Create the XML File

The next step is to create a proper *.xml* file to describe the DVD structure. You can use the *makexml* tool that is included with tovid to create an XML file that is compatible with the *dvdauthor* tool. *makexml* supports more options when used for DVDs.

Section 28 Table 3-3 lists the DVD-specific options.

AU: Do the -group and -endgroup options enclose the list of video files to include as one title? If so, perhaps clarify the description under Function?

Table makexml arguments

Argument	Function
`-group -endgroup`	List of video files to include as one single title. This is useful if you have split a movie into several video files.
`-titlesets`	Forces the creation of a separate titleset per title. This is useful if the titles of a DVD have different video formats—e.g., PAL + NTSC or 4:3 + 16:9. If used with menus, there must be a `-topmenu` option that specifies a menu file with an entry for each of the titlesets.
`-chapters` *n*	Creates a chapter every *n* minutes within the video. This option can be put at any position in a file list and is valid for all subsequent titles until a new `-chapters` option is encountered. Using this option may increase burn time, since the duration of the video is calculated.

These options are generally for special cases, apart from the last option. By default, *makexml* won't define chapters in your DVD, which means you won't be able to easily skip through it. To add chapters, use the `-chapters` option and specify an interval, such as 5 or 10 minutes. That way, you can more quickly skip through the DVD. To create an XML file for the sample video with a chapter every five minutes, type:

```
$ makexml -dvd -chapters 5  output.mpg output
-----------------------------------------
makexml
A script to generate XML for authoring a VCD, SVCD, or DVD.
```

```
Part of the tovid suite, version 0.18b
Written in 2004 by Eric Pierce
http://tovid.sourceforge.net/
----------------------------------------
Adding title: output.mpg as title number 1 of titleset 1
Calculating the duration of the video using the following command:
idvid -terse "output.mpg"
This may take a few minutes, so please be patient...
The duration of the video is 00:21:07
Closing titleset 1 with 1 title(s).
========================================
Done. The resulting XML was written to output.xml.
You can create the DVD filesystem by running the command:
    dvdauthor -x output.xml
    Thanks for using makexml!
```

where `output.mpg` is the name of the movie to work on and `output` is the name of the XML file to create (the *.xml* is appended automatically)

Create the DVD Filesystem Structure

With the XML file created, the next step is use *dvdauthor* to create the DVD filesystem. *dvdauthor* has a number of options you can use to create special DVD filesystems, but since *makexml* has already done the work for us, you can just pass your XML file to *dvdauthor* as an argument. *makexml* also listed the appropriate command to use near the end of its output, so to create a DVD filesystem for this example, type:

```
$ dvdauthor -x  output.xml
DVDAuthor::dvdauthor, version 0.6.11.
Build options: gnugetopt magick iconv freetype fribidi
Send bugs to

INFO: Locale=en_US
INFO: Converting filenames to ISO-8859-1
INFO: dvdauthor creating VTS
STAT: Picking VTS 01

STAT: Processing output.mpg...
STAT: VOBU 3184 at 529MB, 1 PGCS
INFO: Video pts = 0.178 .. 1268.077
INFO: Audio[0] pts = 0.178 .. 1267.506
STAT: VOBU 3194 at 530MB, 1 PGCS
INFO: Generating VTS with the following video attributes:
INFO: MPEG version: mpeg2
INFO: TV standard: ntsc
INFO: Aspect ratio: 4:3
INFO: Resolution: 720x480
INFO: Audio ch 0 format: ac3/2ch, 48khz drc

STAT: fixed 3194 VOBUS
INFO: dvdauthor creating table of contents
INFO: Scanning output/VIDEO_TS/VTS_01_0.IFO
```

dvdauthor will create a directory named *output* and store the *AUDIO_TS* and *VIDEO_TS* DVD filesystem there. If you want to test the DVD before you burn it, you can use *mplayer* to play from this filesystem with the `-dvd-device` option:

```
$ mplayer dvd://1 -dvd-device output/
```

This command plays the first title from the DVD filesystem under the `output` directory. If you want to play a different title, specify it on the command line.

Burn the DVD

Now it's time to burn the file structure to DVD. With K3B [Hack #34], click File→New Project→New Video DVD Project. Find your DVD filesystem in the top pane and then drag and drop the files inside the *AUDIO_TS* (if any) and *VIDEO_TS* directories into their respective directories in the bottom pane. Then click the Burn button to set the DVD-burning options and, finally, to burn the filesystem to DVD.

If you want to burn the DVD from the command line, you need to install *dvdrtools*, which is a fork of the *cdrecord* utility that is designed to support recordable DVD drives. *dvdrtools* is already packaged for Ubuntu, so install it with your preferred package manager.

Once *dvdrtools* is installed, the first step is to use the included *mkisofs* utility to create a DVD image out of your file structure:

```
$ mkisofs -dvd-video -udf -o dvd.iso output/
```

With the *dvd.iso* file created, you can locate the *dvd.iso* file in your file browser, right-click it, and select "Burn to DVD." Otherwise, if you prefer the command line, use the *dvdrecord* utility to burn it:

```
$ dvdrecord -dao speed=2
                dev=/dev/dvdrw
              dvd.iso
```

Replace the `dev=/dev/dvdrw` option with the correct values for your DVD burner.

With the DVD created, pop it in your DVD player and test your results.

Hack 37. Connect to a Digital Camera

Under Ubuntu, connecting to your digital camera is just about as easy as plugging it in.

I've used digital cameras under Linux in the past, and while it hasn't been too difficult to get pictures off of them, when I wanted to automate the process, I had to do a considerable amount of hacking. Not so under Ubuntu, where importing images from a camera, whether it's a USB storage device or not, is just a matter of plugging it in and clicking a few buttons. In this hack, I'll describe the general process to connect to a digital camera under Ubuntu.

Before I go into the individual steps in the process, I should mention that not all digital cameras are created equal. You can categorize digital cameras under Linux into two categories: cameras that have USB-storage-device support and cameras that don't. Cameras that have USB-storage-device support appear as a standard USB hard drives when you plug them into a computer. For instance, if you plug such a device into a Windows or Mac machine, you can browse through the camera just as if it were a USB thumb drive. Under Ubuntu this isn't too different—these devices will show up as a hard drive, and you can browse through them as such. There are also a number of cameras that don't tout USB storage device support. For these cameras, you will have to rely on Ubuntu's *libgphoto* libraries to communicate with the cameras over their specific protocols.

For the most part, under Ubuntu, I've had pretty good success importing photos, even on relatively new digital cameras that required *libgphoto*, but the best way to find out about your particular camera's support is simply to plug it in.

To import photos from your digital camera on Ubuntu, just plug the USB connector in and power on the camera. If Ubuntu recognizes the device, it will present you with a dialog similar to the one in Figure 3-9. If you have already imported your photos, then you can click Ignore; otherwise click Import Photos. The window that appears next will vary depending on which type of camera you have. These are different enough that I'll go into each of them.

Figure 3-9. The dialog Ubuntu presents when you plug in a digital camera

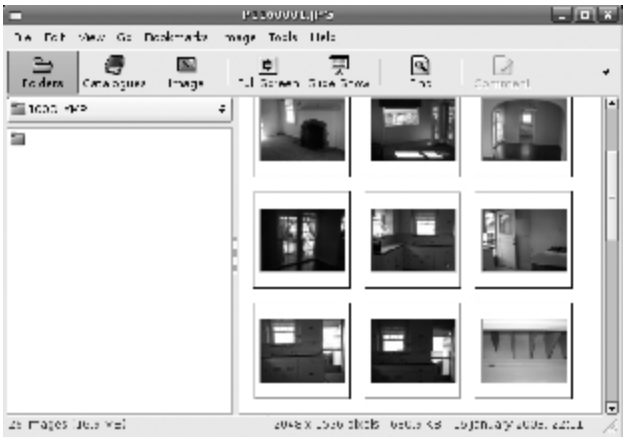
Import from a non-USB-Storage-Device Camera

If your camera does not have USB-storage-device support, Ubuntu will need to perform an intermediary step to import the photos from your drive. After you select Import Photos, Ubuntu will present you with a window like the one in Figure 3-10. This program uses *libgphoto* to pull the photos off of the camera. (While this program will allow you to rotate photos, I recommend you wait until the next step and rotate them in the gThumb program.) Select where you would like to import these photos from the Destination drop-down menu, and click Import. Ubuntu will then import each photo from the camera into a new directory in the destination that is conveniently timestamped so that you can keep track of when the photos were imported.

Figure 3-10. Import photos using libgphoto from this window

Once Ubuntu finishes importing the photos, it will open the gThumb program with that directory selected. Figure 3-11 shows how gThumb presents your photos in thumbnail format. From here, you click the Rotate button (or click Tools→Rotate), display a slideshow of every photo, or double-click on a particular photo to take a closer look. Since the import program already copied all of the photos off of your camera, you can now disconnect it whenever you please.

Figure 3-11. gThumb displaying my imported photos



Import from a USB-Storage-Device Camera

If your camera has USB-storage-device support, Ubuntu will bypass the intermediary step of importing the photos and instead take you directly to a gThumb display of your image directory, as shown in Figure 3-11. From here, you can rotate images, view slideshows, and examine your

images— however, note that any changes you make are made on the camera itself. If you want to copy the files to your computer, you will need to create a directory to store the photos; then you can drag and drop the photos from gThumb into your new directory.

Before you disconnect a camera that has USB storage device support, be sure to click Places→Computer, locate the camera's drive in the window, then right click on it and select Eject to be sure that any files have been synced and that the camera is safe to unplug.

Automatically Rotate Your Photos with gThumb

One nice feature of some newer digital cameras is the ability to store the orientation of the camera when a picture is taken. The image will then have stored within its metadata information that says not only when the picture is taken, but whether you took the picture in portrait or landscape mode. If your camera supports this feature, you can save a lot of effort rotating images within gThumb. Type Ctrl-A to select all of the images in the directory, then click Tools→Rotate to open the Rotate Images dialog. Then check the "Adjust photo orientation" and "Apply to all images" checkboxes in that window, as shown in Figure 3-12. Click Apply, and gThumb will go through each photo and rotate it according to the orientation stored in the metadata. This can be quite a time-saver when you have a lot of images.

Figure 3-12. Check these checkboxes to automatically rotate images

Chapter 4. Mobile Ubuntu

Ubuntu is a mobile-friendly Linux distribution. It will detect and configure almost any wireless hardware you can throw at it. In some cases, you will run into some hardware that, because its maker won't provide open source developers the information they need, can't run on Ubuntu. However, you can usually fiddle with the proprietary Windows drivers to get it working. And once you do get your wireless adapter up and running, you'll need to configure it to work with the various Wi-Fi networks you use. The hacks in this chapter cover all these topics and more.

For anyone who spends a lot of time away from a power source, power management is a major concern. This chapter also shows you how to put your computer into sleep and deep sleep, and provides some tricks on prolonging your battery life while are you are away from a power source. You'll also learn how to work with notebook-specific peripherals, such as PC Cards and hotswappable optical drives.

Hack 38. Put Your Laptop to Sleep

Close the lid and save some power.

Part of proper power management is the ability to put your laptop to sleep. *ACPI sleep* is defined as a state where the system is still technically powered on, but the screen and hard disk are powered down and the computer is using just enough power to keep the contents of RAM alive. The Ubuntu development team has devoted an immense amount of effort toward getting ACPI power management working properly. As it stands, Ubuntu is power-management-friendly right out of the box, thanks to the recent addition of the *gnome-power-manager* package. It turns out there's not much required to get most modern laptops to sleep and wake up correctly.

Getting Some Sleep

The Dapper Drake release of Ubuntu Linux includes the new *gnome-power-manager* package, which enables ACPI sleep much like the system-tray power applet in Windows. Finally, sleep "just works" in Linux. The *gnome-power-manager* applet is configured to start automatically, and it lives in GNOME's panel notification area. If you right-click on the little battery icon, you'll see a menu pop up, as shown in Figure 4-1.

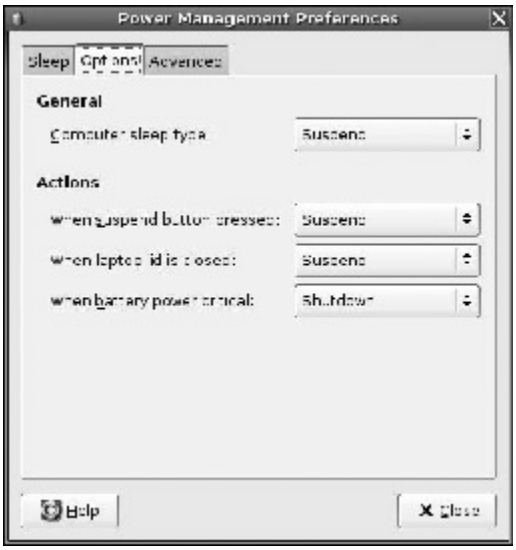
Figure 4-1. Gnome-power-manager in action

This deceptively simple little application gives you a lot of control over your laptop's sleep behavior. If you click on Preferences, you'll see the Power Management Preferences dialog box shown in Figure 4-2. In this dialog's Sleep tab, you can configure different behavior depending on whether you're plugged into AC power or running on battery. One of the most interesting features is your ability to control the backlight brightness of your laptop's screen depending on the machine's power state. It happens to work out that a large consumer of power in a laptop is the screen's backlight, so being able to automatically turn down that lamp while on battery will help squeeze more runtime out of the system while it's unplugged.

Figure 4-2. The Sleep tab of gnome-power-manager

The Options tab (see Figure 4-3) is where you can set the default type of sleep you wish for the system to use, as well as what actions will engage the sleep mechanism. For this hack, I have the default sleep type set to Suspend, which refers to ACPI sleep. (Hibernate [Hack #39] is a totally different type of sleep mechanism.) If you wish, under the Actions section of the dialog box, you can set the system to automatically sleep when the laptop lid is closed. This is a very handy feature if you're on the go: simply shut the lid and run off to your next appointment; then open the lid later, and the machine will wake up without any intervention.

Figure 4-3. gnome-power-manager's options



The Advanced tab of the dialog (Figure 4-4) controls the notification icon's behavior. If you want the icon to appear only when you're charging or discharging, or you want to turn the icon off all together, here's where you change those settings.

Figure 4-4. gnome-power-manager's Advanced dialog

If you're not seeing the notification icon, perhaps it's because you're plugged into AC power. If you'd like to see the applet all the time, unplug your laptop for a moment and the icon should appear. You can then use the Advanced tab to change the notification icon settings.

When you've got all your settings configured to your liking, simply click the Close button, and the dialog box will close, saving your configuration changes.

Testing ACPI Sleep

Your system is now ready for you to test sleep mode. Ensure that your system is running on battery; then simply shut the lid of the laptop and see what happens. You should hear the hard disk power down, and one of the power LEDs should indicate a power-state change by blinking or some other method. Hopefully, your machine went to sleep properly. Now you need to see if it wakes up correctly. Simply open the lid, and the computer should start waking up. When it's ready for use, you'll be prompted for your system password by *gnome-screensaver*. Enter your password, and your system should be in the exact same state as it was when you powered it off.

Thanks to the hard work of the Ubuntu developers, something that used to be extremely difficult to accomplish in Linux has been made very easy.

Hack 39. Hibernate Your Laptop

Sleeping is a good way to pause, but it still uses power. Hibernate mode lets you save the contents of memory to disk so you can resume later on.

In "Put Your Laptop to Sleep" [Hack #38], you learned how Ubuntu supports ACPI sleep. However, because ACPI sleep does require a trickle of power to keep the CPU and RAM alive, it may not be desirable in all situations. Hibernate has been designed as the complement to ACPI sleep. It takes the contents of RAM and saves them to the system's swap partition. As a result, it requires absolutely zero power to maintain that hibernated state. There is a downside to hibernating a machine, however. Due to the fact that the system saves the contents of RAM before powering off, and then loads the contents of RAM after the kernel loads on reboot, it does take a fair amount of time to enter and exit the hibernated state. However, hibernating is still faster than powering off and restarting your machine, and there's the added benefit of saving application state.

Since hibernation saves the exact contents of RAM to your swap partition, the size of your swap partition *must be larger* than the amount of RAM you have in the machine. Ubuntu configures this automatically if you have done a default installation, but if you override the default partitioner during installation, you need to keep this point in mind.

Configuring Hibernation

Since hibernation and sleep are similar and use the underlying Linux ACPI subsystem, they both use *gnome-power-manager* to configure and control their settings. You can adjust the default type of sleep to be hibernation from within the preferences of *gnome-power-manager*, but keeping the default set to Suspend makes sense due to the time involved in entering and exiting a hibernated state. If you do decide to adjust this setting, you can right-click on the *gnome-power-manager* icon, select Preferences, and click on the Options tab (see Figure 4-3 in "Put Your Laptop to Sleep" [Hack #38]).

Using Hibernation

Now that you've got your preferences set how you like them, it's time to test hibernation. Assuming you're using the stock Ubuntu preferences for *gnome-power-manager*, you'll need to engage hibernation by right-clicking on the *gnome-power-manager* applet and selecting "Hibernate." Once you do, the system will immediately dim the screen, and you should hear a good deal of hard disk activity. Once the disk stops churning, the system will power off. At this point, the system is in hibernate mode and can be left in this state indefinitely without using any battery power. To exit the hibernated state, simply power on the system normally. The bootloader will come up and the kernel will boot normally, until it detects a RAM image on the swap partition. At that point, the system will load up the RAM image and should return to where you left it. Typical times to enter hibernation run between 30 seconds and 1 minute, and times to exit hibernation (including the BIOS test) run about the same.

Between sleep and hibernation, you've got all the great power management capabilities at your disposal.

If your system is configured to boot multiple operating systems, you need to be careful here. In theory, you should be able to hibernate your Ubuntu system and then boot into a different operating system. But this is fundamentally risky: if you change anything on the Ubuntu partition, you'll be in heaps of trouble. And if you're sharing a swap partition between Ubuntu and another Linux distribution, you'll be in a world of trouble if that other Linux distribution boots up, since it will erase your hibernated state (or may itself try to resume from that hibernated state). Play it safe: if you are hibernated, don't boot into anything except the system you hibernated from.

Hack 40. Prolong Your Battery Life

Throttle your CPU, dim your display, and slow your hard drive to conserve precious battery power.

GNOME has a built-in CPU-frequency-monitor applet that will show you the current speed of your processor. This is great for laptops that have CPUs that can support dynamic frequency scaling. Additionally, the same applet will also let you alter the processor-speed governors and/or lock in the speed at a fixed frequency. This will let you override the built-in processor-speed governors for maximum performance or maximum battery life, depending on your needs at the time. This isn't overclocking or anything that's possibly damaging to your CPU; rather, this will let you use the built-in SpeedStep or other CPU-throttling techniques to their maximum.

This hack does not work for Transmeta-equipped CPUs with LongRun technology. However, Ubuntu does have *longrun* and other tools available via *apt-get* for these processors.

To be able to switch CPU speeds, you must set the *cpufreq-selector* program to be *suid root*:

```
bill@defiant:~$ sudo chmod +s /usr/bin/cpufreq-selector
```

This may be a slight security issue. If there is a some vulnerability in *cpufreq-selector*, anyone who exploits the vulnerability has the potential to get *root* access on the machine. See "Manage Security Updates" [Hack #68] for information on keeping up-to-date with the latest security fixes.

Once you've done this, enable the CPU Frequency Monitor if you don't have it running already. Right-click on your top GNOME panel, and select Add to Panel. Then select the CPU Frequency Monitor applet from the list and click on Add. The applet will appear in your panel. At this point, you can left-click on the applet and adjust the current CPU speed governor. If you right-click on the applet and select Preferences, you can change the menu that's displayed from "Governors" to "Frequencies and Governors." After setting this, you'll be able to tailor your CPU speed for any situation you may encounter (see Figure 4-5).

Figure 4-5. The CPU Frequency Monitor in action.

Your choice of Governor puts some constraints on your CPU speed choices. If you want to be able to specify the CPU speed yourself, choose the *userspace* governor, and then select the desired speed from the Frequencies menu. The *ondemand* and *conservative* governors adjust the speed based on demand, but *conservative* won't raise the CPU speed as quickly as *ondemand*. The *performance* and *powersave* governors will keep the CPU speed at either the maximum or minimum.

Hack 41. Get Proprietary Wireless Cards Working

If your computer has a Wi-Fi card that's not Linux-friendly, don't fret. Ndiswrapper and ndisgtk will let you use that card under Ubuntu by encapsulating the native Windows driver.

In "Roam Wirelessly" [Hack #42], you'll see that Ubuntu ships with built-in drivers for a good deal of the wireless network cards on the market today, such as the Intel "ipw" line of miniPCI adapters and the Prism 2/3–based PCMCIA network adapters. If you happen to use a computer that's equipped with such hardware, Ubuntu's wireless networking will likely "just work" right out of the box. However, systems equipped with Broadcom-based Wi-Fi adapters and the like traditionally don't work with Ubuntu or other Linux distributions. The primary cause for this is usually that the manufacturer doesn't release specifications for their hardware, which keeps open source developers from writing drivers for it.

Recently, however, there's been a development that enables almost any Wi-Fi adapter to work in Linux. It's known as ndiswrapper, and it works by "wrapping" the manufacturer's own drivers with a Linux driver layer that the OS can utilize. It's not ideal, but for obscure and difficult wireless hardware, ndiswrapper is often the way to go.

If you're using Linux on a platform other than x86 (such as on a Mac), then ndiswrapper won't work for you because it uses drivers that are compiled for x86 CPUs. However, there is an open source driver available for some Broadcom chipsets, including the one used in Apple's AirPort Extreme (802.11g). You'll still need native drivers for your chipset, though, but only to extract the firmware needed to operate the card correctly. See http://bcm43xx.berlios.de/ for more information.

Installing ndiswrapper and ndisgtk

Start by installing the *ndisgtk* package, which is a GUI installation program for ndiswrapper. You can open a terminal window and install it from the command line or use your favorite package-management tool. Please ensure you have the universe repository enabled [Hack #60]. (Aptitude will automatically get *ndiswrapper-utils*, since it's a dependency for *ndisgtk*.) Also, make sure you have the Windows drivers for your particular wireless adapter ready, since *ndisgtk* will prompt you for them later on. If you find that the Windows drivers are trapped in a *.cab* file, try installing *cabextract* and *unshield* and using those to get at the precious drivers trapped within (both of these are available in the *universe* repository):

```
conner@firetruck:$ sudo aptitude install ndisgtk
```

Wrapping the Windows drivers

Once that is done, insert or enable your wireless adapter, and run *ndisgtk* from a terminal:

```
conner@firetruck:$ sudo ndisgtk
```

ndisgtk will present you with a window (see Figure 4-6), allowing you to install a new driver.

Figure 4-6. Ndisgtk's main window

Click on "Install new driver," and *ndisgtk* will ask you for the location of your Windows drivers, as shown in Figure 4-7. Choose your Windows drivers in the Location field, and click Install.

Figure 4-7. Where are your drivers?

Once *ndisgtk* is finished copying your drivers and setting up the files, you will see the first window again, but *ndisgtk* will have populated it with your driver, as shown in Figure 4-8.

Figure 4-8. Windows-only hardware? I think not...

Configuring the Adapter

At this point, the wireless adapter is just another piece of Linux-enabled hardware. You can use the standard Ubuntu tools, such as the network wizard in the System menu or NetworkManager [Hack #42], to manage the connection. As a sanity check, you can run *dmesg* and see if ndiswrapper picked up your adapter:

```
conner@firetruck:$ dmesg
[4295357.170000] pccard: CardBus card inserted into slot 0
[4295357.181000] ndiswrapper: driver tnet1130 (Texas Instruments,06/17/2004,6.0.5.30) loaded
[4295357.182000] PCI: Enabling device 0000:02:00.0 (0000 -> 0002)
[4295357.182000] ACPI: PCI Interrupt 0000:02:00.0[A] -> Link [LNKD] -> GSI 11 (level, low) -> IRQ 11
[4295357.183000] PCI: Setting latency timer of device 0000:02:00.0 to 64
[4295357.876000] ndiswrapper (IoCreateUnprotectedSymbolicLink:744): --UNIMPLEMENTED--
[4295357.876000] ndiswrapper: using irq 11
[4295358.916000] wlan0: vendor: 'TNET1130'
[4295358.916000] wlan0: ndiswrapper ethernet device 00:12:0e:0d:3c:38 using driver tnet1130, 104C:9066:13D1:AB
[4295358.916000] wlan0: encryption modes supported: WEP; TKIP with WPA
```

Also, the standard *iwconfig* command-line tool can be used to configure your adapter, if you're comfortable with that:

```
conner@firetruck:~$ iwconfig wlan0
wlan0     IEEE 802.11g  ESSID:"NullDevice.Net"
          Mode:Managed  Frequency:2.462 GHz Access Point: 00:0C:41:6E:98:8E
          Bit Rate:54 Mb/s   Tx-Power:10 dBm  Sensitivity=0/3
          RTS thr:4096 B   Fragment thr:4096 B
          Power Management:off
          Link Quality:100/100  Signal level:-42 dBm  Noise level:-256 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

There is one flaw with ndiswrapper. Due to the way it operates, signal strength meters (such as those provided by NetworkManager [Hack #42] and the command-line *iwconfig* tool) do not work properly. Those tools will always report 100-percent signal strength.

Don't let the fear of incompatible wireless network cards keep you from using Ubuntu on your computer. Now you can free yourself and your operating system by using ndiswrapper on Ubuntu!

Hack 42. Roam Wirelessly

Moving from one wireless network to another has traditionally been a pain under Linux. Here's how you can simplify this aspect of mobile computing using your wireless adapter and NetworkManager.

One of the coolest things about having a laptop that's Wi-Fi-enabled is being able to go from place to place and freely associate with any hotspots that may be around. If you do this often, it really changes the way you work, as places like your local coffee shop have the potential to become your office. If you use a Windows or Mac OS–equipped laptop, bouncing from place to place isn't much of a challenge: you simply open up your computer, it tells you what access points were around, you click on one to associate with that network, and you're be off and running. Under Linux, however, that's been problematic, as there haven't been any tools that make Wi-Fi easy...until now.

Using Your Wireless Network Card

Ubuntu supports a good number of wireless cards and adapters out of the box; simply plug in your adapter and it should be recognized automatically (if not, see "Get Proprietary Wireless Cards Working" [Hack #41]). To verify your adapter is recognized, open a terminal window, and issue the following command:

```
bill@defiant:~$ iwconfig
lo        no wireless extensions.

eth0 no wireless extensions.

eth1      IEEE 802.11g  ESSID:"Its A Grind"
          Mode:Managed  Frequency:2.412 GHz  Access Point: 00:06:B1:14:C7:49
          Bit Rate=24 Mb/s   Tx-Power=20 dBm
          Retry limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=53/100  Signal level=-69 dBm  Noise level=-87 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:41  Invalid misc:0   Missed beacon:16
```

If you see output similar to what's above mentioning an IEEE interface, then your card is recognized by Ubuntu and you can either use the built-in networking tools (System→Administration→Networking) to manage it, or proceed on with this hack to install NetworkManager. If you don't see a wireless card listed there, then you'll need to follow the instructions in "Get Proprietary Wireless Cards Working" [Hack #41] to get your card working using ndiswrapper.

NetworkManager and You

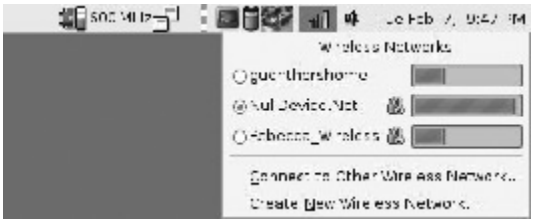
The folks over at Red Hat have come up with a tool that makes Wi-Fi management as easy as a couple of mouse clicks. The tool's called NetworkManager (see Figure 4-9), and it's a GNOME applet that sits in your notification area. NetworkManager will not only manage Wi-Fi connections, it'll automatically hook your laptop up with a wired connection when you plug an Ethernet cable in.

NetworkManager cannot manage any cards that have entries in */etc/network/interfaces*. If you've added your network card to that file, make sure you remove it before you start working with NetworkManager.

From the NetworkManager web site:

A laptop user should never need to use the command line or configuration files to manage their network; it should "Just Work" as automatically as possible and intrude as little as possible into the user's workflow. NetworkManager attempts to make networking invisible. When moving into areas you've been before, NetworkManager automatically connects to the last network the user chose to connect to. Likewise, when back at the desk, NetworkManager will switch to the faster, more reliable wired network connection.

Figure 4-9. NetworkManager in action



So, now that I've whetted your appetite, let's get NetworkManager installed. First, you've got to meet a few prerequisites:

- You must have a wireless Ethernet adapter, either built-in or PCMCIA.
- NetworkManager only works with DHCP-assigned addresses. If you *must* have a static IP address, work with your DHCP administrator to get a unique DHCP reservation, or you'll need to use another tool (such as System→Administration→Networking).

Let's get NetworkManager installed and working. It's surprisingly easy.

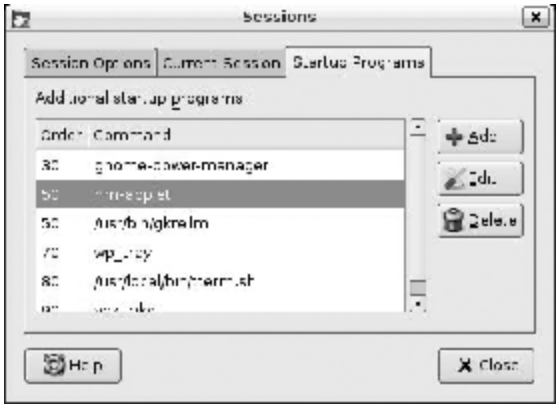
First, *apt-get* the package. From a terminal, run:

```
bill@defiant:~$ sudo apt-get install network-manager
```

You'll notice that the NetworkManager service gets started when the installation completes. All that remains is to configure GNOME to start the NetworkManager applet and to reboot the system, since changes to HAL (the hardware abstraction layer) were made in the previous step.

To add NetworkManager's applet to your GNOME autostartup configuration, click on the System Menu in your menu bar, select Preferences, and then click on Sessions (see Figure 4-10). Click on Startup Programs, and then select Add. Under Startup Command, type `nm-applet` and leave Order at "50". Click OK to commit the entry, and click Close to close the Sessions window.

Figure 4-10. The sessions dialog showing Startup Programs



Now all that remains for you to do is to reboot. Once you've rebooted and logged in to your machine, you'll see the NetworkManager icon in your notification area. If you're plugged into Ethernet, you should see a little "plug" icon. To use Wi-Fi, click on the NetworkManager icon and select the access point with which you want to associate. If the Wi-Fi access point requires an encryption code, you'll be prompted to enter it. Also, if this is your first time using the GNOME keyring services, you'll be prompted to select a keyring password. The GNOME keyring caches all the WEP keys for you, so all you need to remember is your GNOME keyring password.

That's all you need to get wireless roaming working, thanks to NetworkManager!

Hack 43. Make Laptop Settings Roam with Your Network

Use the laptop-net program to configure settings that change as you connect to different networks. Make your work settings follow you to work, and your home settings follow you home.

When you use a laptop, particularly as your primary system, you start to notice some of the ways most systems are biased toward a desktop computer. Primarily, the assumption is that all of your settings will stay the same no matter where you are. Yet, if you take your laptop to and from work you know this simply isn't the case. Not only do network settings often change, but printers, file shares, and all sorts of other aspects of the system are different depending on where you are. As a laptop user, it would certainly be nice if there were a way to have classes of settings that applied based on where I was. Lucky for me, the *laptop-net* package takes care of these basic needs and more.

The *laptop-net* program does a number of things, but the primary thing it allows you to do is to run particular scripts and apply particular settings based on which IP an interface currently has. Even though your home, work, and friend's house may all use private IP address ranges, they all probably use slightly different schemes, and more importantly, you probably get assigned a particular IP or range of IPs depending on which network you are on. With *laptop-net*, you can say "Whenever I have this IP, run these programs and enable these settings." Because of the way *laptop-net* is designed, what you can do is limited only by your scripting ability. In this hack, I will describe how to integrate *laptop-net* into a Ubuntu system and walk you through some common configuration schemes.

Install laptop-net

The first step is to install *laptop-net*. Using your preferred package manager install the *laptop-net* and *laptop-net-doc* packages. Once everything has installed, you will want to reconfigure *laptop-net*:

```
$ sudo dpkg-reconfigure laptop-net
```

Now by default, *laptop-net* can do a number of nice things such as detect cable insertion and automatically bring up and take down interfaces. I've noticed however, that these automatic functions can sometimes interfere with the way Ubuntu wants to configure the network—especially if you are using a package like NetworkManager—so you need to disable some of this functionality in *laptop-net*.

As the *laptop-net* configuration program runs, be sure to tell it that you are going to use DHCP for your network and that your Ethernet adapter does not support MII, even if it does.

Since *laptop-net* won't automatically engage when every interface is brought up, we will add a special script to */etc/dhcp3/dhclient-exit-hooks.d*. Scripts in this directory are executed when changes occur in *dhclient* (specifically, when the */etc/dhcp3/dhclient-script* runs and executes the `exit_with_hooks` function). Name the following file */etc/dhcp3/dhclient-exit-hooks.d/laptop-net*:

```
if [ $reason = 'BOUND' ]; then
   /usr/share/laptop-net/profile-change $interface
fi
if [ $reason = 'RELEASE' ]; then
   /usr/share/laptop-net/profile-change $interface down
fi
```

This script will tell *laptop-net* to change the profile based on the current interface when a new interface gets a lease, and when the DHCP lease is released, it will take that interface down.

I noticed a bug in my version of the package where the */var/run/laptop-net* directory was not created, so profiles weren't being changed. To fix this, I added an extra line to */etc/init.d/laptop-net*. Locate the following section in the file:

```
case "${1}" in
    ("start")
        [ $# -eq 1 ] || usage
        clean_state
        start_ifd
    ;;
```

and change it to:

```
case "${1}" in
    ("start")
        [ $# -eq 1 ] || usage
        mkdir -p /var/run/laptop-net
        clean_state
        start_ifd
    ;;
```

Now restart *laptop-net*:

```
$ sudo /etc/init.d/laptop-net restart
```

Configure laptop-net Profiles

laptop-net does most of its configuration work through different profiles that you need to configure. These profiles reside within the */etc/laptop-net/profiles* directory and contain the configuration information for specific modes and networks that you might want to connect to. The */etc/laptop-net/profiles* directory will end up containing one directory for each new profile you want to configure. A *laptop-net* profile itself can contain a number of different configuration files and scripts (check */usr/share/laptop-net-doc/* for more details), but here is a list of the main files:

patterns

: This file contains a list of IP or state patterns to match. If *laptop-net* matches a pattern in this file, it will then select this particular profile.

files.d/

: This directory contains a subset of your root filesystem. When this profile is selected, *laptop-net* will copy any files it finds here to the corresponding system directory, overwriting existing files with the same name. For instance, if you add a file in *files.d/etc/fstab*, when this profile is selected, that file will be copied over the top of the system */etc/fstab*.

after-select

: This is an optional shell script that *laptop-net* will execute after a profile has been selected. If there are any fancy tweaks or service restarts that you want to make, you can put them in this file.

before-deselect

: This script is also optional and is like the *after-select* script, only it is executed before a particular profile is deselected. You might use this script to undo particular changes you have made to a system before the network interface is taken down.

No matter what networks you want to configure for yourself, there are a couple of default profiles you should set up first: one for when the network is offline and one for when you are on some unknown network.

Configure the offline profile

To configure the offline profile, create a new directory called *offline* in */etc/laptop-net/profiles* and change to that directory.

```
greenfly@ubuntu:~$ sudo mkdir /etc/laptop-net/profiles/offline
greenfly@ubuntu:~$ cd /etc/laptop-net/profiles/offline
greenfly@ubuntu:/etc/laptop-net/profiles/offline$
```

Now create a file in this directory called *patterns* and have it contain the following lines:

```
down
unknown
```

Next, create a *files.d* directory here. Whenever you change system files with other profiles, it's a good idea to store the original default version of the file in the offline profile; that way, everything will go back to normal when the network is offline, and you won't risk overwriting a file for good:

```
greenfly@ubuntu:/etc/laptop-net/profiles/offline$ sudo mkdir files.d
```

The next step is to create a file called *after-select* and make it executable:

```
/etc/laptop-net/profiles/offline$ sudo touch after-select
/etc/laptop-net/profiles/offline$ sudo chmod a+x after-select
```

This file will run any programs or restart any scripts that you tell it to, but for now you can have it do something basic to give you diagnostic information when it is selected. Add the following lines to the file:

```
#!/bin/sh

echo "offline" > /tmp/laptop-net-state
```

Configure the default network profile

The next step is to create a "catch-all" default network profile. This profile will execute any network-based operations you want to perform whenever you have a network connection but don't quite know which network you are connected to. First create the profile directory and name it *zzz-default*. The reason you start with the *zzz* is that *laptop-net* goes through the profiles sequentially. Since this profile will match all IP addresses, you want to give your other profiles a chance to be selected first:

```
greenfly@ubuntu:/etc/laptop-net/profiles/$ sudo mkdir zzz-default
greenfly@ubuntu:/etc/laptop-net/profiles/$ cd zzz-default
greenfly@ubuntu:/etc/laptop-net/profiles/zzz-default$
```

Now create the same *files.d* directory and *patterns* file. If there are any configuration settings you want to always have whenever you are connected to a strange network, you can copy them to the *files.d* directory:

```
greenfly@ubuntu:/etc/laptop-net/profiles/zzz-default$ sudo mkdir files.d
greenfly@ubuntu:/etc/laptop-net/profiles/zzz-default$ sudo touch patterns
```

In the patterns file, put the following line:

```
*.*.*.*
```

This line will match any possible IP address.

The final step is to create the *after-select* script that is run when this profile is selected and to put the following lines in it for diagnostics:

```
#!/bin/sh

echo "zzz-default" > /tmp/laptop-net-state
```

Don't forget to run:

```
$ sudo chmod a+x after-select
```

to ensure the script is executable.

One thing I like to do when my *zzz-default* profile is selected is to throw up a very restrictive firewall. Since I'm connected to a presumably unknown network, this ensures that all the shields are up. You could also perform a number of other security-tightening measures in this script.

Test your configuration

With this new configuration in place, restart *laptop-net*:

```
$ sudo /etc/init.d/laptop-net restart
```

Now click System→Administration→Networking and deactivate your Ethernet connection (or wireless if that is what you are using). Now check */tmp/laptop-net-state*:

```
$ cat /tmp/laptop-net-state
offline
```

Now activate your network connection and check again once it comes up:

```
$ cat /tmp/laptop-net-state
zzz-default
```

You can also verify what *laptop-net* is doing by examining the *syslog* file. *laptop-net* will log there when it changes network settings, so you can *grep* the file for *laptop-net*, which will show you only its log information:

```
$ grep laptop-net /var/log/syslog
...
Mar  4 16:12:56 ubuntu laptop-net: Selecting network profile "zzz-default"
Mar  4 15:19:42 ubuntu laptop-net: Deselecting network profile "zzz-default"
Mar  4 15:19:42 ubuntu laptop-net: Selecting network profile "offline"
Mar  4 15:24:06 ubuntu laptop-net: Deselecting network profile "offline"
Mar  4 15:24:07 ubuntu laptop-net: Selecting network profile "zzz-default"
```

Create Personal laptop-net Profiles

Once you have confirmed *laptop-net* is functioning, it's time to create specific profiles for networks you commonly connect to. Let's assume I have two networks, one for home and one for work. To make things simple, I can just copy over the *zzz-default* profile and use it as a base template for new configurations:

```
greenfly@ubuntu:~$ cd /etc/laptop-net/profiles/
greenfly@ubuntu:/etc/laptop-net/profiles/$ sudo cp -a zzz-default home
greenfly@ubuntu:/etc/laptop-net/profiles/$ sudo cp -a zzz-default work
```

Configure the patterns file

The first thing you will need to change is the *patterns* file for each of these new profiles. Let's assume my home network assigns me an IP address somewhere in the 192.168.0.1 network, no matter whether it is on my Ethernet or wireless card, and my work network always assigns me 10.1.1.50 on my Ethernet card and 10.1.1.100 on my wireless card. I would then edit the *patterns* file in my home profile and make it contain only the following line:

```
192.168.0.*
```

This pattern will match any IP from 192.168.0.1 to 192.168.0.255. Next, I edit the *patterns* file in my work profile and make it contain two lines:

```
10.1.1.50
10.1.1.100
```

When you can, try to be as explicit as possible when defining patterns. That `192.168.0.*` pattern will match a great number of home networks, so you might end up setting your home profile when you connect at a friend's house. Ideally, you will try to set up your networks so that you get assigned one or two static IP addresses.

Configure the after-select script

Once the pattern file is set up, edit the *after-select* script in both the home and work profiles. Change the diagnostic line in that script from:

```
echo "zzz-default" > /tmp/laptop-net-state
```

to:

```
echo "home" > /tmp/laptop-net-state
```

and:

```
echo "work" > /tmp/laptop-net-state
```

in your home and work scripts, respectively. Now add any particular programs you want to run or other things you would like to do in these *after-select* files.

For instance, if you want your default printer to be set to your home printer when you are at home and your work printer when you are at work, configure your printer settings the way you want them for home, and then copy the */etc/cups/printers.conf* file to */etc/laptops-net/profiles/home/files.d/etc/cups/printers.conf* (you will need to create the *etc* and *etc/cups* directories within the *files.d* directory). Next, configure the printer settings to suit your work environment and copy the */etc/cups/printers.conf* to */etc/laptops-net/profiles/work/files.d/etc/cups/printers.conf*. Finally, in the *after-select* scripts for both home and work profiles, add the line:

```
/etc/init.d/cupsys restart
```

to refresh the settings when that profile is selected.

Repeat this sort of process for any other settings you want to change. Copy the relevant settings to the corresponding directory under *files.d*, and if a change requires the service to be restarted, add the corresponding command to the end of your *after-select* script.

For more information about configuration options in *laptop-net*, check out the documentation that was installed under */usr/share/doc/laptop-net-doc/*.

Hack 44. Make Bluetooth Connections

Create a personal-area Bluetooth network to exchange files, send messages, and share network connections.

Bluetooth allows you to create short-range (usually 10-meter), low-bandwidth (1–2 megabits per second), ad hoc connections between devices. It's ideal for sending photos from your camera phone to your computer, or for using the wireless modem found in certain models of cell phones.

To get started with Bluetooth, first make sure it's working. Some computers have built-in Bluetooth adapters. If you are using an external USB Bluetooth dongle, after you insert it you may need to run this command to reset the Bluetooth Host Controller Daemon:

```
$ sudo killall -HUP hcid
```

Now, to make sure Bluetooth is really up and running, you need to make sure you have at least one device in the vicinity that is set as "discoverable" (this is a Bluetooth mode that allows devices to find one another). Check your device documentation for details on how to set discoverable mode.

Some devices allow you turn discoverable mode on only for a limited time (for example, two minutes). So, make sure you execute the next command quickly. The *hcitool* utility lets you manage Bluetooth connections. Run the command `hcitool scan` to enumerate the discoverable devices in your vicinity. Here, *hcitool* has found three devices (my Nokia 3650 cell phone, my Mac, and my Palm Treo):

```
bjepson@scoobuntu:~$ hcitool scan
Scanning ...
        00:60:57:50:AB:9C       BrianJ3650
        00:14:51:89:4D:C7       Jepstone
        00:07:E0:6A:FE:54       bjepson
```

Pairing

Now that you know Bluetooth is working, you can *pair* your Ubuntu system with one of the devices. Pairing creates a bond between the devices so that they can exchange data and connect to one another.

First, initiate pairing from the device you want to use. Each device is different. On the Treo, for example, you need to tap the Bluetooth icon, and click Setup Devices→Trusted Devices→Add Device. Figure 4-11 shows my Palm Treo discovering several computers, including *scoobuntu-0*, my Ubuntu system. Select your Ubuntu system, and tap OK. When prompted for your passkey, pick something at random. A dialog should pop up on your Ubuntu system asking for the passkey. Type the same thing into that dialog, and click OK.

Figure 4-11. Palm Treo discovering nearby Bluetooth devices

Manual Pairing

If that dialog doesn't appear, you will need to take matters into your own hands. Cancel the pairing procedure on your device (or wait for it to time out), and go back to your Ubuntu system. Edit */etc/bluetooth/pin* (you need to use *sudo* to edit this, as in `sudo vi /etc/bluetooth/pin`) and change the PIN code there to the PIN you want to use as the one-time password for pairing devices (some devices only allow numeric PINs).

Next, create a script called */usr/local/bin/bluepincat* and make it executable:

```
#!/bin/sh
# file: /usr/local/bin/bluepincat

echo -n "PIN:"
cat /etc/bluetooth/pin
```

Comment out the existing `pin_helper` line in */etc/bluetooth/hcid.conf* (again, you need to use *sudo* to edit this file) and add the following:

```
pin_helper /usr/local/bin/bluepincat;
```

Next, reset the *hcid* daemon with `sudo killall -HUP hcid` and try to initiate pairing from your device again, this time using the PIN code that you put into */etc/bluetooth/pin* as the passkey. Once you're done pairing devices, you should restore the original configuration and reset the *hcid* daemon again (this avoids the possibility of someone guessing your PIN and pairing with your computer without your knowledge or consent).

Bluetooth File Transfer

Suppose you have something on your cell phone, such as a photo you took, that you want to send to your computer. If you install the *obexserver* package, you can receive files sent over Bluetooth:

```
$ sudo apt-get install obexserver
```

To receive files, you need to register the OBEX (Object Exchange) push service. This needs to be done each time you restart your computer (or the Bluetooth subsystem):

```
$ sdptool add --channel=10 OPUSH
OBEX Object Push service registered
```

Now, when you want to receive a file, start the obexserver, and send the file from the other device. The obexserver will terminate after the transfer:

```
$ obexserver
Waiting for connection...

..............................................................
..............................................................
..............................................................
..............................................................
................................................HEADER_LENGTH = 307254
put_done() Skipped header 05
put_done() Skipped header 42
put_done() Skipped header c0
Filename = screenshot0000.bmp
Wrote /tmp/screenshot0000.bmp (307254 bytes)

$
```

Connect to the Internet

Many Bluetooth phones, including certain Treo models as well as many Nokia and Sony Ericsson phones, can act as a gateway between your computer and the Internet. In this configuration, the phone is commonly thought of as a "wireless modem," but in fact, the connection between your computer and the Internet is digital all the way. Still, because this configuration uses the same protocol (PPP) as most dial-up connections, it's easy and sometimes helpful to just think of it as a modem.

Before you try this, you need to contact your wireless provider to find out whether this service is supported and how much it will cost you. Some providers, such as T-Mobile in the U.S., have inexpensive flat-rate pricing of between $20 and $50 a month, depending on your level of service. But most providers will charge you *by the kilobyte* if you don't have a data plan on your account. If you're not totally clear on how much it will cost, be careful, since cellular bills of hundreds or even thousands of dollars are not unheard of, especially when you throw international roaming into the mix!

To make the connection, you first need to set up a link between your Ubuntu system and the paired device. Edit (using *sudo*) */etc/bluetooth/rfcomm.conf* and add an entry for your device. You'll need to specify your phone's device ID (which you can get with `hciutil scan` as shown earlier):

```
rfcomm0 {
  bind yes;
    # Bluetooth address of the device
    device 00:07:E0:6A:FE:54;
    # RFCOMM channel for the connection
    channel     1;
    # Description of the connection
```

```
    comment "My Treo";
}
```

Next, use the command `sudo rfcomm bind rfcomm0` to make the connection. Now you've got a modem you can use on */dev/rfcomm0*. Next, you need to create some PPP scripts. The first one is a *chat script*. As *root*, create */etc/chatscripts/BluetoothPhone*. You will need to replace *APN* with the Access Point Name that your cellular provider uses (contact your cellular provider to get this information, or check Opera's list of APNs at http://www.opera.com/products/mobile/docs/connect/):

```
TIMEOUT 10
ABORT   'BUSY'
ABORT   'NO ANSWER'
ABORT   'ERROR'
SAY     'Starting GPRS connect script\\n'

# Get the modem's attention and reset it.
""      'ATZ'

# E0=No echo, V1=English result codes
OK      'ATE0V1'

# Set Access Point Name (APN)
SAY     'Setting APN\\n'
OK  'AT+CGDCONT=1,"IP","APN"'

# Dial the number
ABORT   'NO CARRIER'
SAY     'Dialing...\\n'
OK      'ATD*99***1#'
CONNECT ''
```

The next is */etc/chatscripts/BluetoothPhone-Disconnect*:

```
""      "\\K"
""      "+++ATH0"
SAY     "GPRS disconnected."
```

Finally, you need a PPP peers script (*/etc/ppp/peers/BluetoothPhone*):

```
/dev/rfcomm0  # Bluetooth modem
115200        # speed
defaultroute  # use the cellular network for the default route
usepeerdns    # use the DNS servers from the remote network
nodetach      # keep pppd in the foreground
crtscts       # hardware flow control
lock          # lock the serial port
noauth        # don't expect the modem to authenticate itself
local         # don't use Carrier Detect or Data Terminal Ready
debug

# Use the next two lines if you receive the dreaded messages:
#
#    No response to n echo-requests
#    Serial link appears to be disconnected.
#    Connection terminated.
#
lcp-echo-failure 4
lcp-echo-interval 65535
```

```
connect "/usr/sbin/chat -V -f /etc/chatscripts/BluetoothPhone"
disconnect "/usr/sbin/chat -V -f /etc/chatscripts/BluetoothPhone-Disconnect"
```

With all these scripts in place and your rfcomm device connected using `rfcomm bind`, run the command `sudo pppd call BluetoothPhone` to connect to the Internet anywhere you have cellular service. When you are done, press Ctrl-C to terminate the connection.

For more information on working with cellular data connections and Bluetooth, see *Linux Unwired* by Roger Weeks et al. (O'Reilly).

Brian Jepson

Hack 45. Expand your Laptop

Expand your laptop's capabilities through the use of PCMCIA cards.

Most laptops have PCMCIA (also called PC Card) slots for card-based expansion. Under Windows, before you install a PCMCIA card, you usually have to install a software driver in order for the card to operate. Under Ubuntu Linux, however, the drivers are typically built into the operating system, shipped as kernel modules. It's the job of the PCMCIA subsystem to load and unload these modules when a card is inserted.

Usually, you don't need to do anything more than insert a card for the system to recognize it and load the proper driver. If you open a terminal window and examine the system log, you can see the card-insertion event and note what driver the PCMCIA subsystem loads. The system log will also show what new devices were created for that particular card. You can view the log with the command `tail -f /var/log/messages`.

Here, I am inserting a 3Com 10/100 Ethernet adapter into my laptop's PCMCIA slot while monitoring the log:

```
Mar  8 23:54:19 localhost kernel: [4554419.074000] PCI: Enabling device 0000:02:00.0 (0000 -> 0003)
Mar  8 23:54:19 localhost kernel: [4554419.074000] ACPI: PCI Interrupt 0000:02:00.0[A] -> Link [LNKA] -> GSI 11
Mar  8 23:54:19 localhost kernel: [4554419.074000] 3c59x: Donald Becker and others. www.scyld.com/network/vorte
Mar  8 23:54:19 localhost kernel: [4554419.074000] 0000:02:00.0: 3Com PCI 3CCFE575BT Cyclone CardBus at 0x4000.
Mar  8 23:54:19 localhost pci.agent[17859]:      3c59x: loaded successfully
Mar  8 23:54:19 localhost kernel: [4554419.155000] ACPI: PCI Interrupt 0000:02:00.0[A] -> Link [LNKA] -> GSI 11
```

Once the driver is loaded, I can address this device via */dev/eth2*.

Some combination cards are also supported. In this example, I'm inserting a 10/100 Ethernet card and 56K modem combo card:

```
Mar  8 23:57:29 localhost kernel: [4554609.400000] eth2: NE2000 (DL10019 rev 05): io 0x300, irq 11, hw_addr 00:
Mar  8 23:57:29 localhost kernel: [4554609.442000] ttyS1 at I/O 0x2f8 (irq = 11) is a 16550A
```

The Ethernet portion of the card is addressable as */dev/eth2* on this machine, and the 56K modem is addressable as */dev/ttyS1*.

Some wireless cards, such as the classic Orinoco and Prism 2/3–based adapters also are supported by the PCMCIA subsystem out-of-the-box. In this case, I inserted an Orinoco adapter. It's also addressable as */dev/eth2*:

```
Mar  8 23:59:40 localhost kernel: [4554740.196000] eth2: New link status: Connected (0001)
```

Ubuntu even supports pretty obscure devices. I own an old Iomega Clik PCMCIA 40 MB hard disk and a Socket Communications serial card. Both devices work great under Ubuntu. The Clik drive is mountable as */dev/hdc4*, and the serial card shows up as */dev/ttyS0*.

```
Mar  9 00:10:57 localhost kernel: [4555416.871000] hdc: IOMEGA Clik! 40 CZ ATAPI, ATAPI cdrom or floppy?, assum
Mar  9 00:10:57 localhost kernel: [4555417.177000] ide1 at 0x180-0x187,0x386 on irq 3
Mar  9 00:10:57 localhost kernel: [4555417.182000] hdc: 39441kB, 78882 blocks, 512 sector size
Mar  9 00:10:58 localhost kernel: [4555417.185000]  /dev/ide/host1/bus0/target0/lun0: p4
Mar  9 00:10:58 localhost kernel: [4555418.047000] ide-cs: hdc: Vcc = 5.0, Vpp = 0.0
Mar  9 00:11:29 localhost kernel: [4555449.006000] ttyS0 at I/O 0x3f8 (irq = 3) is a 16550A
```

Just about any device you can put in a PCMCIA slot will work just great under Ubuntu (although some wireless cards need an extra configuration step [Hack #41]). Give it a shot!

Hack 46. Hotswap your Laptop's Optical Drive

If you've got a laptop with a removable optical drive, you can use the hotswap utility to remove and reinstall it without rebooting.

If you've purchased a laptop within the last couple of years, your machine most likely has an optical drive that's in a removable caddy or drive bay. Typically, these type of drives are enabled under Windows to be removed and replugged with the system power on, but the drivers that allow this haven't been available in Linux until recently.

The *hotswap* package included in the universe repository [Hack #60] provides the ability for you to hotswap an optical drive out and insert a battery or other module, and then reinsert the optical drive back into the bay, all without putting the laptop to sleep or shutting it down. To install *hotswap*, first you must have the *universe* repository enabled in your */etc/apt/sources.list*; then you can install *hotswap* by using the following command:

```
gooley@falcon:~$ sudo apt-get install hotswap
```

apt-get, as usual, will take care of all the dependencies and install all the software. Once *apt*'s completed the download and installation of the software, you can institute a hotswap by running *hotswap* via *sudo*. *hotswap* will autodetect your optical drive and ask you if you wish to remove it. Simply follow the prompts and do what the script says:

```
gooley@falcon:~$ sudo hotswap
Password:
I/O warning : failed to load external entity "/etc/hotswaprc"
hotswap 0.4.0
Copyright 2001 Tim Stadelmann
This program is free software, licensed under the conditions of the
GNU General Public License version 2, or (at your option), any later
version.

The following IDE device is currently configured:
HL-DT-STCD-RW/DVD DRIVE GCC-4242N
Do you want to remove this device? y
You can now remove the device from the module bay.
```

```
Would you like to insert an IDE device into the module bay? n
Aborting
```

At this point, you can eject and remove the optical drive. To reinsert the drive, simply run *hotswap* via *sudo* again (occasionally, *hotswap* will report an error, as shown here, but it will actually register the drive with the IDE subsystem):

```
gooley@falcon:~$ sudo hotswap
I/O warning : failed to load external entity "/etc/hotswaprc"
hotswap 0.4.0
Copyright 2001 Tim Stadelmann
This program is free software, licensed under the conditions of the
GNU General Public License version 2, or (at your option), any later
version.

There is currently no IDE device configured.  (Floppy disk drives,
batteries, and 'travel modules' are not managed by this utility.  If
you want to swap such a module, you should remove it now.)

Would you like to insert an IDE device into the module bay? y
Please insert the new device into the module bay and press RETURN.

No IDE device has been found in the module bay.  If you are sure that
the device you want to configure connects to the IDE bus, make sure
that the module is inserted completely.

Do you want to try again? n
Aborting
```

A quick check of *dmesg* will confirm that the drive did indeed get re-registered with the system:

```
gooley@falcon:~$ dmesg | tail
[4296527.332000] Probing IDE interface ide1...
[4296527.698000] hdc: HL-DT-STCD-RW/DVD DRIVE GCC-4242N, ATAPI CD/DVD-ROM drive
[4296528.004000] ide1 at 0x170-0x177,0x376 on irq 15
[4296528.008000] hdc: ATAPI 24X DVD-ROM CD-R/RW drive, 2048kB Cache
```

Another thing to note: hotswapping the optical drive usually results in a loss of the drive's DMA settings. As of this writing, the only way to re-enable DMA is to reboot the system.

Hotswapping is one more capability Linux has recently gained on laptops and portable machines. Now you can enjoy another ability that's been a Windows-only feature!

Chapter 5. X11

The X Window System (X11) is the basis of Ubuntu's user interface. X11 is responsible for managing your keyboard, mouse, or touchpad, and also takes care of the hardware acceleration features required by 3-D applications such as games. It's where you spend most of your time, so it's important to get it configured just right.

This chapter will help you customize X11 to work exactly the way you want it to. Although nearly every mouse, keyboard, or touchpad will work right out of the box with Ubuntu, you can use the X11 configuration file and some related utilities to get a lot more out of them. If you want to tune X11 to take advantage of all the acceleration features offered by your graphics adapter, spread your desktop across multiple screens, or get your fonts looking just right, the hacks in this chapter will help you out.

Hack 47. Configure Multibutton Mice

Seven buttons, a tilt/scroll wheel, and who knows what else? Find out how to put all those bells and whistles to good use.

Gamers love mice with more buttons than the Space Shuttle because the extra buttons can be mapped to provide quick access to common functions, but getting them working under Linux can be a bit tricky.

Open the Xorg configuration file at */etc/X11/xorg.conf* in a text editor (for example, using the command `sudo vi /etc/X11/xorg.conf`) and look for a stanza labelled `InputDevice`. If your computer is a laptop with a touchpad, it may have several `InputDevice` entries, so make sure you find the one that refers to your mouse. If it was configured automatically by Xorg, it will probably look something like this:

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

Start by changing the `Protocol` value. The *ExplorerPS/2* driver supports more devices than the older *ImPS/2* driver so substitute this line:

```
    Option        "Protocol" "ExplorerPS/2"
```

Since your multibutton mouse almost certainly has a middle button, you don't need the `Emulate3Buttons` option anymore, so delete it.

Unfortunately, there is no way for your computer to automatically determine the number of buttons available on a mouse, so you need to add an option that explicitly tells Xorg how many it has. It's obvious that you need to count all the actual physical buttons on the mouse, but remember that you usually need to add three more: one for clicking the scroll wheel, one for scroll-up, and one for scroll-down. A typical scroll mouse with two main buttons on the top, two on the side, and a scroll wheel actually has seven buttons as far as the driver is concerned, so add a line like this:

```
    Option        "Buttons" "7"
```

Next, map the action of the scroll wheel to virtual buttons using the `ZAxisMapping` option. In the case of a simple scroll wheel that moves only up or down, you can assign two values, which are associated with negative (down) and positive (up) motion, respectively:

```
    Option        "ZAxisMapping" "4 5"
```

Some mice have a scroll wheel that also rocks from side to side, and some even have two scroll wheels, in which case you can add mappings for negative and positive motion in the second axis:

```
    Option         "ZAxisMapping" "4 5 6 7"
```

Unfortunately, you may find the second scroll direction isn't recognized by Xorg at all because there is currently no standard for defining how mice should encode horizontal scroll data when it's transmitted to the driver. Even if it is recognized, you may find the rocker or second scroll wheel moves in the opposite direction to what you expect, so you may need to reverse the third and fourth values.

Some browsers such as Firefox are hardcoded to use buttons 4 and 5 as shortcuts for "back" and "forward," but because some wheel mice report wheel-up and wheel-down as the fourth and fifth button events, respectively, you may need to do some extra work to use the side buttons as back and forward. You can remap the reported button events by calling *xmodmap*:

```
xmodmap -e "pointer = 1 2 3 6 7 4 5"
```

The *xmodmap* command will need to be run each time you log in to GNOME, so go to System→Preferences→Sessions→Startup Programs and put in the whole line; then compensate for the offset button values by using a modified `ZAxisMapping` line in */etc/X11/xorg.conf*:

```
    Option       "ZAxisMapping" "6 7"
```

One final option you can configure is mouse resolution. Many multibutton gaming mice run at very high resolutions to enable accurate targeting, but you may find that it throws off Xorg's response curve. In that case, it may help to add a `Resolution` option in dpi (dots per inch):

```
    Option       "Resolution" "2400"
```

Once you have made all of these changes, your mouse configuration will look something like this:

```
Section "InputDevice"
    Identifier   "Configured Mouse"
    Driver       "mouse"
    Option       "CorePointer"
    Option       "Device" "/dev/input/mice"
    Option       "Protocol" "ExplorerPS/2"
    Option       "Buttons" "7"
    Option       "ZAxisMapping" "4 5"
EndSection
```

To apply changes, you need to restart Xorg. The easiest way to do so is to log out of your current session and then press Ctrl-Alt-Backspace, which will kill Xorg and force GDM to restart it (if GDM doesn't restart it, log in at the console and run the command `sudo /etc/init.d/gdm restart`).

Log back in and launch `xev`, the X EVent reporter, and click each button and scroll the scroll wheel in both directions. Each event will cause a button number to be reported in the terminal. If everything went as planned, each button will report a different button number.

Hack 48. Enable Your Multimedia Keyboard

Make the most of the additional keys on your multimedia keyboard.

When you press a key on your keyboard, it detects the key press and sends a corresponding "keycode," which is then matched using a lookup table that converts the keycode to a character. However, many modern keyboards have a variety of additional keys that typically return keycodes that aren't included in the standard lookup table, so Linux doesn't know what to do with them.

In Ubuntu, you can use these extra keys as shortcuts for such tasks as opening email, launching a browser, or changing audio volume. With the help of GNOME Preferences, you can configure some special actions, but for maximum flexibility, Hotkeys lets you completely customize shortcut-key behavior.

Assign GNOME Keyboard Shortcuts

If you want to create shortcuts only for basic tasks, start by opening System→Preferences→Keyboard Shortcuts to see a list of predefined actions, as shown in Figure 5-1.

Click on the shortcut entry for an action and then press the key combination you want to associate with it. For example, if you want F7 to launch your web browser, click the shortcut entry next to "Launch web browser" and then press F7. Changes are applied immediately, so there is no need to save the new settings.

Figure 5-1. Use System Preferences Keyboard Shortcuts to assign common actions to keys

The built-in GNOME keyboard-shortcut settings include only predefined actions, so if you want to launch a custom script or an arbitrary program, you'll need something more flexible, such as Hotkeys.

Hotkeys

Hotkeys is a program that intercepts keycodes from multimedia keys and maps them to specific events. The events are totally user-definable, so you can configure keys to run programs or even custom scripts. For example, in addition to basic actions such as controlling volume, opening email, or launching a browser, you can set keys to do things such as initiate an SSH connection to a remote machine, change your desktop background, launch OpenOffice.org, start a backup, or pretty much anything else you can think of.

Hotkeys is in the *universe* group of extra software that's not officially supported by Ubuntu, so if your computer is not set to use software from *universe*, you may need to "Modify the List of Package Repositories" [Hack #60] before you can install it:

```
$ sudo apt-get install hotkeys
```

Understand Hotkeys configuration parsing

Because Hotkeys uses a two-stage configuration system, it may not be immediately obvious what you need to change to customize commands, and, sometimes the changes you make don't seem to be applied. However, it all makes perfect sense once you understand how Hotkeys parses its configuration files.

The first stage is the keyboard definition file, which does two kinds of mapping: indirect conversion of keycodes to predefined events, and direct conversion of keycodes to user-defined commands. The second stage is the Hotkeys configuration file, which does a second mapping for predefined events, associating them with explicit commands.

This approach allows Hotkeys to map the keycode generated by the Favorites button on your keyboard with a predefined Favorites event using the keyboard definition, and then map the Favorites event to a specific command such as `gnome-moz-remote --remote=openBookmarks` in the Hotkeys configuration. That way, the majority of standard actions can be defined in the Hotkeys configuration, and making them work with a different keyboard is just a matter of switching the keyboard definition file.

However, you can also define custom events right in the keyboard definition. This approach maps a keycode straight to a command directly without referring to the Hotkeys configuration file at all. This is how you will probably define more unusual actions, such as launching a terminal and automatically opening an SSH session to a predefined host.

Set keyboard type

The first step to configuring Hotkeys is to define your keyboard type so the correct keycodes can be recognized. Hotkeys comes with sample configurations for a variety of standard keyboard types, so if you're lucky, you may be able to just select your keyboard from the samples and everything will just work. To see a list of supported keyboard types, run:

```
$ hotkeys -l
```

Then you can specify one of those keyboard types when you launch hotkeys using the `-t` flag.

But what if your keyboard isn't in the list of supported models?

No problem. You can define a custom keyboard type by copying the */usr/share/doc/hotkeys/sample.xml* sample file into a *.hotkeys* directory inside your home directory, and you can then customize the entries in it to match the keycodes returned by your keyboard:

```
$ mkdir ~/.hotkeys
$ cp /usr/share/doc/hotkeys/sample.xml ~/.hotkeys/mykeyboard.def
```

Be sure to check over the XML in this file. At the time of this writing, the `<description>` tag at the end of the file was improperly closed (it should be `</description>`).

If you then run `hotkeys -l` again you'll see there is now a keyboard type of *mykeyboard* at the top of the list in addition to all the default keyboards. You can later refer to your custom keyboard layout by name when launching Hotkeys.

The sample file is commented, but it may make more sense if you have a look at one of the keyboard definition files provided in */usr/share/hotkeys/* to see how they are structured. You may even prefer to start by copying one of the supplied definitions instead if you find one that is close to what you want.

To set up the file, you will need to know the exact keycodes for the various multimedia keys you wish to use. You can detect keycodes using a utility called *xev* ("X events"), which does nothing but watch input devices including the mouse and keyboard and display the codes of any events sent to X. By running *xev* and then pressing each of your multimedia keys in turn, you can see exactly what is being returned and then add those codes to your *.hotkeys/mykeyboard.def* file. First, install *xev*:

```
$ sudo apt-get install xev
```

Then launch *xev* from a terminal to open a small test window. Set mouse focus on the test window and then press one of the multimedia keys. In the terminal that you used to launch *xev*, you will see several lines of event information reported for each key press. What you need to look for is an entry that says `keycode` with a number after it. That number is the keycode that is associated with that particular key by X. Write down a list of the multimedia keys and their matching keycodes.

Once you have finished with *xev*, you can close it using Ctrl-C in the original terminal or by clicking the close box in the window manager.

Next, you need to open your *.hotkeys/mykeyboard.def* config file in a text editor and match events to keycodes according to the list you just made. For example, to match keycode 178 to the `Favorites` event, you would make an entry like:

```
<Favorites keycode="178"/>
```

The sample file contains a number of predefined events that you can uncomment and configure to suit your keycodes.

Customize predefined actions

Now that Hotkeys has a mapping between keycodes and predefined actions, you can customize the command associated with each action. The global config file is located in */etc/hotkeys.conf*, but if you want to customize the action list, you can make a local copy that will automatically override the global config:

```
$ cp /etc/hotkeys.conf ~/.hotkeys/hotkeys.conf
```

Following the example above, you could tell Hotkeys to open your favorites list in Mozilla with an entry like:

```
Favorites=gnome-moz-remote --remote=openBookmarks
```

Create custom actions

The previous step allows you to customize the predefined actions, but that's not much use if you want an action like "Launch *xterm* with reversed video and open an SSH session to *www.example.com*"! No problem: you can use the keyboard definition file to associate keycodes with any arbitrary command you like. For example, if you wanted to use the F2 key for a purpose such as opening an SSH session to a remote machine, you could use *xev* to find the keycode for F2 and edit *~/.hotkeys/mykeyboard.def* and put in:

```
<userdef keycode="68" command="xterm -fg black -bg white -e ssh www.example.com">SSH www.example.com</userdef>
```

In this case, `68` is the keycode for F2, and `SSH www.example.com` is the event label that will be displayed onscreen by Hotkeys when it executes the command.

Test Hotkeys

Nearly there! By now, you have a keyboard definition file that maps keycodes to actions, plus a Hotkeys configuration file that maps actions to commands. Now it's time to launch Hotkeys and try it out.

Specify the keyboard type and launch *hotkeys* itself:

```
$ hotkeys -t mykeyboard -b
```

The `-b` flag tells Hotkeys not to launch itself into the background; that makes it easier to make quick changes if things don't go quite as planned, because you can just Ctrl-C to kill the program and launch it again. Later, when you're happy with the configuration, you can leave off the `-b`, and Hotkeys will automatically background itself on startup.

Alternatively, if Hotkeys is backgrounded and you want to make changes, you can simply use:

```
$ killall hotkeys
```

and then launch it again.

As soon as you launch it, Hotkeys will start watching for key-press events, so try pressing one of the keys you mapped and see what happens.

To save yourself the hassle of passing the keyboard-type argument to Hotkeys every time, you can put an entry in *.hotkeys/hotkeys.conf* like this:

```
Kbd=mykeyboard
```

which will then be applied automatically.

Onscreen display settings

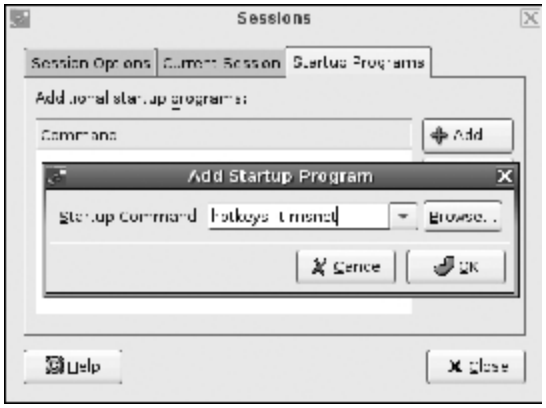
Hotkeys displays each action onscreen while it's executing it to give you feedback that something is happening. By default, the OSD is large, bright green, and hangs around on your display for a couple of seconds, so you may want to change the settings to make it less obtrusive.

Look at the bottom of your *hotkeys.conf* file for settings starting with `osd_`. The sample configuration is well commented, so make changes as you see fit, and then kill and relaunch Hotkeys for your changes to take effect.

Start Hotkeys Automatically

Because Hotkeys needs to be running to do its job, you will need to launch it every time you log in to X. That can be a pain, so to have it start automatically, just add it to the Startup Programs in System→Preferences→Sessions. Click Add and type in the command you used to launch Hotkeys from the command line; then save it (see Figure 5-2).

Figure 5-2. Add Hotkeys to Startup Programs in your session



Next time you log in to GNOME, Hotkeys will be started for you automatically and your multimedia keys will be functional right away.

Hack 49. Configure a Touchpad

Configure the special functions of your laptop touchpad and start tapping like crazy.

Most laptops today include a touchpad as a method of controlling the mouse pointer. These touchpads have various configuration options that most people aren't even aware of. Here's how to get all these advanced features working under Linux.

Ubuntu includes the *synaptics* mouse driver by default, and if your laptop's touchpad is detected during installation, it will be automatically installed and configured. The stock touchpad configuration will include some advanced features like double-finger tap support for middle-clicks and using the right side of the touchpad to do vertical scrolling. To really tweak your touchpad configuration, though, you've got to make a small edit to your X11 configuration, restart the GUI, and install a configuration program.

Preparing Your X11 configuration

In order to use the configuration program for your touchpad, first you've got to edit the X11 configuration to add a directive to enable shared memory in the Xserver. If you're fairly new to Linux, I suggest you use the *nano* editor; otherwise, use the editor of your choice.

The permissions on the X11 configuration are restricted, so you need to make this change using *sudo*.

Back up the configuration file first:

```
bill@lexington:~$ sudo cp /etc/X11/xorg.conf /etc/X11/xorg.conf.old
```

Then edit the file:

```
bill@lexington:~$ sudo nano /etc/X11/xorg.conf
```

Once you get the file open, you should see a section pertaining to the touchpad. It may look like this:

```
Section "InputDevice"
   Identifier      "Synaptics Touchpad"
   Driver          "synaptics"
   Option          "SendCoreEvents"      "true"
   Option          "Device"              "/dev/psaux"
   Option          "Protocol"            "auto-dev"
   Option          "HorizScrollDelta"    "0"
EndSection
```

You'll need to add a line in that section of the file (before the `EndSection` line) with the `SHMConfig` option set to `on` (this will allow you to use the GUI client to make changes without restarting the Xorg server). A sample snippet after editing:

```
Section "InputDevice"
   Identifier      "Synaptics Touchpad"
   Driver          "synaptics"
   Option          "SendCoreEvents"      "true"
   Option          "Device"              "/dev/psaux"
   Option          "Protocol"            "auto-dev"
   Option          "HorizScrollDelta"    "0"
   Option          "SHMConfig"           "on"
EndSection
```

At this point, save the file and reboot your system (or log out, switch to a console with Ctrl-Alt-F1, log in, and run the command `sudo /etc/init.d/gdm restart`) to load the new X11 configuration. The GUI should come up as usual. If there was a problem, to get your GUI back, simply log in at the command prompt and run the following command:

```
bill@lexington:~$ sudo cp /etc/X11/xorg.conf.old /etc/X11/xorg.conf
```

Assuming that your configuration works, you should now install the configuration program for the touchpad: *qsynaptics*.

Installing qsynaptics

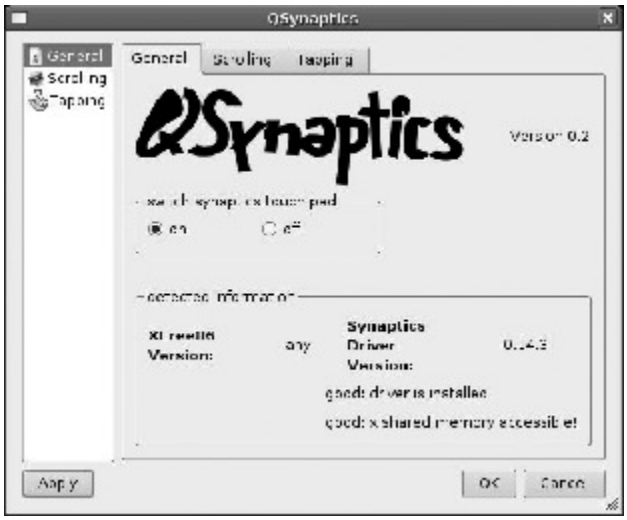
Much like any other software installation, *qsynaptics* can be installed by using *apt-get* or *aptitude* from a terminal:

```
bill@lexington:~$ sudo aptitude install qsynaptics
```

Configuring the touchpad with qsynaptics

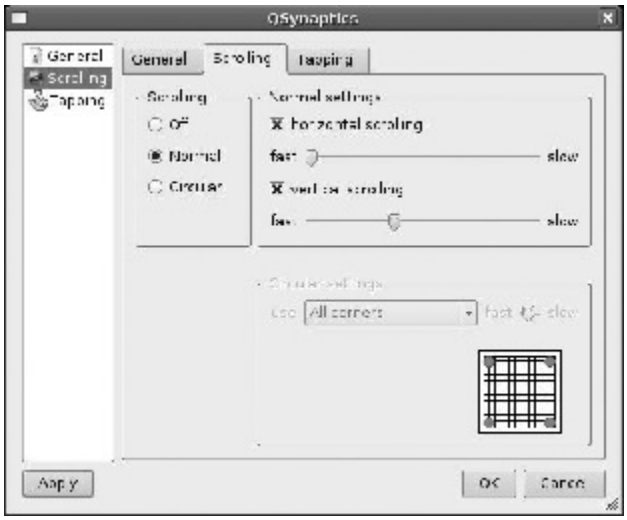
Qsynaptics is a graphical program, so it can be run from within the GNOME desktop. To start it, launch a terminal and run `qsynaptics`. The Qsynaptics dialog will appear (see Figure 5-3). The General tab lets you enable or disable the extended features of the Synaptics touchpad driver.

Figure 5-3. Qsynaptics at launch



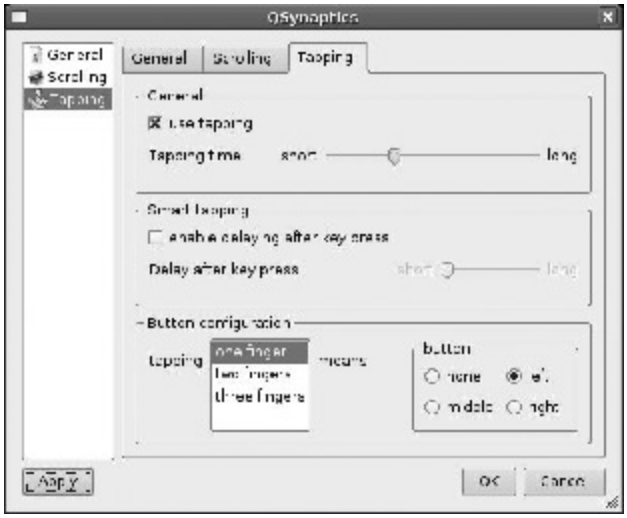
The Scrolling tab (see Figure 5-4) lets you adjust the scrolling behavior. You can adjust the sensitivity of the scroll area, enable horizontal scrolling, and even change the scrolling motion from an up-and-down swipe to a circular gesture on the touchpad.

Figure 5-4. Scrolling methods



Tapping gestures are configured on the Tapping tab, shown in Figure 5-5. You can disable tapping altogether; insert a tap-disabling delay after any keyboard press; and create tap gestures for one, two, or three finger taps.

Figure 5-5. Playing with tap settings



Once you spend some time adjusting your trackpad for your particular needs, you'll wonder how you lived without this flexibility. It's great to be able to turn on new features or disable features you may find annoying.

Hack 50. Connect Multiple Displays

You plug an external monitor into your laptop or a second monitor into your desktop computer. At best, nothing happens. At worst, something horrible happens. Find out how to get it right, whether you want to clone your display or spread it across multiple monitors.

Laptops give us the ability to go mobile but usually with a number of compromises compared to a more full-featured desktop machine. The most obvious compromise is screen size. The typical laptop screen is tiny compared to what you probably have sitting on your desk, but setting up your laptop to use an external monitor can give you the best of both worlds.

But multiple screens aren't limited to laptops. If you spend hours in front of a desktop computer every day, a second or even third monitor can help you work more efficiently.

All it takes to add a second monitor is an extra video card, or a video card with multiple outputs and a bit of configuration of Xorg.

First, you need to determine your video chipset, which in turn will determine which drivers you can use. If you don't know the chipset, you can open System→Administration→Device Manager, and then scroll down the devices on the left until you find a device labeled "AGP" or "video." Alternatively, you can open a terminal and type `lspci` to see a list of all PCI devices in your system: look for a line that says "VGA compatible controller" or similar. If your laptop has an nVidia or Radeon (ATI) video chipset, you have the option of using proprietary binary drivers made available by the manufacturer. These proprietary drivers offer more features and generally better performance than the currently available open source drivers, but many people specifically avoid them on idealistic grounds. Hopefully, we'll eventually see these proprietary drivers released under an open source licence, but in the meantime they do still offer some advantages if you don't mind using proprietary software.

Grok the Xorg Conf File

No matter which technique you use to configure your displays, you will need to make changes to the Xorg configuration file located at */etc/X11/xorg.conf*, so it helps to be familiar with the way it's structured and the terms that are used.

xorg.conf contains a series of stanzas, each of which defines a specific part of the X configuration. Various elements such as video cards, screens, and input devices are defined separately and then progressively bound together toward the end of the file. There are a multitude of additional stanzas in the Xorg configuration file, but the following ones are the important sections when running multiple displays (or *heads*).

Section "Device"

A *device* is a video card or video adaptor, so you will need one `Device` section for each card in your system. A video card with multiple outputs is considered a single device, but requires two `Device` sections to distinguish the outputs. Each `Device` section includes, at a minimum, entries for `Identifier` and `Driver`, but if you have more than one video card installed, you will also need to enter a `BusID` value so Xorg knows which section is associated with each card; if you have a card with multiple outputs, enter a `Screen` value so Xorg knows which section is associated with which output. Use `lspci -X` to determine the BusID for each card in a format suitable for entry into *xorg.conf*:

```
Section "Device"
  Identifier "NVIDIA Corporation NV34M [GeForce FX Go5200]"
```

```
  Driver      "nvidia"
  BusID       "PCI:1:0:0"
EndSection
```

If you have a single device with multiple outputs, just number the Screen entries starting from 0. You should also give each `Identifier` a unique name (in this example, I've simply appended 0 or 1 to the end):

```
Section "Device"
        Identifier      "NVIDIA Corporation NV40 [GeForce 6800] 0"
        Driver          "nvidia"
        BusID           "PCI:3:0:0"
        Option          "RenderAccel"           "true"
        Option          "AllowGLXWithComposite" "true"
        Screen          0
EndSection

Section "Device"
        Identifier      "NVIDIA Corporation NV40 [GeForce 6800] 1"
        Driver          "nvidia"
        BusID           "PCI:3:0:0"
        Option          "RenderAccel"         "true"
        Option          "AllowGLXWithComposite" "true"
        Screen          1
EndSection
```

Section "Monitor"

You need one Monitor section per monitor you have attached to your computer. Give each one a sensible and unique `Identifier` value:

```
Section "Monitor"
        Identifier      "Main Monitor"
        Option          "DPMS"
        HorizSync       28-80
        VertRefresh     43-60
EndSection

Section "Monitor"
        Identifier      "Second Monitor"
        Option          "DPMS"
        HorizSync       28-80
        VertRefresh     43-60
EndSection
```

Section "Screen"

In X terminology, a *screen* is a combination of a device and a monitor. Screen sections have `Device` and `Monitor` values that refer to the appropriate sections by name:

```
Section "Screen"
        Identifier      "Main Screen"
        Device          "NVIDIA Corporation NV40 [GeForce 6800] 0"
```

```
        Monitor         "Main Monitor"
        DefaultDepth    24

        # Subsections for other display depths not shown...
        SubSection "Display"
                Depth           24
                Modes           "1600x1200"
        EndSubSection
EndSection

Section "Screen"
        Identifier      "Second Screen"
        Device          "NVIDIA Corporation NV40 [GeForce 6800] 1"
        Monitor         "Second Monitor"
        DefaultDepth    24
        SubSection "Display"
                Depth           24
                Modes           "1280x1024"
        EndSubSection
EndSection
```

Section "ServerLayout"

The `ServerLayout` section is in turn a combination of one or more `Screen` sections, along with `InputDevice` sections that define keyboards, mice, and trackpads. You'll see some examples of a `ServerLayout` section shortly.

Tracking Down X Errors

Once you start modifying the Xorg configuration file, there is a very high likelihood that you will end up with your computer in a state where X can't start at all. The first place to go looking for debugging information is the Xorg logfile, which is regenerated each time X starts and is stored in */var/log/Xorg.* `0` *.log* or similar. The `0` is the identifier that Xorg uses to identify the X session; if you run multiple X sessions you may also have an *Xorg.1.log*.

The Xorg log prepends a severity key to the start of each line, making it very easy to see exactly what caused the problem. `(II)` lines are informational only, `(WW)` lines are non-fatal warnings, and `(EE)` lines are fatal errors. Problems often cascade down through the logfile as one failure causes another, such as a failure to initialize a video card leading to a failure to initialize a screen.

Configure Xinerama

X can natively support multiple heads, but by default, each one is considered to be totally independent of the others. With independent heads, it's not possible to have windows span multiple screens, drag windows between screens, or to move the mouse cursor from one to the other by dragging it off the edge of one screen and onto the other. Each head is effectively a totally separate environment.

Xinerama is a feature of X that overcomes this limitation by allowing multiple screens to be associated with each other and bound together into a single, unified "virtual screen." You can define the relationship of the screens to each other so your computer knows where each screen is physically located relative to the others, allowing you to drag windows around and have them overlap the screens in a way that makes sense given the position of your monitors on your desk.

First, make sure Xorg has Xinerama support activated by adding an appropriate `ServerFlags` section if one doesn't exist already:

```
Section "ServerFlags"
   Option "Xinerama" "true"
EndSection
```

Then add sections for your second device, monitor, and screen as described earlier, giving them descriptive names. Finally, add your second screen into the ServerLayout stanza, describing its relationship to the primary screen. Here's an example with two screens:

```
Section "ServerLayout"
        Identifier      "Twin Head"
        Screen          "Main Screen"
        Screen          "Second Screen" RightOf "Main Screen"
        InputDevice     "Generic Keyboard"
        InputDevice     "Configured Mouse"
EndSection
```

Exit GNOME, press Ctrl-Alt-Backspace to kill off Xorg and force it to respawn, and watch as both monitors flash to life.

Configure TwinView on nVidia

nVidia cards with multiple outputs support a special video mode called TwinView using the proprietary (binary) drivers. TwinView is a bit like Xinerama because it binds multiple displays together into a single virtual display, but the difference is that it's done at a very low level within the video driver and the hardware itself. Both heads share a single framebuffer, which basically tricks Xorg into thinking you only have a single monitor, giving it much better performance than Xinerama, as well as allowing OpenGL to operate across both heads at full speed. As far as Xorg is concerned, it's only driving one screen: it just happens to be a screen with very odd dimensions.

By default, Ubuntu installs the open source `nv` driver, but to use TwinView, you will need to install the proprietary *nvidia* driver instead. The *nvidia* driver is part of the *linux-restricted-modules-[arch]* package, which comes in several different versions depending on what kernel you are running. Full instructions for installing this binary driver are in "Enable 3-D Video Acceleration" [Hack #52].

Once you have Xorg working successfully with a single monitor and the binary drivers, run the command `sudo nvidia-xconfig -twinview` to modify your *xorg.conf* to support TwinView. If you want to tweak anything (or see what was done), open */etc/X11/xorg.conf* and go to the Device section for your nVidia card. Rather than defining the second monitor separately, the nVidia driver accepts a number of additional options to enable TwinView mode and configure the monitor directly:

```
Section "Device"
    Identifier     "NVIDIA Corporation NV40 [GeForce 6800]"
    Driver         "nvidia"
EndSection

Section "Screen"
    Identifier     "Default Screen"
    Device         "NVIDIA Corporation NV40 [GeForce 6800]"
    Monitor        "Generic Monitor"
    DefaultDepth    24
    Option         "RenderAccel" "true"
    Option         "AllowGLXWithComposite" "true"
    Option         "TwinView" "True"
    Option         "TwinViewOrientation" "RightOf"
    Option         "UseEdidFreqs" "True"
    Option         "MetaModes" "1280x1024,1280x1024; 1024x768,1024x768"
```

```
    # Subsections for other display depths not shown...
   SubSection     "Display"
        Depth     24
        Modes     "1600x1200"
    EndSubSection
EndSection
```

The `MetaModes` option tells the driver what resolutions to use for the two monitors, with the resolution reported back to Xorg being the sum of the dimensions. Multiple resolution pairs can be specified; in this case the monitor orientation is side by side (`LeftOf`), so the effective resolution is either 2560×1024 or 2048×768.

AU: Should the orientation specified in the previous sentence be “RightOf,” as indicated in the above code example?

You can also explicitly set offsets to accurately control the logical position of the screens:

```
Option "MetaModes" "1600x1200 +0+0, 1024x768 +1600+0;"
```

Other possible values for `TwinViewOrientation` are `RightOf`, `Above`, `Below`, and `Clone`. `Clone` is a special mode that displays the same image on both monitors, ideal for running presentations on a laptop connected to an external projector.

If above change is made, change “RightOf” in prev. paragraph to “LeftOf”

Configuring MergedFB on ATI/Radeon

Recent ATI video cards support an Xorg mode called `MergedFB`, which is quite similar to nVidia’s TwinView in that it manages connection to the second monitor directly within the driver and pretends to be a single large virtual monitor:

```
Section "Device"
   Identifier "Radeon7000"
   Driver     "radeon"
   BusID      "PCI:0:9:0"
   Option     "MergedFB" "true"
   Option     "MonitorLayout" "AUTO, AUTO"
   Option     "CRT2Hsync" "30-65"
   Option     "CRT2VRefresh" "50-75"
   Option     "CRT2Position" "LeftOf"
   Option     "MetaModes" "1280x1024-1280x1024"
EndSection
```

Like TwinView, there are options to configure the frequency, position, and resolution of the second monitor.

Combining Methods

Sometimes even two monitors aren’t enough, but unfortunately video cards with more than two outputs are quite rare, so you may need to combine techniques if you want to run three or more heads. For example, you may have a twin-head nVidia card running two heads using TwinView, plus a third head connected to an older PCI video card. Because TwinView presents itself to X as being a single virtual screen, you can then combine it with the third head using Xinerama to achieve a triple-head system with relatively little effort.

With a bit of creativity, it’s even possible to combine TwinView, MergedFB, and Xinerama all on the same machine (see Figure 5-6).

Figure 5-6. Quintuple-head desktop

Hack 51. Change the Ubuntu Splash Screen

Want to change what you see while Ubuntu boots up? Learn how to put your own image on that splash screen.

When Ubuntu starts up, it displays a progress bar, the Ubuntu logo, and scrolling messages showing what service is being started. Usplash, the userspace bootsplash system, allows you to customize the background, text, and progress bar of the splash screen.

To customize your splash screen, install GCC and the BOGL framebuffer library development package:

```
$ sudo apt-get install gcc libbogl-dev
```

Then create a PNG file to use as the splash-screen background. This is trickier than it sounds because there are some very severe restrictions on the format of the image: it must be 640×480 pixels in size and only 16 colors, so forget about using a nice photograph! Some of the colors are used for special purposes such as to draw the text messages that appear onscreen as services are started so the usable palette is even more restricted.

The special palette entries are shown in Table 5-1.

Table Special palette entries

Index	Use
0	Main and text backgrounds
1	Progress bar color
2	“OK” text color
4	Progress bar background color
8	Description text color

Index	Use
13	“Failed” text color

Make a directory to work in, copy your PNG in, and rename it to something like *usplash-mine.png*:

```
$ mkdir usplash
$ cp myimage.png usplash/usplash-mine.png
```

Enter the working directory and convert the PNG to a hexadecimal source file; then “compile” it into a shared object file:

```
$ pngtobogl usplash-mine.png > usplash-mine.c
$ gcc -Os -g -I/usr/include/bogl -fPIC -c usplash-mine.c -o usplash-mine.o
$ gcc -shared -Wl,-soname,usplash-mine.so usplash-mine.o -o usplash-mine.so
```

AUs: Need “sudo” prepended to previous 2 blocks of commands?

In this case, the resulting object file is called *usplash-mine.so*, but you can give it a different name, as long as you don’t call it *usplash-artwork.so*. That name is used by the system to find the current bootsplash artwork, so the safest thing to do is give all the object files unique names and reference the current file with a symlink:

```
$ sudo cp usplash-mine.so /usr/lib/usplash/usplash-mine.so
$ sudo ln -sf /usr/lib/usplash/usplash-mine.so \\
              /usr/lib/usplash/usplash-artwork.so
```

To give the kernel access to the splash image early in the boot process, it needs to be stored in the *initramfs* (initial RAM filesystem). The *initramfs* is built when the kernel package is installed, so force the currently installed kernel package to be reconfigured:

```
$ sudo dpkg-reconfigure linux-image-\Quname -r\Q
```

Once they have been created and installed in */usr/lib/usplash*, you can switch between various splash screens, including the original Ubuntu splash, simply by updating the symlink and regenerating the *initramfs*. For example, to return to the original splash screen:

```
$ sudo ln -sf /usr/lib/usplash/usplash-default.so \\
              /usr/lib/usplash/usplash-artwork.so
$ sudo dpkg-reconfigure linux-image-\Quname -r\Q
```

Hack 52. Enable 3-D Video Acceleration

After a fresh install, X11 is probably not configured to take full advantage of your graphics card. Learn how to take advantage of proprietary drivers to configure your video card for top speed.

Modern graphics cards are actually quite powerful little computers in their own right. Squeezing maximum performance out of them requires them to be set up the right way.

Measure Performance

To see how well your system is currently performing and give yourself a baseline to compare against after making changes, you can use a game such as Unreal Tournament 2004 or Quake 3 to measure graphic-card performance, both of which have "timedemo" modes that run through a defined sequence of events as fast as possible and report the average framerate achieved. An alternative is to use the *glxgears* utility (run `glxgears -printfps`), which is included with the *mesa-utils* package, but it's such a simple application that it really doesn't stress any decent 3-D hardware, so the framerate figures it generates aren't much use as a benchmark. If you get anything less than about 2000 fps in *glxgears*, your video card could probably do with a tune-up—or replacement!

Identify Your Video Card

To get your video card running at maximum speed, you need to know what brand and model it is so you can select the best drivers. Video cards are generally installed as PCI, PCI-X, or AGP cards, or are provided directly on the motherboard. In reality, all of these card types are just variations on the basic PCI technology, and even onboard video cards are essentially just PCI devices that happen to be permanently installed on the motherboard. This makes it very easy to identify your card by scanning the PCI bus using *lspci*:

```
$ lspci -X
```

The `-X` flag tells *lspci* to report device identifiers in the format used by the X Windows configuration files rather than in native format.

You can also tell lspci to report in "verbose" mode:

```
$ lspci -v
```

Verbose mode provides you a bit more information about each device. Unfortunately, the `-X` and `-v` flags can't be combined, so you have to run them separately.

Look in the reported device list for an entry that says "VGA compatible controller." The device string will include the manufacturer and model. The most common brands of high-performance 3-D card are nVidia, ATI, and Matrox, although some embedded video cards use chipsets from Intel or other manufacturers.

Drivers

Drivers for video cards have been something of a sore point for open source developers over the last couple of years. Modern video cards are essentially very small, very high-performance computers with their own CPUs and memory, and the drivers are an essential part of what makes them work. The manufacturers tend to guard their driver technology greedily and release only precompiled drivers in binary format that may be used but not modified or reverse-engineered. This runs counter to the ideals of many open source developers who insist that all software, including drivers, should be available in source code format. Thus, many high-performance video cards have at least two sets of drivers

available: binary drivers provided by the manufacturer and open source drivers developed by third parties who examine the cards and try to write drivers that will work with them.

Binary drivers have the advantage that they generally offer higher performance and make use of all the features of the card, but they cannot be modified or examined by other developers and can't be compiled to run on different hardware architectures not supported by the manufacturer.

Open source drivers have the advantage of running on more types of hardware because developers can modify them to suit their own requirements, but the disadvantage is that the developers don't have access to internal information about the video-card hardware and may not be able to achieve high performance or make use of all features of the card.

The Ubuntu commitment to ship only free software means binary drivers aren't available in the base distribution, but if you want maximum performance from your 3-D card, you will need to install the binary drivers from the *universe* and *multiverse* repositories, or directly from the card manufacturer's web site. To enable *universe* and *multiverse*, you may need to "Modify the List of Package Repositories" [Hack #60].

nVidia

The open source driver for nVidia cards is called *nv* and comes as part of the basic Xorg installation in Dapper, but for maximum performance, install the binary *nvidia* driver which is part of the *linux-restricted-modules* package. The *linux-restricted-modules* package comes in several different versions to suit the different kernels, so check which kernel version you are running:

```
$ uname -r
2.6.15-17-686
```

Then install the matching *linux-restricted-modules* package:

```
$ sudo apt-get install linux-restricted-modules-2.6.15-17-686
```

You will also need either the *nvidia-glx* or the *nvidia-glx-legacy* package, depending on the specific card model. If your card is an older model —such as the TNT, TNT2, TNT Ultra, GeForce, or GeForce2—install the *nvidia-glx-legacy* package. If it's a newer model—such as a 4-series, 6-series, nForce, Quadro, or FX-series—install *nvidia-glx*:

```
$ sudo apt-get install nvidia-glx
```

Edit */etc/X11/xorg.conf* and look for a line that says something like:

```
Driver          "nv"
```

Edit it to specify the *nvidia* driver:

```
Driver          "nvidia"
```

Exit from your GNOME session; then when you are back at the login prompt, press Ctrl-Alt-Backspace to kill off the X server and force it to restart. As X restarts, you will probably see a big nVidia logo flash up on the screen, showing that it has loaded the binary *nvidia* driver rather than the default `nv` driver. Log back in to GNOME, start up *glxgears*, and see if your framerate has improved.

ATI

If you have an ATI card that was auto-configured by Ubuntu, it's probably running either the *ati* or *radeon* driver. For maximum performance with cards in the "Radeon" series, 9- series, X-series, or TV-Out–capable ATI cards, you can install the *fglrx* driver, and if your card is an 8500 or better, you can also install ATI's proprietary driver.

Installing the *fglrx* driver is very easy:

```
$ sudo apt-get install xorg-driver-fglrx
```

Then edit */etc/X11/xorg.conf*, look for a line that lists either the *ati* or *radeon* driver:

```
Driver          "ati"
```

and replace it with the *fglrx* driver:

```
Driver          "fglrx"
```

Installing ATI's proprietary driver requires a few more steps but may produce better performance on the latest cards. Start by going to http://support.ati.com and navigating to the Linux drivers. Download the ATI Driver Installer, not the specific Xorg driver—because that's packaged as an RPM, and you need to build a *.deb* instead. Once the (very large!) driver file has downloaded, install some supporting packages and configure the driver. It helps keep things neat if you do this in a subdirectory:

```
$ sudo apt-get install fakeroot gcc-3.4 module-assistant \\
             build-essential debhelper
$ mkdir ATI
$ mv ati-driver-installer-8.22.5-i386.run ATI/; cd ATI
$ chmod +x ati-driver-installer-8.22.5-i386.run
$ fakeroot ./ati-driver-installer-8.22.5-i386.run
```

The installer will ask you a couple of questions: answer "Generate distribution specific packages," then "Ubuntu," and then "Dapper." Once the installer finishes, it will have created a nice little Debian package that you can install:

```
$ sudo dpkg -i *.deb
$ sudo module-assistant build,install fglrx-kernel
```

Edit */etc/X11/xorg.conf* to use the `fglrx` driver as described earlier, reboot, and you're done.

Matrox

Matrox also make available a proprietary driver that you can build as a Linux kernel module. First, install the kernel headers to use when building the new module:

```
$ sudo apt-get install linux header \Quname -r\Q
```

Then download the latest driver installer from Matrox at http://www.matrox.com/mga/support/drivers/latest/.

Building the driver must be done with GCC 3.4, so install it, set an environment variable that specifies that it should be used in preference to other installed versions, and run the installer:

```
$ sudo apt-get install gcc-3.4 gcc-3.4-base
$ export CC=gcc-3.4
$ sudo sh mtxdriver-x86_32-1.4.3.3.run
```

Back up your existing *libGL* installation, since it won't work with the *mtx* driver:

```
$ sudo mkdir /usr/lib/libGL.back
$ sudo mv /usr/lib/libGL.so* /usr/lib/libGL.back/
```

Then edit */etc/X11/xorg.conf*, find the `Device` section for your video card and alter the `Driver` line to use the *mtx* driver:

```
Driver          "mtx"
```

Finally, reboot your computer so the kernel can load the new module.

Hack 53. Make Your Fonts Pretty

Ubuntu configures GNOME and KDE to use some very pretty fonts, but there are some tweaks you can use to make them even nicer.

Most computer users don't even think about fonts. They just expect them to work and assume that text will look the same whether it's viewed onscreen, printed, or sent to another user in a document. However, font management is actually a surprisingly complex task due to the many subtle variations in the ways fonts can be created and used.

Fonts Are Not Created Equal

Fonts can be defined in a number of different ways and have a variety of file formats. Each operating system has its own method of managing and displaying them. Some fonts are designed as bitmaps to be displayed on screen while others are in vector format so they can scale up or down and be printed at high resolution. Some come as bundles that include both bitmap and vector formats in the same package, with one used for onscreen display and the other used in printing or to generate files in output formats such as PDF. And some come as *families*, with several variations such as bold and italic bundled together with the base font, providing much better results than working from a single base font and then applying such variations algorithmically.

Font Management With Defoma

Ubuntu uses Defoma, the "Debian Font Manager," to centralize and simplify font management across all applications. Applications can vary dramatically in how they manage fonts, so when a new font is installed on your computer it's not always obvious how to tell each application that the font exists and where to find it.

Defoma gets around this problem by allowing applications to register themselves by providing a Defoma configuration script. Then when a new font is installed, Defoma works through all the configuration scripts and performs whatever action is necessary to enable the font for each application.

The first thing you should do then is make sure that your system is configured to use Defoma to manage fonts. Run:

```
$ sudo dpkg-reconfigure defoma
```

If Defoma is not currently set to manage fonts, you will be asked if you want to use it; answer Yes.

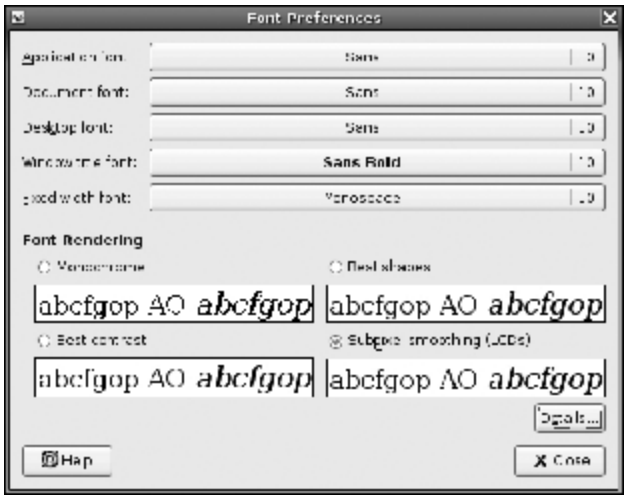
If your system has ended up in an unclean state with some manually installed fonts or applications that can't see some fonts, you can force Defoma to totally rebuild its configuration. This process rescans all your installed fonts and makes sure all registered applications have been updated to use them:

```
$ sudo defoma-reconfigure
```

Onscreen Font Rendering Preferences

Various displays have different characteristics, and what looks good on a CRT doesn't necessarily look good on an LCD. Ubuntu provides a number of font options through System→Preferences→Font (see Figure 5-7).

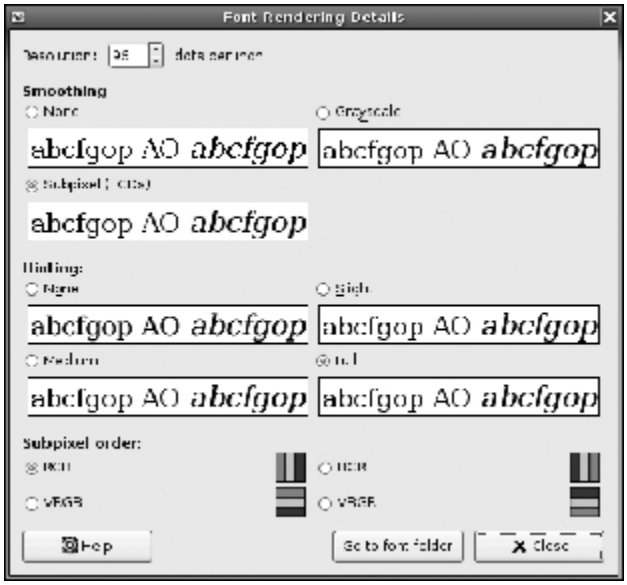
Figure 5-7. GNOME Font Preferences



You can change the default system fonts to suit your preferences, but if you have an LCD, the item to pay attention to is the subpixel smoothing option under Font Rendering. Each pixel in an LCD consists of three subpixels, one each for red, green, and blue. Subpixel smoothing takes the physical layout of the subpixels into account to display fonts as smoothly as possible.

Advanced options (shown in Figure 5-8) are accessible through the Details button near the bottom right.

Figure 5-8. Advanced font preferences



From here you can specify screen resolution, smoothing, hinting, and subpixel order.

Screen resolution

When the font renderer displays text onscreen, it needs to convert between various units to determine how large the text needs to be. Often font sizes are specified as *points*, which is a unit of measure used (rather inconsistently!) for hundreds of years by printers. Nowadays, most people agree on one point being equal to $^{1}/_{72}$nd of an inch, but if you tell your computer to display, for example, 18 point text, it needs to know the resolution of your display so it can figure out how many pixels are equivalent to $^{18}/_{72}$nds (i.e., $^{1}/_{4}$) of an inch on your particular screen.

Screen resolution is usually expressed as dpi, or dots per inch. To figure out the horizontal and vertical resolution of your screen, measure its width and height and then divide those values into the pixel dimensions set in System→Preferences→Screen Resolution. For example, a typical so-called 17-inch LCD will have physical dimensions of about 13.3 inches by 10.75 inches and run at a native resolution of 1280×1024 pixels. That gives a horizontal resolution of 1280 ÷ 13.3 = 96.2 dpi, and a vertical resolution of 1024 ÷ 10.75 = 95.3 DPI. Close enough to call it 96 dpi for both.

By determining the actual physical resolution of your display and setting the correct Resolution value in the Font preferences, you can ensure that when your computer displays a font onscreen at a specific size it will be scaled to appear at that actual size.

Smoothing

The Smoothing setting actually controls the level of antialiasing to apply when fonts are rendered. Antialiasing can have a dramatic impact on the clarity of fonts, particularly when displayed on an LCD. It smooths out jaggy corners and edges on fine lines by visually filling in gaps using surrounding pixels set to intermediate shades of grey. If you have an LCD, for the best-looking fonts, you should definitely select Subpixel as the Smoothing setting.

Hinting

Because computer screens operate at a much lower resolution than what we are used to seeing with printed material, fonts that are scaled down to a small size can sometimes suffer from effects whereby the shape and position of individual letters don't interact well with the pixel structure of the display itself, producing visible artifacts. For example, two letters next to each other that both have thin vertical lines may happen to fall slightly differently onto the pixel grid of the display, with the result that one line appears fatter than the other. A similar effect can occur with rounded letters, where fine curves may disappear or be inconsistent. Often the relative placement of letters will alter the visual effect of other letters around them. Hinting is the process of making tiny adjustments in the outline-filling process while rendering fonts to compensate for effects that might cause individual characters to appear differently from the way they were designed.

Doing accurate hinting requires more processor power whenever your computer needs to render large quantities of text, but the end result is text that appears smoother, more consistent and easier to read. You can choose from four hinting levels in Font Rendering Details: None, Slight, Medium, and Full. The difference might seem subtle if you're not used to closely examining text and you don't know what to look for, but if you have a relatively modern machine, it's worth turning hinting on. LCDs in particular can benefit greatly from it, giving you much more readable text and less eyestrain.

Subpixel order

In the main Font Preferences dialog, there was an option to turn on subpixel smoothing, but for it to be really effective, you also need to make sure your computer knows the physical structure of the individual subpixels. In reality, subpixels are not dots: they're typically very short lines placed side by side.

The vast majority of LCDs use an RGB order, but some reverse that and place the subpixels in BGR order. Then there are variations on those two options, with some manufacturers stacking subpixels vertically instead of placing them side by side. Selecting the option that matches your particular monitor structure will let your computer do the best job possible of smoothing fonts onscreen.

Install Microsoft Core Fonts

Microsoft Windows comes bundled with a number of core TrueType fonts. Because Windows is so widely used, many documents and web sites are designed around the core Microsoft fonts, and if you don't have them installed, your computer may not be able to display some documents as the author intended. Licence restrictions prevent the Microsoft fonts being distributed directly as part of Ubuntu, but Microsoft does make them available for free download directly from their web site, and there is even an Ubuntu package which takes care of downloading and installing them for you:

```
$ sudo apt-get install msttcorefonts
```

The *msttcorefonts* package is part of the *multiverse* repository, so it's not available on a standard Ubuntu installation and you may need to "Modify the List of Package Repositories" [Hack #60] before you can install it.

The package doesn't include the fonts themselves but instead connects to the Microsoft web site and downloads and installs them in the correct location on your computer. The fonts will then be available to applications the next time they start up.

Install Macintosh and Windows TrueType Fonts

Installing TrueType fonts is very easy on Ubuntu. On your desktop or in a file-browser window, just type Ctrl-L to access the Open Location window; then type in `fonts:///` and click Open. You will then see a list of all the fonts you currently have access to on your system. Drag your new TrueType font from your desktop or file manager into the font-list window, and it will be automatically installed and made available to applications through Defoma next time they start up.

It's actually not quite that simple if the fonts come from a Macintosh system, because Mac OS embeds extra font information using a special format that Linux can't read directly. Before you drag Mac OS fonts into your *fonts:///* folder, you need to convert them with a utility called *fondu*, which you can install with the following command:

```
$ sudo apt-get install fondu
```

Then copy your Mac OS font directory to your Linux machine and run:

```
$ fondu *
```

inside it to generate converted TTF files.

The *fonts:///* location isn't a real location in the filesystem. It's a virtual view that lets you manage the fonts that have been installed without having to worry about where they are actually located on disk. The fonts shown by default are the system-wide fonts that have been installed on your machine for all users to access, but when you drag a new font into the window, it actually stores it inside a hidden folder called *.fonts* inside your home directory.

Chapter 6. Package Management

One of Ubuntu's key strengths is the quality of the software packages that are included in the main distribution: they have been relentlessly tested, and come together to form a rock-solid Linux distribution. These packages are managed by a tool called APT (Advanced Packaging Tool) as well as a variety of frontends that make *apt* a bit easier to work with. In this chapter, you'll learn how to work with apt and those frontends.

Although the main Ubuntu distribution is integrated and very solid, that stability comes at a price: many of the optional packages you may want have been excluded from the main distribution. However, there are many ways to bring in optional packages, and there are hacks in this chapter to help you do so. These hacks will also help you understand the consequences of adding optional packages. You'll also learn how to compile applications from source, create your own Ubuntu packages, and host packages in a custom repository for others to use.

Hack 54. Manage Packages from the Command Line

Dive under the hood and manage packages directly from the command line.

Ubuntu provides a number of very nice graphical tools for managing software packages from the desktop, but sometimes you just have to get hands-on to really get things done. Servers don't generally have a graphical desktop environment installed, and if you manage machines remotely through a shell session, you need to know how to use Ubuntu's command-line package-management tools.

Ubuntu strives for consistency in the process of managing software. All software is packaged using a strictly defined format that contains the program itself plus information about how it needs to be installed, and all packages are stored on your computer in the exact same way. The package-management tools are layered, with each layer building on the levels below. At the lowest level is *dpkg*, which directly manages packages; mid-level tools like *apt* sit on *dpkg* and provide more functionality, such as automatic dependency resolution; and high-level tools like Synaptic and Adept sit on *apt* and let you graphically browse package lists and do simple point-and-click installation.

dpkg

dpkg is the basis of the Debian package-management system and allows direct manipulation of packages. If you have a local package on disk called *program-1.0-1.deb* that you want to install, *dpkg* is the tool to use. Because it's such an important part of the package-management stack, it has a whole hack of its own [Hack #57].

apt And Friends

While it was not originally intended as a frontend tool, but rather as an intermediate layer between *dpkg* and end-user tools such as Synaptic and Adept, running *apt* directly is probably the most common method for managing packages on the command line.

Some commands require root privileges so you need to prepend them with *sudo* if you're running as an unprivileged user.

Here are some commands you'll wonder how you ever lived without:

Retrieve the current list of packages from all servers

If you don't do this from time to time your local list of available packages may become out of date. Run this command occasionally before doing a *dist-upgrade* or searching for a new package. The package lists are large, and doing an update may result in several MB of data being retrieved from the Internet:

```
$ sudo apt-get update
```

Do a keyword search

The `search` command searches through the list of available packages, including package names and descriptions. You can put in several keywords—for example, `apt-cache search text editor` to find a list of text editors:

```
$ apt-cache search &lt;keywords&gt;
```

Get more information

Once you've found a package that looks interesting using `apt-cache search`, you can display more detailed information about it using `apt-cache show program`. This will tell you things like the size of the package (important if you are installing it via the Internet) and an extended description, as well as what other packages it depends on in order to work and the name of the developer who maintains the package:

```
$ apt-cache show
                        program
```

Install a package

This will get the latest version of the specified package and install it, along with any other packages that it depends on in order to work. If the requested package is already installed, this will upgrade it to the latest available version:

```
$ sudo apt-get install
                        program
```

Remove a program

If you've previously installed a program and decide you don't want it anymore, you can remove it using this command. Because some software packages can depend on others, removing one program may break other programs. Running `apt-get remove` therefore checks first to see if any other software programs need the program to work, and offers to uninstall them as well. This is just one example of the way the Ubuntu package-management tools have been designed to try to keep your computer in a sane state, without broken or half-installed software. It's certainly possible to break an Ubuntu system, but generally you have to try to do it. The *remove* command also has a `--purge` option that removes not just the program itself but also all configuration files associated with it:

```
$ sudo apt-get remove
                        program
$ sudo apt-get remove --purge
                        program
```

Upgrade your system

Over time, most of the software packages on your computer will become out of date, as new versions are released to add features or fix bugs. You could manually do `apt-get install foo` on each one, but that's not very convenient, so *apt* provides a simple way to upgrade your entire system at once. Just type `apt-get upgrade` to have *apt* check every single package on your system for a new version and then download and install it. This command will never install new packages; it will only upgrade packages that are already installed:

```
$ sudo apt-get upgrade
```

Perform a full upgrade

Sometimes, you'll have a software package installed, and a new version will come out that has a lot of new features and therefore now depends on some other program to run. For example, you may have a movie player installed that supports a lot of different movie formats. When new formats come out, modules for those formats may be added in separate packages, so the latest version of the movie-player software now depends on a new package that you don't yet have installed on your system. If you just run `apt-get upgrade`, you'll get the latest movie player, but you won't get all the new movie-format packages. The `apt-get dist-upgrade` command solves that problem for you: not only does it get the latest version of every package already installed just like `apt-get upgrade`, it also installs any new packages they need that may not be on your system yet. If you want to keep your system up-to-date with all the latest updates and security patches, running `apt-get update; apt-get dist-upgrade` from time to time is the best way to do it:

```
$ sudo apt-get dist-upgrade
```

Clean up

When you ask *apt* to install a software package, it downloads the package and stores it in a cache on your disk before it does the actual installation. If you then remove the package, but later change your mind again and reinstall it, *apt* doesn't need to fetch it from the Internet again because the package is sitting in the local cache. That's great for saving bandwidth, but after a while, the cache can use up space on your disk, so it's a good idea to periodically delete old packages from it. Running `apt-get clean` will totally flush the package cache, possibly freeing up some precious disk space. This command is quite safe because the worst that can happen is *apt* may need to download a package again if you remove it and then reinstall it:

```
$ sudo apt-get clean
```

Clean smarter

`autoclean` is almost the same as `clean`, except it's just a little bit smarter: instead of cleaning out your entire package cache, it deletes only superseded packages. For example, your package cache may contain packages for the last four versions of a text editor that has been upgraded a number of times. Running `apt-get autoclean` will delete the oldest three versions from the cache, leaving only the latest one. That makes sense because you're not likely to reinstall anything except the latest version anyway:

```
$ sudo apt-get autoclean
```

Lazy like a fox

If you spend a lot of time working on the command line, you can make things easier for yourself by creating shortcuts for the common commands. Add these lines to your *~/.bashrc*:

```
alias agi='sudo apt-get install'
alias agu='sudo apt-get update'
alias ags='apt-cache search'
alias agsh='apt-cache show'
alias agr='sudo apt-get remove'
alias agd='sudo apt-get dist-upgrade'
```

Then you won't need to type the whole command. To do a search, for example, just type `ags` *`foo`* and enter your password if required. Laziness is a virtue!

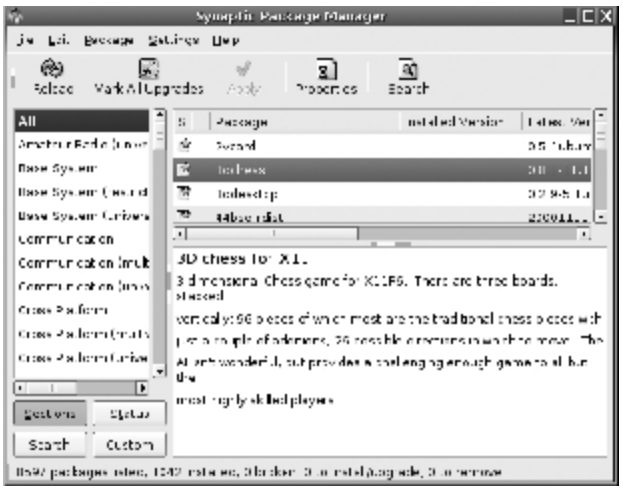
Hack 55. Manage Packages with Synaptic

Use the default Ubuntu package manager to easily install and update programs on your system.

One thing that can be difficult to get used to if you are new to Linux is *package management*. On one hand, the way that Ubuntu handles program installation and upgrading is much simpler than on other operating systems. On the other hand, it's rather different from what you might be used to, so it takes some time to learn the ins and outs. Luckily, all you need to learn is a single tool and you can then apply that knowledge to install and upgrade any of thousands of Ubuntu programs. In this hack, I talk about how to use the default Ubuntu package manager called Synaptic to manage all of Ubuntu's packages efficiently and easily.

The first step is to start Synaptic. Click System→Administration→Synaptic Package Manager to launch the program. Synaptic requires *root* privileges to run, so you will be prompted for your password when it starts. Figure 6-1 shows the default interface, which has a lot of information on it, but once you are familiar with its functions, you'll find it's not too difficult to navigate.

Figure 6-1. The default Synaptic window



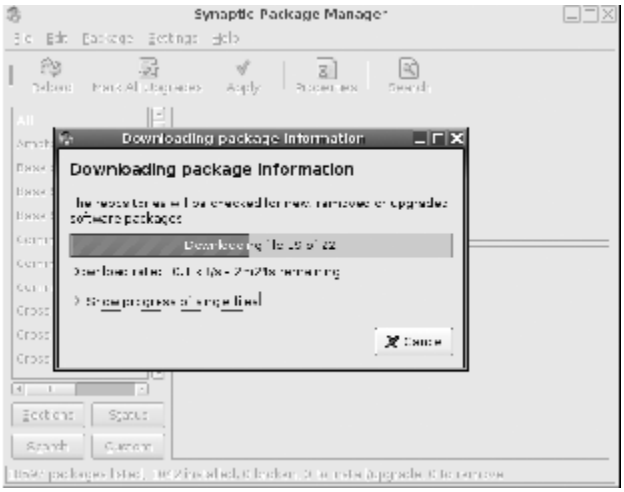
The main window is split up into a few sections. The sidebar displays the different package categories so that you can quickly drill down to a particular group of packages (like games) and not have to read through packages in other categories. The first category, All, display all packages. On the right side of the window along the top is the list of packages, along with information about whether they are installed and what the latest version is. If you select a particular package, detailed information is displayed in the pane below. Near the top of the screen are a few buttons that correspond to the main actions you'll want to perform when using Synaptic, described next.

Reload the Package List

In Ubuntu, packages reside in what are known as *package repositories*. "Modify the List of Package Repositories" [Hack #60] goes into more detail about the different package repositories that exist for Ubuntu, but in short, these repositories allow different types of packages to be kept together in a central place. As a user, this means that you don't need to search around on the Internet to find a program; you just find the program in the package manager, and it retrieves that program's package, as well as any other packages that program needs from the appropriate package repositories.

There are often updates to these package repositories as new versions of software are released, so before you do anything else, you will want to get the very latest list of packages and their versions. To do so, click the Reload button at the top of the window. Synaptic will download and update its local database of packages with these new versions (Figure 6-2). Once it is finished, you are ready to install and update packages.

Figure 6-2. Click Reload to grab the latest list of packages



Install Packages

To install a package in Synaptic, find the particular package in the package list. You can either scroll down manually or, if you know the name of the package, you can start typing in that list to highlight it. If you aren't sure about the category or the name of a particular program, you can also search within Synaptic for it. More information on how to use the Synaptic search feature can be found in "Search for Packages," later in this hack.

After you have identified the package you want to install, right-click on it and select "Mark for Installation." If you want to install more than one package, go through and mark them all for installation at this point. Synaptic will prompt you if a package depends on another package to install so that you can approve the installation of it. Once all of the packages you wish to install are selected, click the Apply button, and Synaptic will display a prompt letting you know which packages it will install along with how much space it will need. Then Synaptic will automatically download and install all of the programs onto the system.

Update Packages

One particularly nice feature of a centralized package repository is that it makes upgrading all of the programs on your system a breeze. To update your system, click the Mark All Upgrades button. Synaptic will go through and mark any and all packages that have an update available. Click Apply to apply the changes, and Synaptic will download and install all available updates for your system: from desktop programs to libraries to games. You will want to periodically update the programs on your system to make sure you have the latest bug fixes and security patches.

If you have a lot of packages to upgrade and you are on a slow connection, it can take some time to download everything. Don't worry if you need to shut down your computer and can't finish the download at the moment—the downloads are *resumable*, so the next time you start the upgrade, it will pick up where it left off. However, note that once Synaptic finishes downloading and starts upgrading the packages, you will want to let it complete the process; otherwise you may end up with half-installed packages, and it will take further work to complete the installation.

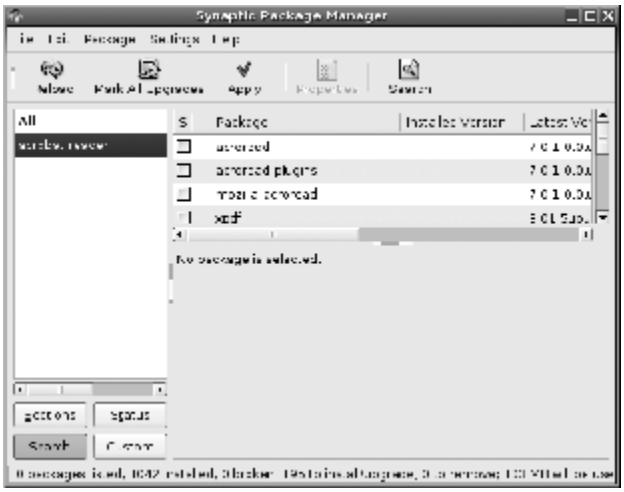
Remove Packages

You can remove packages in Synaptic in much the same way as you can install them. Find the package you want to remove in the package list, right-click, and select “Mark for Removal” if you want to remove the program but leave behind any configuration files you may have changed. You can also select “Mark for Complete Removal” if you want to purge even the configuration files from the system.

Search for Packages

Sometimes you know what sort of program you want to install, but you aren’t completely sure what the exact name of the package is. Synaptic includes a search tool to make it simple to find the packages you need. Click the Search button at the top of the window and enter any keywords you want to search for. Synaptic will then display a filtered package list that contains only the packages that match those keywords (Figure 6-3). You can then more easily locate and install them. To get back to the standard package list, click the Sections button at the bottom of the window.

Figure 6-3. The Synaptic search interface



Edit Preferences and Add Repositories

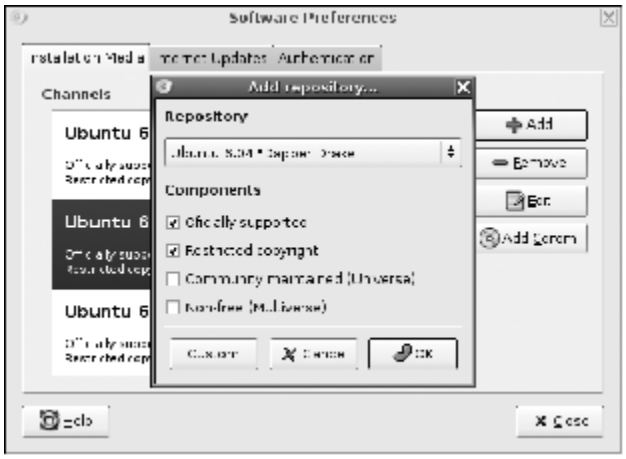
The above steps are enough to do standard package management, but as you get used to Synaptic you may want to tweak some of its settings. Click Settings→Preferences to see the main preferences window (Figure 6-4). In the Preferences window, you can configure some of Synaptic’s default behavior, such as how many undo operations to allow, what colors and fonts to use for packages, which columns to display in the main window, and the details of your network proxy settings.

Figure 6-4. Synaptic's Preferences window lets you tweak default behavior

You may also want to add extra software repositories to your system. Extra repositories such as universe and multiverse [Hack #60] contain a much larger set of software; however, the software is packaged and supported by the Ubuntu community instead of the Ubuntu project itself, so these packages may not be as well tested. To change your repository settings, click Settings→Repositories. Figure 6-5 shows the main window that displays all of the main repositories you have configured. To add subcategories to a particular repository, select the repository and click Add. To add *universe* or *multiverse* subcategories, click their corresponding checkboxes. If you are somewhat familiar with the *sources.list* layout, you can also click the Edit button to more directly configure the repository settings.

If you have changed repository settings, be sure to click Reload to get the latest package lists from the new repositories you have added.

Figure 6-5. Add to Synaptic's package repositories



Hack 56. Manage Packages With Adept

Adept is the KDE equivalent of Synaptic, allowing you to manage packages on a Kubuntu system.

Ubuntu's package-management tools are some of its greatest strengths. A huge amount of work has been put into the underlying management code in order to make it as flexible as possible, separating the user interface from the code that performs the actual heavy lifting. The result is that there are many different ways to interact with *apt*, allowing you to manage packages from the command line [Hack #54] in a variety of ways or use one of many graphical frontends that hides the details from you and provides a point-and-click GUI. This approach allows graphical frontends to be relatively lightweight: they don't need to include basic package-management logic because all of that is provided by *apt* itself, which in turn makes the GUI application simpler and more robust.

Adept is a graphical package-management interface that sits on the *libapt* API and uses the *Qt* libraries, making it an ideal alternative to Synaptic for Kubuntu systems. You can use it to install, upgrade, and remove packages, of course, but it also comes with a couple of little extra utilities to make your life easier: *adept-notifier* and *adept-updater*.

Basic Adept Usage

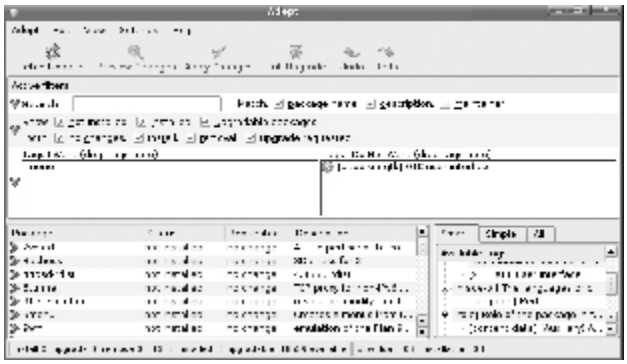
If you don't already have it installed, use the command line or an alternative package-management tool to install Adept:

```
$ sudo apt-get install adept libqt-perl
```

The reason for installing *libqt-perl* is that it enables some extra features in Adept, including the ability to configure and reconfigure packages directly.

Launch Adept from K menu→System→Adept. The main management window is quite similar to Synaptic but includes some interesting additional features, such as Debtags support, shown in Figure 6-6.

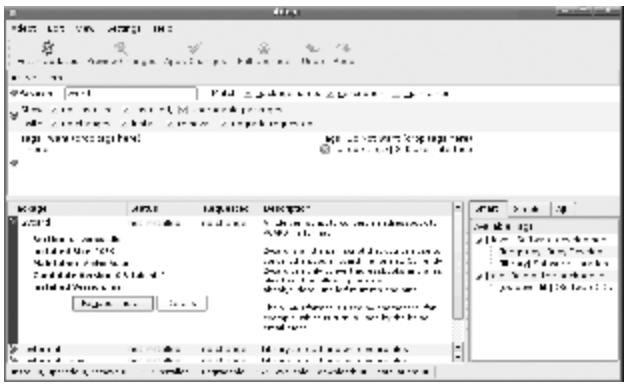
Figure 6-6. Package management with Adept



Debtags is an initiative to associate specific keywords with packages, rather than rely on searching package titles and descriptions to find what you want. It's a lot like the tags used for free-form taxonomy by Flickr and various blogging tools. You can browse through the currently available tags at the bottom right of the window, and drag them into the Tags I Want and Tags I Do Not Want boxes to restrict package search results. For example, if you don't want to see any GNOME packages listed in search results, just browse through the tags to the "GTK user interface" tag and drop it on Tags I Do Not Want.

The search system is dynamic, so you can just start typing search terms, and the list of packages will be narrowed automatically (see Figure 6-7). Then you can click the triangle next to a package name to see a description of the package and a Request Install button to install it. If the package is already installed, the button will say Request Removal instead.

Figure 6-7. Adept package search



Installation and removal of packages doesn't happen immediately. Instead, flags are set on the packages, and when you are happy with the changes you have made to package selections, you can click the Apply Changes button at the top of the window.

Adept also allows you to modify the list of software repositories used by your computer to fetch new packages. You can manually modify the list of package repositories directly [Hack #60], or go to View→Manage Repositories in Adept to see the repository list conveniently divided up into Type, URL, Distribution, and Components columns.

After making any changes to the repositories, click the Fetch Updates button at the top of the window to force Adept to fetch the latest package lists.

Adept also provides a convenient Full Upgrade button that finds and marks all packages on your system that have upgrades available, so a quick way to keep your entire system up-to-date is to open Adept, click Fetch Updates, then Full Upgrade, and then Apply Changes.

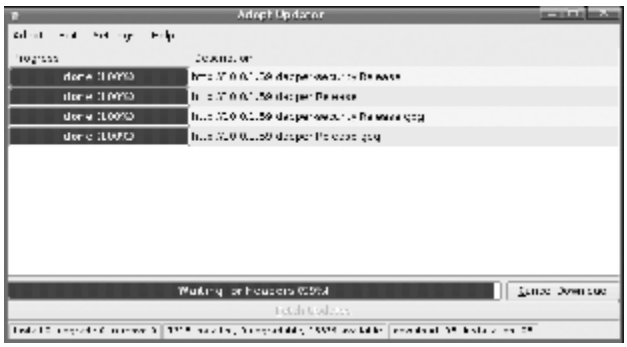
Receive Update Notifications Automatically

Updated software packages are released quite frequently, and if you want to stay on top of the latest security updates, it's important to know when updates are available. Instead of manually updating the package list every day and checking for new software, you can just use the *adept-notifier* panel applet to keep you informed.

adept-notifier sits in your system tray and regularly checks the package repositories for new releases. If there are no updates to install, it gives you a green light in the panel, but if new versions of any software you have installed become available, it pops up a notification message to let you know. That way you can get on with just using your computer and not worry about checking for updates manually.

Adept Updater is a convenient way to apply updates to all your currently installed packages. Launch it by going to K menu→System→Adept Updater, and then click the big Fetch Updates button at the bottom. You will see progress as the latest package lists are retrieved, as shown in Figure 6-8.

Figure 6-8. Adept Updater



If any packages need to be updated, you will then be given the option of downloading and applying those updates.

Hack 57. Install and Remove Standalone .deb Files

Use command-line tools to install individual .deb files when other automated tools aren't an option.

The package management for Debian-based distributions like Ubuntu is very powerful and saves a lot of effort that could be wasted finding the latest packages and tracking down dependencies. Automated tools like *apt-get*, Synaptic, and Adept should serve most users' needs almost all of the time, and you should stick to those tools whenever possible. However, there are some circumstances when you need to install a *.deb* package directly.

Ubuntu has automated tools for package installation for good reason. These tools provide you with a safety net that ensures packages stay compatible and have the libraries they require. If you instal standalone *.deb* files (especially ones that aren't packaged for your particular Ubuntu release), you not only lose a lot of these advantages, you might break parts of your system due to incompatible libraries, overwrite files other Ubuntu programs depend on, or add unique versions that make it more difficult to upgrade your system down the road. Before you install a standalone *.deb* package, especially if you are new to Ubuntu, please exhaust all other available resources, including the universe, multiverse, and metaverse Ubuntu respositories [Hack #60].

You have built your own kernel "the Ubuntu way"

If you have compiled your own kernel source using *make-kpkg*, you will end up with a *.deb* package for the kernel binaries, along with *.debs* for any extra modules you may have built (see "Build Kernels the Ubuntu Way" [Hack #78] for the specific steps in this procedure.)

You have compiled your own package from Ubuntu source

As when compiling a kernel, when you compile from Ubuntu source using the *dpkg* tools, you end up with a standalone *.deb* that you will need to install manually.

You want to roll back to an older version of a program

This circumstance might show up particularly if you are using a development release of Ubuntu. Sometimes the latest version of a package in a development release has bugs. In these circumstances, you might want to roll back to the previous version of a package. One of the simplest ways to do so is to track down the older *.deb* (possibly in your local package cache at */var/cache/apt/archives/*) and install it manually.

A program you want to install has a *.deb* but isn't in Ubuntu repositories

There might be some circumstances where a *.deb* you want to install doesn't appear in your Ubuntu repositories, possibly because it's a newer version than Ubuntu provides, because you have found a generic *.deb* file you want to install, or because a third party has provided a *.deb* for Ubuntu but hasn't created its own package repository (such as how the Opera web browser is currently packaged).

Nine times out of ten, if you find a program packaged as a *.deb*, there's a good chance Ubuntu has already packaged it in one of its repositories. Read "Modify the List of Package Repositories" [Hack #60] for more information about the extra repositories that are available from Ubuntu and third parties.

A program you have installed or upgraded won't fully install due to another package installing the same files

I have seen relatively rare circumstances where two packages provide the same file and when one is upgraded, it errors out with the message that it would overwrite files from another package. In these cases, you need to manually install that package using some of the `--force` options available to *dpkg*.

Install a .deb

Whatever the reason, when you find yourself with a *.deb* to install, it's time to turn to *dpkg*. *dpkg* is the tool that Debian-based distributions like Ubuntu use to install *.deb* files. (Even when you use an automated package-management tool, *dpkg* is used behind the scenes to actually install the packages to the system.) If you are familiar with the *rpm* tool for RPM-based distributions, you'll find *dpkg* has similar syntax. To install an ordinary *.deb* from the command line, type:

```
whiprush@ubuntu:~$ sudo dpkg -i
```

```
                packagename.deb
```

Replace *packagename.deb* with the *.deb* file you wish to install. If you have multiple files you want to install at the same time, you can either list them one after another on the command line:

```
whiprush@ubuntu:~$ sudo dpkg -i
                package1.deb package2.deb package3.deb
```

or use a file glob [Hack #13] to install all *.deb* files in the current directory:

```
whiprush@ubuntu:~$ sudo dpkg -i
            *.deb
```

dpkg also has a recursive option (`-R`). If you have a directory full of debs you want to install, type:

```
whiprush@ubuntu:~$ sudo dpkg -i -R
                /path/to/directory
```

and *dpkg* will recursively find and install all *.deb* files within that directory and its subdirectories.

Occasionally, when you install a package with *dpkg*, it might abort due to a number of reasons, including a package that is marked *hold*, a package that conflicts with another package, a package that depends on other packages that aren't installed, a situation in which installing the package would overwrite files from another package, or a number of other reasons. *dpkg* provides a number of `--force` options you can use to ignore these problems and proceed with package installation.

The `--force` options are intended only for experts who are extremely familiar with *.deb* packaging and dependencies within the system. Usually packages refuse to install for good reasons, and if you are a new user and force the installation anyway, you will more likely than not end up with a broken system. Proceed with caution.

To see the complete list of `--force` options, type:

```
whiprush@ubuntu:~$ dpkg --force-help
```

Some of the more useful options include:

`--force-hold`

Install a package even if it is marked *hold*.

`--force-overwrite`

Install a package even if it might overwrite files from another package.

`--force-depends`

List any missing dependency errors as warnings and proceed anyway.

`--force-conflicts`

Even if a package conflicts with another package, install anyway.

So if you have a *.deb* you need to install that overwrites files from another package, and you have done your homework and confirmed that it is okay to proceed, type:

```
whiprush@ubuntu:~$ sudo dpkg -i --force-overwrite
                   packagename.deb
```

Remove a Package

Occasionally, you may need to remove a standalone package manually. *dpkg* provides the `-r` and `-P` options to remove and purge packages respectively. To remove a package, type:

```
whiprush@ubuntu:~$ sudo dpkg -r
                   packagename
```

Note that you don't specify the name of the complete *.deb* file you might have installed previously, only the name of the package itself. When given the `-r` option, *dpkg* will find and remove all of the files for this package except for configuration files, which it leaves behind in case you install the program again. If you want to purge the system of all files, including configuration files, use `-P` instead:

```
whiprush@ubuntu:~$ sudo dpkg -P
                   packagename
```

Hack 58. Search for Packages from the Command Line

When you aren't exactly sure which Ubuntu package has the files you need, use these handy command-line tools to find out.

The Ubuntu package-management system is truly impressive. Simply choose a package name, and Ubuntu will automatically download and install it and any dependencies it may have. What happens, though, when you don't know the exact name of the package you want to install?

The graphical package manager Synaptic has a search tool, but there are two separate command-line tools, *apt-cache* and *apt-file*, that you'll find are even more powerful. Each tool suits a particular type of search. *apt-cache* lets you search for a package based on keywords (among some of its other uses), and *apt-file* allows you to search for a package based on files inside that package. Each approach is useful in different circumstances, and this hack introduces you to how to use both tools effectively.

Search for Packages Based on Keywords

The *apt-cache* tool is actually much more than a keyword-based package search utility. *apt-cache* is a general tool used to query and manipulate the *apt* package cache. To search within the *apt* package cache, use the `search` argument followed by one or more keywords. For instance, if I wanted to find packages having to do with Adobe's Acrobat Reader, I could run the following search:

```
$ apt-cache search acrobat reader
xpdf - Portable Document Format (PDF) suite
xpdf-utils - Portable Document Format (PDF) suite -- utilities
acroread - Adobe Acrobat Reader: Portable Document Format file viewer
acroread-debian-files - Debian specific parts of Adobe Acrobat Reader
acroread-plugins - Plugins for Adobe Acrobat(R) Reader
mozilla-acroread - Adobe Acrobat(R) Reader plugin for mozilla / konqueror
xpdf-chinese-simplified - Portable Document Format (PDF) suite -- simplified Chinese language support
xpdf-chinese-traditional - Portable Document Format (PDF) suite -- traditional Chinese language support
xpdf-japanese - Portable Document Format (PDF) suite -- Japanese language support
xpdf-korean - Portable Document Format (PDF) suite -- Korean language support
```

The resulting packages listed in the output either directly relate to Acrobat Reader, or indirectly relate to the PDF format. Note that because this is a keyword-based search, keywords will sometimes turn up packages that are only marginally related to what you are looking for. Use multiple keywords to narrow things down. Alternatively you can pipe the results through *grep* to look for particular keywords in the package names themselves. For instance, if I knew the package I was looking for contained the word *acro*, I could type:

```
$ apt-cache search acrobat reader | grep acro
acroread - Adobe Acrobat Reader: Portable Document Format file viewer
acroread-debian-files - Debian specific parts of Adobe Acrobat Reader
acroread-plugins - Plugins for Adobe Acrobat(R) Reader
mozilla-acroread - Adobe Acrobat(R) Reader plugin for mozilla / konqueror
```

Search for Packages Based on Files They Contain

There are times when you wish to have a particular file, and you want to track down which package installs this file. Or perhaps you know that you need a particular library for a program, but you don't know the exact name of the package that contains it. While you could use *apt-cache* and guess, Ubuntu offers a more precise approach with the *apt-file* tool. *apt-file* lets you specify a particular file or directory name, and it will search through a package database and print out the name of a package that contains that file or directory, along with the complete path to it.

By default, Ubuntu does not include the *apt-file* tool. To install, it you will need to make sure that you include the universe repository [Hack #60] in your list of package repositories. Then choose "apt-file" from the Base System (*universe*) section if you use Synaptic [Hack #55]; otherwise open a terminal and type:

```
$ sudo apt-get install apt-file
```

Unlike *apt-cache*, *apt-file* does not use the included *apt* package cache for its searches. Instead it maintains its own database that needs to be updated as you do with *apt-get update*. Before you use *apt-file* for the first time, you will need to do a full update:

```
$ sudo apt-file update
```

Depending on your Internet connection, this might take some time, since *apt-file* needs to download all of the repository lists you currently have configured. Every once in a while, you will want to update this repository list, especially if you intend to do a number of searches and want newer packages to show up in the results.

After the repository list is up-to-date, you can search for packages much as you do with *apt-cache*. For instance, to find out what package contains the configuration file */etc/jvm*, type:

```
$ apt-file search /etc/jvm
java-common: etc/jvm
```

Notice that the results not only tell you which package contains this file (java-common), but also give you the full path to that file. This can be useful when more than one package provides a file with the same name but in different directories. With the full path displayed, you can more easily figure out which package you were looking for.

Hack 59. Install Software from Source

When there's no repository and no package, there's still hope. Here's how to build and install standalone programs from source.

It used to be that when you wanted to install a new program under Linux, you would locate the project's home page, find a source tarball, download it, and then extract and compile the source. After a lot of text scrolled by in your terminal, finally your program would be installed. This sort of method was so common, in fact, that many distributions would ship with all of the basic compilers and libraries installed. Ahh, the good old days....

These were *not* the good old days. The reality was that along with the steps I mentioned above, you had some additional chores:

1. You had to revisiting the project home page when you realized that there were numerous library dependencies that you needed to first track down, along with any other programs this software relied upon.
2. If the software depended on other programs, you usually needed to download and compile those first, only to discover that they had external dependencies, too, so you had to revisit their project pages to track them down as well.
3. Finally, after all the time you spent tracking down all dependencies and successfully installing the software, the developer would have just released a new version with more features and fewer bugs, so you would have to start again from scratch.

Nowadays, almost every distribution including Ubuntu not only uses *packages* (software that has been compiled and bundled together for you), but also incorporates some sort of automatic dependency management. When you want to install software, you just tell Ubuntu which program to install and it will go find it and any dependencies, and install it all for you. When you want to upgrade, Ubuntu will manage any new dependencies and new versions for you. So little software these days requires you to download and compile source that (by default) Ubuntu doesn't even install all of the software you need to compile source code. Nine times out of ten, the package you want will either be packaged for the distribution already, or at least be available in a precompiled *.deb* package. ("Install and Remove Standalone .deb Files" [Hack #57] describes how to install these packages.)

Having said all of this, there are still a few circumstances that crop up that require you to download and compile a program from source:

- A brand new open source project has been released. It is so new that no one has yet decided to package it for Ubuntu.
- There is some special feature in the kernel or other piece of software that isn't enabled by default, or a patch you would like to add, so that you must compile the kernel from source. (However, you might find that you can customize the source .deb package [Hack #63] to get what you need.)
- A third-party hardware manufacturer has released drivers for Linux; however, they are released only in source form (or perhaps they are released only as RPMs built for Red Hat or SUSE along with a *.tar.gz* file for any other distribution), and Ubuntu hasn't released some sort of prebuilt package for it.
- You are a programmer yourself, and you would like to help out with an open source project by submitting patches, or perhaps you want to start a whole new project.
- A new version of a program has come out, and you can't wait for it to be packaged, you will have to build it yourself. (However note that, depending on the program, you risk breaking other parts of the system that depend on the older version of the program. Typically, when a program has been updated that affects other programs, it and its dependencies will be held back temporarily by the package manager until they are all ready to be installed together.)

In the world of software installation under Ubuntu, compiling a program from source should be the absolute last resort. There is a good chance that you can break your system and its dependencies if you don't know what you are doing. Please try all of the other avenues at your disposal, including Ubuntu's built-in package managers mentioned in "Manage Packages from the Command Line" [Hack #54], "Manage Packages with Synaptic" [Hack #55], and "Manage Packages With Adept" [Hack #56]. If that doesn't work, try to locate a precompiled *.deb* package and follow the steps in "Install and Remove Standalone .deb Files" [Hack #57]. Finally, if none of those choices are available, then read this hack.

Install Compiler Tools

If you do need to compile from source, first you will need to install a compiler and all of the other packages essential to building programs from source. Ubuntu makes this easy: just open your preferred package manager, and find and install the *build-essential* package. This will grab and install the GNU C compiler (*gcc*), *make*, *g++*, and a number of other programs you will need:

```
$ sudo apt-get install build-essential
```

Get and Compile the Source

Once all of these programs have installed, download the tarball for your program and extract it somewhere, such as your home directory. Most tarballs extract into their own directory (often named after the program name), so the next step is to *cd* into that directory:

```
greenfly@ubuntu:~$ tar xvzf
                   program.tar.gz
greenfly@ubuntu:~$ cd
                   program/
```

Most programs these days follow a common three-step compile method of configuring, but before you launch into the process, you will want to read the installation instructions the developer has provided (usually in a file called *INSTALL* or *README* in the main source directory, or otherwise provided on their main project page).

Step 1: configure

The first step in the build process is usually to run the *configure* script that is located in the main source directory:

```
greenfly@ubuntu:~/program$ ./configure
```

This program will scan the system and make sure that all of the libraries that the program needs exist, as well as set any file paths or other settings. If you don't have all of the libraries the program needs, the *configure* script will *error out* and tell you what library you are missing (or what newer version you need). If you see this type of error, you will need to not only track down the package that contains that library, but also the development package that contains all of the header files for that library (in Ubuntu, most of these packages end in *-dev*). Once you get these libraries installed, run the *configure* script again to see if there are any other libraries you need.

Step 2: compile

Once the configure script exits successfully, the next step is to compile the source code. Most projects use *makefiles* these days along with the *configure* script, so the next step is to run *make* from the main source directory:

```
greenfly@ubuntu:~/program$ make
```

You should see a stream of compiler output roll by, and eventually you will end up back at the prompt. It's possible for errors to occur during this step, but these are generally more difficult to debug because they could possibly result from a syntax error in the source code or some other problem. If you do find some sort of compiler error, check the software's mailing list or other support channels (if one exists) for a possible known bug with the software; otherwise, file a bug report.

Step 3: install

Once the source code is compiled, the final step is to install the software on the system. Most programs include a function in their *makefile*s for installation; however, note that you will generally have to install the program as *root*, since it will want to install files under */usr* and other directories that are writable only by *root*. Make sure you are still in the main source directory and type:

```
greenfly@ubuntu:~/program$ sudo make install
```

The program will now be installed to the system.

Some programs also provide an "uninstall" function to remove the program from the system. Just change back to the main source directory and run:

```
greenfly@ubuntu:~/program$ sudo make uninstall
```

Hacking the Hack

In most cases, installing a package using this procedure will install the application in */usr/local*. But if you want to be sure that the installation will stay away from the parts of the filesystem patrolled by Ubuntu, add `--prefix=` *`pathname`* to the `./configure` command, as in:

```
greenfly@ubuntu:~/program$ ./configure --prefix=
                           /opt
```

This will almost always work. However, there are some programs that don't respect the `--prefix` option, and some programs (such as those that include kernel modules) will insist on spreading themselves all over your filesystem.

Hack 60. Modify the List of Package Repositories

Add extra Ubuntu software repositories to your system to gain access to thousands of new pieces of packaged software.

If you are used to installing programs on operating systems other than Linux, the idea of package managers and repositories might seem a bit foreign. To install a program on Windows or Mac OS X, you would insert a CD or download an installer from the Internet and run it. Under Ubuntu, software comes in *packages*. Packages are the different groups of files that make up a particular piece of software, along with the instructions Ubuntu needs to install and remove the software, dependency information, and so on. With packages, Ubuntu lets you install and update all of your software using a single tool. Whether you use Synaptic [Hack #55], Adept [Hack #56], or apt-get [Hack #54], your package manager will automatically download and install packages you specify along with any other packages (*dependencies*) they might require.

Instead of being randomly scattered on project pages around the Internet, Ubuntu packages reside in *software repositories*, which are centralized sites that contain a large number of packages. Each software repository contains a certain class of software that is generally intended for a particular release of Ubuntu, usually in both source and binary form (these are split up into their own repositories) and then further subdivided by what sort of support or software license the package has. Here are some of the main package repositories available for Ubuntu:

dapper

This is the main repository for the Dapper Drake release of Ubuntu. Prior releases of Ubuntu also have repositories named after their release names, so there are Breezy, Hoary, Warty, etc. repositories. When Ubuntu releases a new version, it will also create a repository

named after its release name. This method makes it easy to ensure that packages intended for a particular Ubuntu release don't get mixed up with other versions.

dapper-updates

This repository contains updates for a particular release, so you'll generally want to be subscribed to this repository (and you are by default).

dapper-security

Ubuntu uses this repository to separate out security updates that have been applied to a particular package. These updates are often *backported* to a particular version of software that ships with a release, so you aren't required to update to the latest version of a piece of software to get the security fix.

dapper-backports

This repository is disabled by default, but it contains updated versions of software that has been backported to work with Dapper. This repository is disabled by default because these backports do not undergo the level of testing that other packages do. It has been made available since it might be handy to have a newer version of software with new features.

These repositories are further subdivided into categories based on what level of support a package has and, in some cases, what sorts of licenses or restrictions the software has. If you, for instance, want only software that is fully supported by Ubuntu, you can ensure that by choosing only the *main* subcategory. Here are the primary subcategories:

main

The primary package category is called *main* (also referred to as Officially Supported) and contains all of the default Ubuntu packages that have been tested to such a level that they are officially supported by the distribution. As such, this category is enabled by default.

restricted

This category contains packages that are restricted due to copyright or possibly other aspects of their licenses such that they are considered nonfree. This repository exists so that users who want to use only "free" open source software can disable this category and pull software only from *main*.

universe

Ubuntu is based on the Debian Linux distribution but only contains a subset of Debian's software due to the amount of testing required to support each package. *universe* is a subcategory that contains software that the community has packaged and supports for Ubuntu. If you are looking for a particular piece of software and can't seem to find it, there's a good chance that it is in *universe* or the next subcategory, *multiverse*. Note that since these packages are not officially supported by Ubuntu, there's a chance that you risk some instability by installing software from this category.

multiverse

Like *universe*, the *multiverse* category contains software that is not officially supported by is maintained by the Ubuntu community. *multiverse* is to *universe* as *restricted* is to *main*. Here you will find contributed or nonfree packages that are not officially supported by Ubuntu.

Add and Remove Repositories

There are a few different ways to change the repositories you subscribe to, based on what tools you use for package management. "Manage Packages with Synaptic" [Hack #55] and "Manage Packages With Adept" [Hack #56] cover managing package repositories with those tools. Still, all of these package managers reference the same */etc/apt/sources.list* file. Here is a sample *sources.list* file for Dapper that illustrates the syntax:

```
# dapper
deb http://archive.ubuntu.com/ubuntu/ dapper main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ dapper main restricted

# dapper-updates
deb http://us.archive.ubuntu.com/ubuntu/ dapper-updates main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ dapper-updates main restricted
```

```
# dapper-security
deb http://security.ubuntu.com/ubuntu dapper-security main restricted
deb-src http://security.ubuntu.com/ubuntu dapper-security main restricted
```

Knowing how Ubuntu separates repositories into categories, you can see how the syntax of these lines work. The first section of a line specifies whether Ubuntu is going to pull binary packages (*deb*) or source packages (*deb-src*) for a particular repository. The next section of the line contains the URL for the particular repository which is usually either HTTP or FTP. Next, the particular repository is listed (in this example, *dapper*, *dapper-updates*, or *dapper-security*). Finally, you see a space-delimited list of subcategories to pull from. By default, Ubuntu will use *main* and *restricted* subcategories, but to add new subcategories just add their names at the end of a particular line. So for instance to add *universe* and *multiverse* to the Dapper repository for both binary and source packages, change:

```
deb http://archive.ubuntu.com/ubuntu/ dapper main restricted
deb-src http://archive.ubuntu.com/ubuntu/ dapper main restricted
```

to:

```
deb http://archive.ubuntu.com/ubuntu/ dapper main restricted universe multiverse
deb-src http://archive.ubuntu.com/ubuntu/ dapper main restricted universe multiverse
```

Once you save your changes, you will need to update your package list to use the new repositories either via `apt-get update` or through the Synaptic or Adept interface. To add new subcategories to other repositories, simply make the appropriate changes to their repository entries.

Unofficial Repositories

There are a number of other unofficial package repositories out on the Internet that you can also use with Ubuntu. Often these repositories provide a small subset of special packages that Ubuntu doesn't package by default, or perhaps provides a newer version of a package. These repositories generally will tell you the particular line to add to your *sources.list*, but be sure that the repository is designed to work with your particular Ubuntu release, or you may face compatibility conflicts between dependencies in the third-party repository and Ubuntu's native packages.

Hack 61. Cache Packages Locally With Apt-cacher

Save time and bandwidth when updating multiple Ubuntu machines by keeping a local package cache.

If you manage multiple Ubuntu machines, you probably wish there were some way to download new packages only once and install them on every machine. Better still, it would be good if it worked totally transparently so you could just use the regular package-management tools in Ubuntu and not care about what happens behind the scenes.

Apt-cacher allows you to do exactly that. With Apt-cacher running on one machine on your network, you can configure all your other machines to fetch packages through it.

apt generally uses HTTP to fetch packages from package servers; as a result, it's pretty easy to use a normal HTTP proxy like Squid [Hack #98] to cache packages locally. However, Squid is designed to cache lots of small items, while software packages are usually a few large items. You may find Squid drops large packages from its cache, which are the very packages most important to store for reuse. To make *apt* use a proxy, you can configure the option permanently in the config file (use `man apt.conf` for details) or just export the `http_proxy` environment variable by running a command like `export http_proxy=proxy.example.com:8080` prior to running *apt*.

Apt-cacher is different from many other caching systems because rather than being a standalone program, it runs as a CGI script under Apache. That has a number of advantages, such as making it small, simple, and therefore more robust because it doesn't need its own protocol-handling code. It also makes it very flexible because you can use Apache's built-in access-control mechanism in case you want to let only certain machines use your cache.

Apt-cacher itself needs to be set up on only one machine, the one you decide to use as your local cache. Then all computers on your local network have a setting modified to tell them to direct all package requests to your cache machine rather than directly to the package server.

Apt-cacher works by intercepting requests for packages and fetching them on behalf of local machines, while simultaneously storing them on disk in case other machines later ask for the same package. Once set up, there is no need to do anything differently to install packages: just install a package on one machine with *apt* or Synaptic, and it comes off the Internet; then when you install it on other machines later, it comes from the local cache. Easy!

Installing Apt-cacher

Getting Apt-cacher working involves two parts: setting up the cache server itself and then telling your local machines to use it.

Server setup

First, select a machine to use as your cache server. Apt-cacher puts very little load on the system, so you can safely run it on just about any machine you have available, even one that's normally used as a workstation. Probably the most critical things are to make sure your cache server has a fixed IP address, so other computers on your network can find it, and that there is plenty of disk space, because the cache itself can become quite large. Disk usage depends on how many packages you have cached, so the greater variety of software you run, the more space you will need. A few hundred megabytes is common, while large caches may need several gigabytes.

On the machine nominated to be your cache, start by installing the *apt-cacher* package:

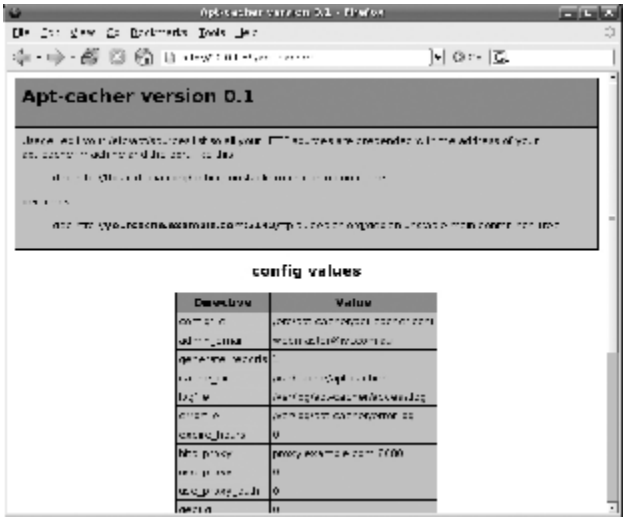
```
$ sudo apt-get install apt-cacher apache
```

This will also install Apache plus a couple of other packages, unless they were already in place. If you already have Apache installed, you should restart it:

```
$ sudo /etc/init.d/apache restart
```

and you're done. (If you installed Apache *after* you installed Apt-cacher, you should run the command `sudo dpkg-reconfigure apt-cacher`.) You can test that the installation worked properly by opening a web browser and going to the address http://*cache.example.com*/*apt-cacher*, replacing *cache.example.com* with the hostname or IP address of your cache server. If all went well, you'll see an information page generated by Apt-cacher, as shown in Figure 6-9.

Figure 6-9. Apt-cacher default screen



Client setup

Client machines don't need to have anything installed to use Apt-cacher: they just need to have their list of package sources modified so that they send their package requests to the cache server.

The list of package sources is stored in a file called */etc/apt/sources.list* (see "Modify the List of Package Repositories" [Hack #60] for information on modifying this file). Here is an HTTP entry for the *dapper* repository:

```
deb http://archive.ubuntu.com/ubuntu/ dapper main restricted
```

Each HTTP entry needs to have the address of your cache server prepended, so the previous entry becomes something like this:

```
deb http://cache.example.com/apt-cacher/archive.ubuntu.com/ubuntu/ dapper main restricted
```

Once you've modified all your entries, run:

```
$ apt-get update
```

to tell your machine to update its package list, and you're set. Any packages you install from now on will come via the cache server.

Configuration Options

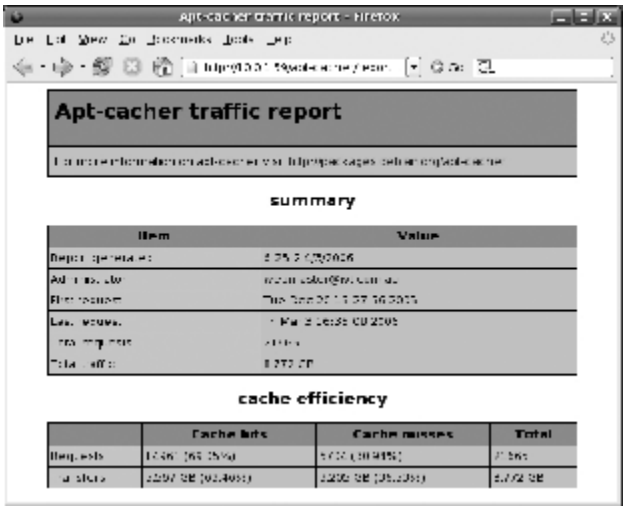
At this point, you've created a working installation of Apt-cacher without touching a single config setting on the cache server. However, Apt-cacher has a number of options you can set by editing the file */etc/apt-cacher/apt-cacher.conf*. You don't need to restart anything after editing this file; all changes are immediate.

The config file is very well commented, so for all the details, just read the file itself. Options include the ability to restrict use of your cache to specific IPv4 or IPv6 address ranges, restrict access to only approved software repositories, and pass requests through an upstream proxy server.

Traffic Reports

Apt-cacher can be configured to generate traffic reports daily. Generating reports is extremely fast even with a high-traffic cache and happens only once per day, so this option can safely be turned on without any impact on performance. To access the report, just point your browser at http:// `cache.example.com` / *apt-cacher/report*, and you should see something like Figure 6-10.

Figure 6-10. Apt-cacher traffic report



The actual report is generated by a script at */usr/share/apt-cacher/apt-cacher-report.pl*, which is run by cron and writes an HTML report out to */var/log/apt-cacher/report.html*.

Hacking the Hack

What if you want the report emailed to you each day? Because the report is in HTML format, it will need to be sent as an attachment. To do so, you can use the extremely versatile Mutt mail client. First install the *mutt* package:

```
$ sudo apt-get install mutt
```

Then edit */etc/cron.daily/apt-cacher* and add the following line to the end:

```
echo "Apt-cacher report \Qdate\Q" | mutt -a \\
  /var/log/apt-cacher/report.html -s "Apt-cacher report" user@example.com
```

(Be sure to replace `user@example.com` with your email address.) Run this whole command manually just to check that it works and to force Mutt to create a mail directory for the user if it doesn't already exist, and from now on you'll receive daily updates on the efficiency of your Apt-cache.

Hack 62. Create a Ubuntu Package

Package your software for easy installation on Ubuntu.

Ubuntu is built on Debian and uses the *.deb* package format to ensure all software is installed in a consistent way. If you want to make your software really easy to install on Ubuntu, you need to know how to put it into a Debian package.

There is a large variety of helper tools and build suites to assist with creating Debian packages, and just about as many approaches to streamlining package creation as there are package maintainers. To fully understand how Debian packages work, start by exploring the Debian New Maintainers Guide (http://www.debian.org/doc/maint-guide), Debian Policy Manual (http://www.debian.org/doc/debian-policy) and Debian Developers Reference (http://www.debian.org/doc/manuals/developers-reference), which between them, run to many hundreds of pages of detailed information.

This hack is a quick-and-dirty introduction to show you how to build a basic binary package as quickly as possible. It assumes that the program you want to package is in a source tarball named *myprogram-version.tar.gz* and can be installed with a simple procedure [Hack #59] such as:

```
$ ./configure
$ make
$ sudo make install
```

The procedure for packaging libraries and other types of software can be more involved, so consult the Debian references mentioned earlier for more information.

Start by installing some of the developer tools:

```
$ sudo apt-get install fakeroot debhelper build-essential \\
             lintian dh-make devscripts
```

Create a directory to work in so all your files stay neat:

```
$ mkdir

                 myprogram
```

Copy the source tarball into the *myprogram* directory and extract it (the directory that this process creates must have a version number in it, as in *myprogram-1.0*):

```
$ tar zxf
                myprogram-1.0.tar.gz
```

Create Package Files

You now have both a tarball and an extracted copy of the source code. Go into the extracted source code directory and run *dh_make* to set up sample files for you:

```
$ cd
                myprogram-1.0
$ dh_make -e
                user@example.com
            -f
                ../myprogram-1.0.tar.gz
```

dh_make will ask you what sort of package you want to create. In this case, it's a single binary package, so select that option (`s`, not `b`). *dh_make* will then create a *debian* directory containing all the files necessary to build the package, and the email address you provided will automatically be inserted into the package changelog. Now go into the newly created *debian* directory and customize some of the files as necessary, but don't run *dh_make* again: it only needs to be run when the package is first set up, and running it again in an already-configured package will almost certainly break it. Here are the files you may want to change:

control

The *control* file contains meta information about the package. Most of the fields in the file should be self-explanatory, so change any entries as necessary and put in short and long descriptions for the package. Note that any blank lines in the long description need to contain a single period.

rules

The *rules* file is a *makefile* that is executed at package build time to compile the package source into a binary. The file will have been automatically configured by *dh_make* to suit a typical build process, but if your package requires any unusual steps to be performed, you may need to edit the appropriate section of the *rules* file.

preinst, postinst, prerm, and postrm

Because package installation and removal can sometimes require arbitrary commands to be run, the Debian package format provides support for four optional scripts that are run before and after installation, and before and after removal. If you want to use any of them, rename the appropriate *.ex* file to remove the extension and edit the file to add commands you want to execute.

Build the Package

Now that the package source directory has been set up, you can build the actual package. This needs to be done from inside the extracted source directory, so if you are still inside the *debian* directory after customizing the package scripts, you need to `cd ..` to back up one level, then run *dpkg-buildpackage*:

```
$ dpkg-buildpackage -b -uc -rfakeroot
```

The `-b` option specifies that only a binary package should be built. If you also want to build a source package, leave that option off. The `-uc` option specifies that the *changes* file in the package should not be GPG-signed, so if you have a GPG key and want to sign the package, you can leave that option off as well. The `-rfakeroot` option specifies that *fakeroot* should be used to create the temporary build environment.

Once that process has finished, you will have a couple of new files outside your source directory, including a package named *myprogram-1.0_i386.deb* or similar. Before going any further, you can use a program called *lintian* to run some automatic checks on the package to make sure there are no obvious mistakes. *lintian* performs hundreds of checks to make sure the package complies with the official Debian policy and packaging guidelines, so even though it may generate warnings that you don't care about for packages you create for personal use, it's still a great way to check that you haven't missed any important steps.

Run *lintian* with the path to the *changes* file that was created along with your new package:

```
$ lintian
                myprogram_1.0.1_i386.changes
```

You can now use the instructions in "Install and Remove Standalone .deb Files" [Hack #57] to install and test your new package, and even "Create Your Own Package Repository" [Hack #65] if you want to publish it for the world to use.

Update the Package

When the software itself is released as a new version, you need to build a new package to suit. Start by copying the new source tarball into your working directory alongside the original version, and then extract it. You should now have two source directories and two tarballs. Copy your customized *debian* directory from the old source directory to the new one:

```
$ cp -a
                myprogram-1.0
            /debian
                myprogram-1.1/
```

Then enter the new source directory and increment the Debian changelog:

```
$ cd
```

```
                myprogram-1.1

$ dch -i
```

The `dch -i` command will throw you into an editor and automatically place a new entry at the top of the package changelog; it will also auto-complete your name and email address as well as increment the version number. If the new version number is not correct, just edit it to suit the new upstream version number, write a brief explanation for the new release, and save and exit the editor.

Now rerun the command to build the package:

```
$ dpkg-buildpackage -b -uc -rfakeroot
```

Hack 63. Compile a Source Package

Rebuild packages with custom options to suit your architecture, environment, or whims.

Software is generally installed on Ubuntu using binary packages that contain a precompiled copy of the program (the *binary*), but sometimes it can be useful to recompile the program yourself using custom options. To make that process easier, all official Ubuntu packages are also available in *source* form, and tools are provided to enable you to build your own custom binary package from the source package.

To compile a source package, take the following steps.

Enable Source Repositories

Each official binary-package repository in your */etc/apt/sources.list* has a matching source-package repository that has the exact same address but a *deb-src* prefix instead of *deb*:

```
deb http://archive.ubuntu.com/ubuntu dapper main restricted
deb-src http://archive.ubuntu.com/ubuntu dapper main restricted
```

You can "Modify the List of Package Repositories" [Hack #60] if you don't already have source repositories enabled.

Install Package Build Tools

Install the basic tools required for building binary packages:

```
$ sudo apt-get install devscripts build-essential fakeroot
```

Install Build Dependencies and Fetch the Source

Packages have varying build requirements that are defined in the package header. For example, rebuilding a package for a PHP module requires that you have the PHP headers installed on your system so they can be linked when the module is compiled. The package system can take care of fetching all the build dependencies for you automatically:

```
$ sudo apt-get build-dep
                packagename
```

For example, this command fetches dependencies for the *php4-apd* module:

```
$ sudo apt-get build-dep php4-apd
```

Next, fetch the source package you want to rebuild. Note that unlike almost every other use of *apt-get* you do not need to run this with *root* privileges because it doesn't actually install the package: all it does is download the source package and place it in your current directory. To keep things neat, you may want to create a directory in which to do the build and change into it before fetching the source:

```
$ mkdir php4-apd
$ cd php4-apd
$ apt-get source php4-apd
```

This will place a number of files in your current location, including a directory containing a fully extracted copy of the package:

```
jon@jon:~/php4-apd$ ls -l
drwxr-xr-x 6 jon jon   4096 2005-11-30 14:43 php4-apd-0.4p2
-rw-r--r-- 1 jon jon    532 2005-12-02 04:25 php4-apd_0.4p2-6.dsc
-rw-r--r-- 1 jon jon 368800 2005-12-02 04:25 php4-apd_0.4p2-6.tar.gz
```

Apply Changes

Enter the extracted package directory:

```
$ cd php4-apd-0.4p2
```

Any necessary changes can now be made to the extracted copy of the package, but note that the build process will be controlled by a special *makefile* in *debian/rules* rather than any *makefile*s provided with the original program. If you need to make changes to the configure or build flags, you should apply them by editing *debian/rules*.

Build Binary Package

If you are not listed as the package maintainer in *debian/control* and *debian/changelog*, you will need to pass in flags to tell *debuild* not to sign the new package:

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

The `-us` and `-uc` flags signify unsigned source and unsigned changes files, and the `-b` flag signifies to build only a binary package.

You will now have a number of new files in your working directory, including a *.deb* binary package file built with your custom options applied. You can use *dpkg* to install and remove standalone packages [Hack #57] and make sure it works as intended, and even set up a package repository [Hack #65] if you want to distribute your custom package to the world.

Hack 64. Convert Non-Ubuntu Packages

Packages created for other Linux distributions can be installed on your Ubuntu system with a little TLC.

Various Linux distributions use their own packaging schemes, and although there are a huge number of packages available directly within Ubuntu, there may be times you come across software that's available only in a package for Red Hat or some other distro, only in source-code form, or only as a Debian package that has been built against a different version of the toolchain.

If the software is available only as a tarball, you can create a package [Hack #62] and install it directly. If the software is available as a Debian package but hasn't been compiled specifically for Ubuntu (perhaps it's only in Debian Experimental, for example), you can fetch the source package and then use the standard package tools to compile a source package [Hack #63]. And if the software is provided in an alternative package format altogether, such as an RPM, you can use Alien to convert it into the correct format.

Convert Packages with Alien

Alien is a utility that understands a number of package formats—including RPM (Red Hat Package Manager), LSB (Linux Standards Base), SLP (Stampede), Slackware, PKG (Solaris), and DEB (Debian package, used by Ubuntu)—and can convert from one to another.

Start by downloading the package you need to convert. Then install Alien:

```
$ sudo apt-get install alien
```

Now you can feed it your package to convert:

```
$ alien -c
                myprogram-1.0.1-1.i386.rpm
```

The `-c` flag tells Alien to also convert any installation or removal scripts that it finds in the original package.

Once Alien has finished, you will have an equivalent *.deb* package; use the instructions in "Install and Remove Standalone .deb Files" [Hack #57] to install it and check that it works. If things didn't go quite as planned, however, read on.

Solve Package Conversion Problems

Software packages can be remarkably complex and contain scripts that perform specific setup routines when the package is installed, upgraded, or removed. These scripts may contain references to programs that are located in different places on different distributions, and this is almost impossible to convert totally automatically, so you have to understand that Alien performs conversions on a "best effort" basis. If the original package relies on another program already existing on your system, the installation may fail in strange and obscure ways that are hard to track down.

If the package appeared to install properly but doesn't seem to work, the first step to debugging the problem is to use *dpkg* to find where the various files were put on disk:

```
$ dpkg -L
                myprogram
```

Doing so may uncover obvious problems immediately, such as a program binary being installed in an unusual location such as */usr/local/bin* instead of the more usual */usr/bin*.

If that doesn't help, you can try running Alien again, but this time add the `-g` flag that tells it to just extract the original package instead of converting it:

```
$ alien -cg
                myprogram-1.0.1-1.i386.rpm
```

This will open the package up into its individual parts, allowing you to look inside and examine any scripts used to install the software, as well as examine the directory structure that will be used when the package is installed. Editing the installation script to suit your environment and then running it by hand may be enough to get the package working.

Convert Ubuntu Packages to Other Formats

The default behavior for Alien is to convert any package passed to it into a *.deb* package. That suits Ubuntu systems just fine, of course, but you can also use Alien to convert *.deb* packages to other formats, or to convert between formats. For example, to convert an Ubuntu package to RPM for installation on a Red Hat machine, run:

```
$ alien -c --to-rpm
```

```
myprogram_1.0.1_i386.deb
```

Hack 65. Create Your Own Package Repository

Create and manage a package repository for distributing your own Ubuntu packages.

Ubuntu uses online *package repositories* to locate available software and then download it to your computer for installation. If you create your own Ubuntu packages, you can install them directly using dpkg [Hack #57], but that's not very convenient when you want to make your packages publicly available or install them on a large number of computers. The solution is to build your own package repository, just like the ones used to distribute the official Ubuntu packages, and populate it with your own custom packages. Once your packages have been published in a repository, anyone can then use it to install your packages using any of the standard package-management tools such as *apt*, Synaptic, or Adept. All they need to do is add the address of your repository to their system [Hack #60].

This also makes it much easier for users of your software to stay up-to-date because their system will be able to automatically detect new versions of your packages and ask them if they want to update.

Anatomy of a Repository

A Ubuntu package repository is actually quite simple. In its simplest form, it can be just a number of packages placed on a web or FTP server along with a special *Packages.gz* file that describes them. Each package is a self-contained *.deb* file that can be downloaded and installed on a computer, while the *Packages* file acts as a directory of the packages in that particular repository and includes information about each package such as the name, description, version, dependencies, name of the maintainer, and location of the actual package files. By fetching the *Packages* files from various repositories, Ubuntu can provide the user with a list of available packages and their descriptions, and determine dependencies without having to download each package individually. For a small repository, the *.deb* and *Packages* files are pretty much all you need to provide, and setting everything up is quite simple.

Larger repositories often include additional files and are structured differently to make them more efficient. For example, some larger repositories place packages into subdirectories named after the first letter of the package name, rather than just lumping them all in the same directory: *audacity* and *aumix* go into an *a/* subdirectory, *blam* goes into *b/*, and so on. That way no single directory contains too many files, and the performance of the server's filesystem doesn't become a bottleneck. That's not generally a problem until you have many thousands of packages, though, so unless you're setting up an extremely large repository, it's not something you should have to worry about.

Some large repositories also implement *package pools*. Pools are a way to save disk space and processor time when you have multiple repositories that share some of the same packages. This is the case with the official Debian repositories, for example, where some packages can exist in exactly the same version in two or more of the Experimental, Unstable, Testing, and Stable repositories at the same time. Pools allow all packages in all versions to be stored in the same big directory structure on the server, and each repository then acts as a sort of virtual index that lists or links to only the packages that are intended to be available in it. The *Packages* file for each of the repositories therefore references just a subset of the packages in the total package pool.

Getting Started

To create your own repository, first you need to create the software packages themselves (see "Create a Ubuntu Package" [Hack #62], "Compile a Source Package" [Hack #63], and "Convert Non-Ubuntu Packages" [Hack #64]). Install the packages manually on your local system and make sure they function correctly before you try publishing them in a repository.

You will also need some web-hosting space on either a server that you run or web space provided by a hosting company. In any case, it needs to have a consistent address so that other computers know where to find it. You could just use the server's IP address, but ideally it should have a hostname set up properly in DNS.

The main trick to maintaining a repository is generating the *Packages* file properly, and there are a number of tools to help you do so simply and consistently. The tool you'll use for this hack is *apt-ftparchive*, a simple program that you can run whenever you want to update your repository. It's best suited to manually maintaining a small-to-medium number of packages in a simple repository. Other tools you can consider include *mini-dinstall*, which runs as a daemon and automatically handles new packages as they are uploaded to a special Incoming directory, and *dpkg-scanpackages*, which is run manually. The granddaddy of all repository managers is *dinstall* itself, which is designed to run very large repositories, such as the official Debian package servers, with tens of thousands of packages.

apt-ftparchive is part of the *apt-utils* package, so the very first thing to do is install *apt-utils*:

```
$ sudo apt-get install apt-utils
```

Next, you need to create a directory inside your web server's document root to store the packages. You could just create a single directory and put everything in there, but for future flexibility, it's a good idea to also create a subdirectory to allow you to have subsets of packages. This is typically used to publish packages built against different releases, such as different versions of the same package for Dapper and Breezy. In this example, you'll create a subdirectory for Dapper packages, but you can name it something else if you prefer. The repository should be in your web server's document root, so if your document root is */var/www* (the standard location in Ubuntu), you could just run:

```
$ mkdir -p /var/www/ubuntu/dapper
```

Now copy the *.deb* packages you want to serve into that directory. At this point, you should be able to access the packages by pointing a web browser at http://localhost/ubuntu/dapper/ *your_package.deb*.

Finally, open a shell and move into the *ubuntu* directory to execute *apt-ftparchive* and then compress the resulting *Packages* file:

```
$ cd /var/www/ubuntu
$ apt-ftparchive packages dapper > dapper/Packages
$ gzip dapper/Packages
```

The `packages` argument tells *apt-ftparchive* to process packages it finds; `dapper` is the path to search, which in this case is the *dapper* subdirectory; and `> dapper/Packages` redirects the output into a file called *Packages*, which is then compressed to make it download faster. Easy!

At this point, you have a repository that you can access from other computers to install the packages you've just published, but first you need to tell them how to find it by adding a line to their */etc/apt/sources.list* file. The entry should look something like:

```
deb http://www.example.com/ubuntu dapper/
```

where *www.example.com* is replaced by the actual address of your package server.

Then run `apt-get update`, and you should see your repository being indexed along with the normal Ubuntu repositories, after which you can `apt-cache search` for your packages, and they should be reported as being available. With a simple `apt-get install` *`packagename`*, you can install any of the packages you've published.

Update the Repository

Created a new or updated package? No problem: just copy it into the repository alongside the other packages and follow the steps described earlier to rerun *apt-ftparchive* and generate a new *Packages* file. Because updated packages have different filenames, you'll need to manually delete superseded packages prior to rerunning *apt-ftparchive*; otherwise, your repository will just keep getting bigger.

Manage a Repository Without Shell Access

While it's simplest to create your repository directly on the web server, it's certainly not essential. If you have access to web space by FTP but can't get shell access to the server, you can create a repository on your local computer following the instructions in this hack and then just use FTP to upload all the directories and files to the correct location on the server. Then, when you do an update to any of the packages in the repository, just rerun *apt-ftparchive* locally and then bring the copy on your web server up-to-date again by FTP.

Hack 66. Convert Debian to Ubuntu

Convert an existing Debian installation to Ubuntu without reinstalling from scratch.

While the usual approach to switching Linux distributions is to erase your disk and reinstall from scratch, Ubuntu is based on Debian, so it's possible to switch directly from one to the other—provided you're willing to spend time tweaking and fixing obscure problems. It's even possible to have your computer track both Debian and Ubuntu simultaneously and install packages from one or the other at will.

Convert Debian to Ubuntu

ED/AUs: Just noticed that the above Head exactly matches the Hack title. Can we change this head somehow?

The naive solution to converting an existing Debian system to Ubuntu is to just edit your */etc/apt/sources.list* to switch all the Debian archive references to the equivalent Ubuntu archives and then upgrade all packages. That's a good start, but unfortunately it will most likely leave you with a fairly broken system because many of the libraries in Debian will have newer versions than their equivalents in Ubuntu, so it's necessary to do a bit more work to end up with a usable system.

Edit your */etc/apt/sources.list* to comment out all the Debian archives and add entries for the Ubuntu archives. If you need to generate an Ubuntu *sources.list* file, you can use the "source-o-matic" tool available online at http://www.ubuntulinux.nl/source-o-matic, or you can just put in some basic entries as a starting point:

```
deb http://archive.ubuntu.com/ubuntu dapper main restricted
deb http://security.ubuntu.com/ubuntu dapper-security main restricted
```

Then update the package list and install *sudo* if you don't already have it installed:

```
# apt-get update
# apt-get install sudo
```

Since Ubuntu relies so much on the primary user having *sudo* privileges, it's also a good idea to edit your */etc/sudoers* by running *visudo* and adding a line like this:

```
%admin  ALL=(ALL) ALL
```

Then place your primary user into the *admin* group, replacing `username` with your actual username:

```
# usermod

                username

            -g admin
```

As a first step to getting all the installed packages in order, you can manually force a reinstall of every package on your system to the specific version available in Ubuntu, but since a typical Debian system has over a thousand packages installed, that can be rather tedious. You can save yourself some time by writing a little script like this:

```
#!/bin/sh
for name in \Qdpkg --get-selections | grep '[[:space:]]install$' \\
  | awk '{print $1}'\Q
do
   sudo apt-get install --assume-yes ${name}=\Qapt-cache show $name \\
     | grep '^Version' | awk '{print $2}'\Q
done
```

It's highly likely that some packages you have previously installed from the Debian archives won't "cross-grade" cleanly to the version available in Ubuntu, and you may have to manually uninstall some packages to allow the process to complete. Depending on how your system is set up, you may need to do a lot of poking and prodding to get to the point where:

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
```

can execute cleanly.

Once you have your existing packages converted to the Ubuntu versions, it's time to pull in the core Ubuntu packages:

```
$ sudo apt-get install ubuntu-standard ubuntu-desktop
```

The result won't be perfect, but you should then have a system that is a reasonable approximation of a full Ubuntu install.

Mix Ubuntu and Debian

If you want to be able to install packages from either Debian or Ubuntu at will, you can make use of a package-management feature called *pinning*, which is designed to allow you to lock packages to specific versions while other packages are upgraded.

Start by adding some basic Debian archive entries to */etc/apt/sources.list*:

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

If you don't have one already, create a file called */etc/apt/preferences* and put in these entries:

```
Package: *
Pin: release a=dapper
Pin-Priority: 700

Package: *
Pin: release a=sarge
Pin-Priority: 600
```

What this does is give the packages in Dapper a higher priority than the packages in Sarge, so if you `apt-get install` *`foo`* a package, the system will try to install the Dapper version of a package if it's available and fall back to using Debian packages if necessary. In reality, the algorithm for determining package priority is rather more complex than a simple priority, and to fully understand it, you should read the `apt_preferences` manpage (`man apt_preferences`).

Because *apt* in Ubuntu uses a new security infrastructure to check package signatures, you will also need to install the Debian archive key, or your system will complain every time it tries to install a package from the Debian archives. The Debian archive keys are replaced every year but are located at an address like http://ftp-master.debian.org/ziyi_key_*`year`*.asc, where *`year`* is the current year. Use *wget* to retrieve the current key, import it into your GPG keyring, and then check the fingerprint:

```
$ wget http://ftp-master.debian.org/ziyi_key_2006.asc
$ gpg --import ziyi_key_2006.asc
gpg: key 2D230C5F: public key "Debian Archive Automatic Signing Key (2006) <ftpmaster@debian.org>" imported
gpg: Total number processed: 1
gpg:               imported: 1
$ gpg --fingerprint 2D230C5F
pub   1024D/2D230C5F 2006-01-03 [expires: 2007-02-07]
      Key fingerprint = 0847 50FC 01A6 D388 A643  D869 0109 0831 2D23 0C5F
uid          Debian Archive Automatic Signing Key (2006) <ftpmaster@debian.org>
```

If you want to be really careful, you can then use this fingerprint to prove the authenticity of the key via Debian's web of trust.

Now install the key into *apt* so that it will trust packages signed by it:

```
$ gpg --armor --export 2D230C5F | sudo apt-key add -
```

You're now ready to install packages from either Ubuntu or Debian. Any packages you want to specifically pull from the Debian archive can be accessed by adding a `-t` (target release) flag to *apt*:

```
$ sudo apt-get install -t sarge
```

Chapter 7. Security

When you hear stories about how much more secure Linux is than other operating systems, it can be tempting to rest easy. However, the truth is that any given Linux system is more likely to be secure only when it's in good hands. This chapter collects a number of hacks that will make you a better steward of your Ubuntu systems.

In this chapter, you'll learn about how Ubuntu relies on *sudo* to keep *root* privileges at the minimum needed to keep the system running smoothly. You'll also learn how to configure your system to download and install security updates, so you can take advantage of the Linux community's famously quick responses to security flaws. But even with these practices under your belt, there's more you can do.

You can protect your network with a firewall to make it even harder for intruders to gain access. And if that's not enough, you can use industrial-grade encryption to protect sensitive information on your system (so if your system is stolen or otherwise compromised, the attackers won't be able to use what they find—unless you used a lousy passphrase or wrote it down on a piece of paper). You'll find hacks showing how to do all this and more in this chapter.

Hack 67. Limit Permissions with sudo

Leverage Ubuntu's default sudo installation to allow fine-grained control over privileged access.

If you have used a number of different Linux distributions in the past, one surprising thing you'll notice the first time you use Ubuntu is that it disables the *root* account. For most other distributions, the installer prompts you for *root*'s password, and when you need to get work done as *root*, you log in or use the *su* command to become *root*, and type in *root*'s password. Since Ubuntu's *root* user has no password by default, you must use the *sudo* command to run commands as *root*. *sudo* sets up a way to allow access to *root* or other user accounts with fine-grained controls over what a person can do as that user. Plus the way *sudo* works is that it prompts you for *your* password, not that of the other user you want to switch to. This allows an administrator the ability to grant particular types of *root* access to users on the system without them all knowing the *root* password.

The default *sudo* configuration in Ubuntu is pretty basic and can be found in the */etc/sudoers* file. Note that you *must never* edit this file using a standard text editor. You must use the *visudo* tool. *visudo* is required because it will perform extra validation on the *sudoers* file before you close it to make sure there aren't any syntax errors. This is crucial because a syntax error in a *sudoers* file could lock out all of the users on your system. Here are the roles defined in the default Ubuntu */etc/sudoers* file:

```
# User privilege specification
root    ALL=(ALL) ALL

# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL
```

The first rule allows *root* to use *sudo* to become any other user on the system, and the second rule allows anyone who is a member of the admin group to run any command as *root*. So when you want to run a command as *root* on a default Ubuntu system, type `sudo` followed by the command to run. For instance if you wanted to run `apt-get update` as *root*, you would type:

```
$ sudo apt-get update
```

/etc/sudoers Syntax

To fully explain the syntax of */etc/sudoers* I will use a sample rule and break down each column:

```
jorge  ALL=(root) /usr/bin/find, /bin/rm
```

The first column defines what user or group this *sudo* rule applies to. In this case, it is the user *jorge*. If the word in this column is preceded by a `%` symbol, it designates this value as a group instead of a user, since a system can have users and groups with the same name.

The second value `(ALL)` defines what hosts this *sudo* rule applies to. This column is most useful when you deploy a *sudo* environment across multiple systems. For a desktop Ubuntu system, or a system where you don't plan on deploying the *sudo* roles to multiple systems, you can feel free to leave this value set to `ALL`, which is a wildcard that matches all hosts.

The third value is set in parentheses and defines what user or users the user in the first column can execute a command as. This value is set to `root`, which means that *jorge* will be allowed to execute the commands specified in the last column as the *root* user. This value can also be set to the `ALL` wildcard, which would allow *jorge* to run the commands as any user on the system.

The last value (`/usr/bin/find, /bin/rm`) is a comma-separated list of commands the user in the first column can run as the user(s) in the third column. In this case, I'm allowing *jorge* to run *find* and *rm* as *root*. This value can also be set to the `ALL` wildcard, which would allow *jorge* to run all commands on the system as *root*.

Show It's Working

To take advantage of his sudo role, *jorge* would use the *sudo* command on the command line followed by the program to execute:

```
jorge@ubuntu:~$ sudo find . ! -name '*.mp3' -exec rm -f \\{\\} \\;
```

If *jorge* tried to run a command other than *find* or *rm*, *sudo* would fail with a warning that he is not allowed to run that command as *root*.

You can use *sudo* to run commands as users other than *root*. If you don't specify a user, *sudo* defaults to the *root* user, but you can use the `-u` flag to designate a particular user to run as:

```
$ sudo -u fred ls /home/fred
```

As you can see, these rules allow you to create specific roles on a system. For instance, you might want to designate a group of administrators as account administrators. You don't want these users to have full *root* access, but you do want them to be able to add and remove users from the system. You could create a group on the system called *accounts* and add these users to that group (see "Manage Users and Groups" [Hack #75] for more information on user and group management under Ubuntu). Then you could use *visudo* to add the following line to */etc/sudoers*:

```
%accounts  ALL=(root) /usr/sbin/useradd, /usr/sbin/userdel, /usr/sbin/usermod
```

Now any member of the *accounts* group can run *useradd*, *userdel*, and *usermod*. If you found there were other tools this role needed to access, you could simply add them to the end of the list.

Notice that when I define particular commands a user can run, I list the full path to the command. This is for security reasons. If, for instance, I just put *useradd* instead of */usr/sbin/useradd*, the user could create her own script called *useradd* that did whatever she wanted and put it in her local path. Then she would be able to essentially run any command she wanted as *root* via that local *useradd* script.

Another handy feature of *sudo* is the ability to specify commands that don't require a password to run. This is useful if you need to run certain commands as *root* within a script noninteractively. For instance, you may want a user to be able to run the *kill* command as *root* without requiring a password (so the user can quickly kill a runaway process, for instance). To enable this privilege, add the `NOPASSWD:` attribute before the command list. To grant this ability to my *jorge* user, I would add the following line to */etc/sudoers*:

```
jorge   ALL=(root) NOPASSWD: /bin/kill, /usr/bin/killall
```

Then *jorge* could run:

```
jorge@ubuntu:~$ sudo killall rm
```

to quickly kill a runaway *rm* process as *root*.

Enable the root Account

So *sudo* is all well and good, but what if you just want to go back to an enabled *root* account that you access with *su*? Essentially, all you need to do is set the *root* password:

```
$ sudo passwd root
```

Now you should be able to log in directly as *root*, as well as use *su*.

sudo is an incredibly powerful tool, and this hack covers only some of the configuration that might be useful to a desktop user. If you want to deploy *sudo* across an enterprise, check out the *sudoers* manpage (`man 5 sudoers`) to see examples of how to configure a number of aliases to define user groups, groups of users to run commands as, host aliases, and command aliases. This modular approach to defining roles really comes in handy when deploying across a large number of computers.

Hack 68. Manage Security Updates

Stay up-to-date with the latest security fixes.

There is an old saying that the only safe computer is one that's disconnected from the network, turned off, and locked in an underground bunker—and even then you can't be sure! Keeping your computer up-to-date with the latest security patches is essential if you want to keep yourself safe from the latest threats and exploits.

Ubuntu Update Policy

When each version of Ubuntu is released, all packages within it are considered "frozen." No new versions of software contained in that release are added to it, so when you install Ubuntu Dapper Drake, the versions of all the software available within that release will remain the same indefinitely. New versions of individual packages are not added because that would make the release a moving target rather than a predictable environment—and might even introduce new bugs and security vulnerabilities.

Of course, software itself doesn't stand still; new versions are always coming out, and sometimes an existing vulnerability is found and fixed by a new release. That means older versions of the software may still be vulnerable, but Ubuntu policy dictates that new versions of software are not shoe-horned into an already-released distribution.

This impasse is resolved by back-porting security fixes to the version of the software that was included with the distribution at the time of release and then releasing a "security update" package for just that particular piece of software. System administrators can then install the security update, safe in the knowledge that they are fixing only a specific security problem and not changing the fundamental way the system operates.

Obtain Security Updates

Security updates are distributed from special package repositories, so check your */etc/apt/sources.list* to make sure you have entries that match your main package sources, like this:

```
deb http://archive.ubuntu.com/ubuntu dapper main restricted universe multiverse
deb http://security.ubuntu.com/ubuntu dapper-security main restricted
```

If not, see "Modify the List of Package Repositories" [Hack #60] to learn how to enable security sources, but note that there are no security updates available for the *universe* and *multiverse* sources because they are not officially supported by the Ubuntu security team.

Automate Update Notifications

The Ubuntu desktop comes with a panel applet called *update-notifier* that pops up an alert if any new packages have been released since your machine was last updated, but if you are running headless servers, you can set up a trivial script to notify you via email directly from the server.

Place the following in */etc/cron.daily/notify-updates*:

```
#!/bin/sh
apt-get -qq update
apt-get -qq --simulate dist-upgrade
```

Then make it executible:

```
$ sudo chmod +x /etc/cron.daily/notify-updates
```

The `-qq` flags tell *apt-get* to run in "really quiet" mode so it produces no output unless it really needs to, so `apt-get -qq update` causes it to silently fetch the latest list of packages from the package servers. The `--simulate` flag causes the *dist-upgrade* to be done in a dry-run mode that pretends to upgrade all available packages without really doing it, and if there are no packages available for upgrade, it will also complete silently.

Because this script is being called by cron, it will be triggered every day, and if it produces no output, cron will just silently move on. However, if there are packages available for upgrade, the *dist-upgrade* command will generate output listing all packages that can be upgraded, and cron will email the output to the system administrator, who can then decide whether to apply the updates manually or not.

If cron does not send email to the right person, you may need to edit */etc/crontab* and put in an entry near the top similar to:

```
MAILTO=user@example.com
```

Package Signatures

Packages distributed through the official Ubuntu archives are cryptographically signed so that you can verify that the packages haven't been tampered with or forged using a man-in-the-middle attack. The official archive keys are included in the *ubuntu-keyring* package and installed in */etc/apt/trusted.gpg* by default as part of Dapper. You can use the *apt-keys* tool to verify and manage the keys that are trusted by your system when new packages are installed:

```
jon@jbook:~$ sudo apt-key list
/etc/apt/trusted.gpg
--------------------
pub   1024D/437D05B5 2004-09-12
uid                  Ubuntu Archive Automatic Signing Key <ftpmaster@ubuntu.com>
sub   2048g/79164387 2004-09-12

pub   1024D/FBB75451 2004-12-30
uid                  Ubuntu CD Image Automatic Signing Key <cdimage@ubuntu.com>
```

If you attempt to install any packages that originated in archives that aren't verified by one of those trusted keys, *apt* will complain but allow you to proceed anyway if you choose:

```
WARNING: The following packages cannot be authenticated!
  myprogram lib-blah lib-foo
Install these packages without verification [y/N]?
```

Monitor Security Advisories

One of the most important sources of up-to-the-minute information on threats and vulnerabilities is CERT, the Computer Emergency Response Team, run by the Carnegie Mellon University's Software Engineering Institute. The CERT Coordination Center (CERT/CC) acts as a global clearing-house for advisories relating to computer security, and even makes its advisories available as RSS and Atom news feeds so you can stay on top of the latest problems as they come to light.

However, the CERT advisory listing can be overwhelming because it includes notifications for all operating systems and software packages. A much more concise list of advisories that relate directly to Ubuntu is available online at http://www.ubuntu.com/usn, along with links to the Ubuntu Security Announcements mailing list and list archives.

If you believe you have found an unreported vulnerability in a Ubuntu package, you can contact the Ubuntu security team via email at security@ubuntu.com.

Hack 69. Protect Your Network with a Firewall

Protect your network with a firewall managed from your Ubuntu desktop.

Linux has an excellent kernel-based network packet-management system called *iptables* that can be configured either directly from the command line or through a variety of GUI administration interfaces. One of the most powerful firewall management interfaces is called Firewall Builder, a system designed to separate policy from implementation and allow you to concentrate on what you want your firewall to do, rather than how you want it to do it.

The Firewall Builder interface presents hosts, routers, firewalls, networks, and protocols as objects, and allows you to drag and drop those objects to define your firewall policy. Firewall Builder then compiles your policy into the actual rules needed to enforce it, with multiple policy compilers available to suit different types of firewall. You can define your policy using Firewall Builder running on an Ubuntu desktop and then have it compiled for a firewall running *iptables* on Linux, *ipfilter* on BSD, or any of about half a dozen other firewall technologies. The policy can be defined exactly the same way, regardless of the technology deployed on the target firewall. And because Firewall Builder can support multiple firewalls simultaneously, you can use it as a central management console to configure a variety of firewalls and individual hosts throughout your network, all from a single, unified interface.

You can run Firewall Builder directly on your firewall if you choose, but as a general policy, it's a good idea to have your firewall running the absolute minimum system possible, so a better approach is to have a dedicated machine as your firewall and run Firewall Builder on a desktop or laptop management machine. Then whenever you want to update your firewall policy, you can run Firewall Builder on your management machine to generate new rules and push them out to the firewall.

Initial Firewall Setup

Start by setting up your firewall machine with a minimal Ubuntu installation: run the installer in server mode [Hack #93] so that it installs only basic packages, and preferably install at least one extra Ethernet card so that you can keep untrusted Internet traffic away from your internal network. A standard approach is to run three network interfaces on a firewall: one for your internal network (*downstream*), one to connect to the Internet (*upstream*), and one to a separate local network called the De-Militarized Zone (*DMZ*), where you can put servers that you want to be exposed to the Internet. Configure the network interfaces to suit the networks they are connecting to and make sure that your firewall can connect to each one of them individually by using *ping* to check whether you can see hosts on each network.

Your firewall machine is now sitting at the crossroads between the Internet, your internal network, and any servers that you want to run, but it doesn't yet know how to pass data from one to another so everything will be effectively isolated. To enable your firewall to pass packets through from one network interface to another and perform packet filtering and network/port address translation, you will need to install *iptables*, and to allow the firewall to be managed remotely, you will need to install an SSH server:

```
$ sudo apt-get install iptables ssh
```

Initial Management-Machine Setup

Install Firewall Builder on your management machine along with RCS and the Firewall Builder documentation package:

```
$ sudo apt-get install fwbuilder rcs fwbuilder-doc
```

Now you're ready to perform the rest of the steps.

Create a firewall project

First, launch Firewall Builder:

```
$ fwbuilder
```

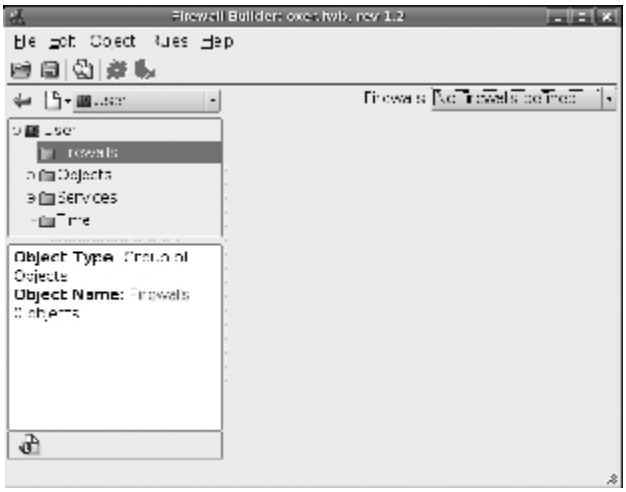
Select Create New Project File and specify a location to save it. Ideally, you should create a special directory to hold the project file because Firewall Builder will also generate other files in the same directory, and having them all in one place makes it easier to back up your firewall configuration.

You are then given the option of activating revision control for the project as well as setting it to be opened automatically when Firewall Builder starts up. Turn on both options.

The revision-control option tells Firewall Builder to store the configuration file in RCS, allowing you to see the entire history of the file, including all changes that have ever been made to it. This feature can be extremely handy if you manage to break your firewall and need to roll back to a known-good working configuration.

Firewall Builder will initially start with an empty configuration containing only a number of predefined services in two libraries (see Figure 7-1). The libraries are called User and Standard, and you can switch between them using the drop-down menu near the upper left. The Standard library is a read-only library that ships with Firewall Builder and contains predefined services for almost every TCP and UDP service in common use, along with predefined network ranges and time ranges. The User library is where objects you define will be stored, including firewalls, custom TCP and UDP services, and custom network ranges.

Figure 7-1. Firewall Builder policy management

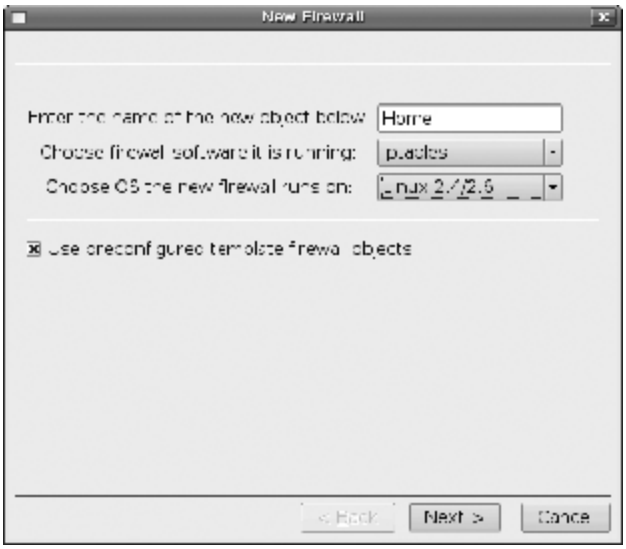


Define a new firewall

Make sure the User library is selected, right-click on the Firewalls folder, and select New Firewall. The New Firewall dialog appears, as shown in Figure 7-2. Give your firewall a name, select the firewall software (typically "iptables" if you run Linux on your firewall), and select the firewall operating system (Linux 2.4/2.6). This is where you start to see the flexibility of Firewall Builder and its support for multiple firewall types.

You also have the option of using preconfigured template firewall objects, which is a good idea if you're just getting started with Firewall Builder. The templates make it very easy to get started with a typical firewall scenario, rather than starting from scratch with a totally blank configuration. After you've made your selection, click Next.

Figure 7-2. New firewall



Click on each of the firewall template names to see a diagram and a brief explanation of how it works. For most small networks, “fw template 1” is a good choice, giving you a typical firewall with a dynamic external address, static internal addresses on the 192.168.1.0/24 network, unrestricted outbound access from the network, and access to the firewall itself only via SSH from inside the network. “fw template 2” is similar but also allows the firewall to operate as a DHCP and DNS server for the internal network.

Once you have selected a template, you will be returned to the main Firewall Builder policy-management screen, but you will now have a default policy defined for your new firewall. Firewall Builder displays policy rules in a list, with the rules applied in order, starting at the top and working down until a match is found, as shown in Figure 7-3.

Figure 7-3. Policy management

There are multiple rules lists that can be accessed using tabs across the top of the list. The primary list is Policy, which are the rules that control allowed and disallowed activity throughout the firewall. Then there is an individual rule list for each interface on the firewall, and finally a NAT list that allows you to configure Network Address Translation rules. You don’t usually need to worry about the individual interfaces; most changes will be to Policy and NAT.

Add a host-specific policy

To understand how Policy and NAT are managed, try applying a specific scenario, such as providing external access to an internal web server.

Start by adding a Host object for the server. Go through the User object tree to User→Objects→Hosts, right-click on Hosts, and select New Host. Enter a name such as *www.example.com* for your host, select the checkbox labeled Use Preconfigured Template Host Objects, and then click Next.

Select the “PC with 1 interface” template and click Finish. An object will be added to your Hosts list with an interface predefined, so click through to User→Objects→Hosts→*www.example.com*→eth0→*www.example.com*, then double-click on it to edit the interface values. Change the IP address to match the actual internal address of your server, apply the changes, and close the dialog.

Assuming you have a range of public, static IP addresses assigned to the external interface of your network, right-click on “eth0” and select Add IP Address. Enter the public IP address and network mask, apply changes, and close.

Your server now has two IP addresses defined: the real address assigned to its interface and the public address that you want people to use to access it.

Now click the NAT tab at the top of the rules list. Add a new, blank NAT rule by either right-clicking on the number of an existing rule or selecting Rules→Add Rule Below from the menu.

You can now build your NAT rule by dragging and dropping object icons into appropriate places. The columns are:

Original Src

The original source address of packets before translation

Original Dst

The original destination of packets before translation

Original Srv

The original service (port) the packets arrived on

Translated Src

The new source address to report that packets came from

Translated Dst

The new destination address to apply to packets

Translated Srv

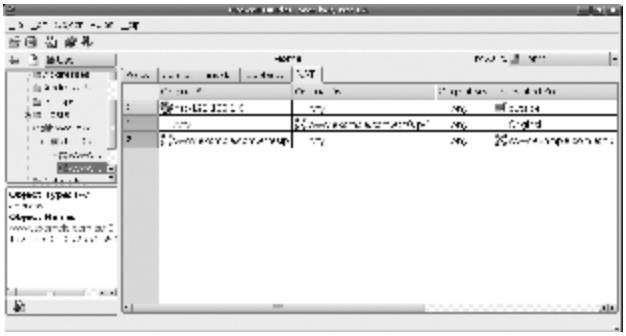
The new port to direct packets to

The first rule to set up is the translation to apply to packets directed toward the server from outside the firewall. Click the icon representing the external IP address of the server and drag it into the Original Dst box, and then click the internal IP address icon and drag it into the Translated Dst box.

Now add another blank rule and set it up to translate packets directed from the server to the outside world. Click the internal IP address icon and drag it into the Original Src box, and the external IP address icon and drag it into the Translated Src box.

You now have rules that will cause packets traveling to and from the server to be modified as they pass through the firewall, with external machines seeing only the external apply address (see Figure 7-4).

Figure 7-4. NAT rules



In these examples, only the IP addresses were translated. However, you can also apply translations to services. For example, you may have a reverse-proxy cache running as a web-server accelerator internally on port 8080, and you want external users to be able to access it using the standard HTTP port, port 80. By dragging and dropping services into the Original Srv and Translated Srv boxes from the Standard library (and creating services in the User library as necessary) you can apply port translation in the same way as address translation. Firewall Builder gives you the flexibility to convert the source and destination addresses and port of packets at will, so adjust the NAT rules until they represent the transformations you want to apply to network traffic.

However, NAT rules alone are not enough. Without a matching policy rule, no packets will be allowed through the firewall, even if they match NAT rules. While the NAT rules define what *can* happen, policy rules define what is *allowed* to happen. Policy rules have the final say.

Click the Policy tab and go down through the list of existing rules to find an appropriate place to add rules for your host, most likely right near the end, before the general network rule and the fall-through Deny All rule. Right-click in the rule-number column and add a rule. Drag the external IP address icon of your web server into the Destination box, and then right-click the Deny icon in the Action column and change it to Allow.

If you want to allow full access to every port on your web server from outside your network, that's all you need to do. However, it's safest to make the rule more specific and allow only certain services through so click the User/Standard drop-down near the top left and switch to the Standard library. Browse down through the library to Standard→Services→TCP→"http" and drag its icon into the Service box of your new rule. If you want to allow SSL connections, you can also drag the "https" icon into the same box. You now have a rule that explicitly allows connections from any host to your web server but only on ports 80 and 443 (see rule 5 in Figure 7-5).

Figure 7-5. Policy rules

Compile and Install the Policy

Once you are happy with the policy you have created, you need to apply it to the firewall. Firewall Builder does this in two steps: first the policy is compiled into a script to suit the software on your firewall; then it's pushed out to the firewall and loaded.

Select Rules→Compile or click the gear icon to have Firewall Builder compile your rules. You will see a dialog reporting progress, and when it finishes, you will have a new script placed in Firewall Builder's working directory alongside the project file. You can now copy the script to the target firewall by SSH and execute it to have the rules applied. To keep everything neat, it's a good idea to create a directory on the firewall to store the script:

```
$ sudo mkdir /etc/firewall
```

Then copy your script into that directory and execute it manually to test it. Even though it will have a *.fw* extension, the file is actually just a shell script that you can run in the normal way:

```
$ sudo /etc/firewall/
                    firewallname.fw
```

Firewall Builder provides a large number of configuration options to control the way the script is generated for each firewall, so if the script didn't work properly on your firewall, you may need to right-click the firewall icon in the object tree, select Edit, and then click the Firewall Settings button. The tabs at the top give you access to a lot of options, so go through them carefully and change any settings that may apply to your particular firewall; then compile the rules and test them again.

Automatic Policy Startup

The firewall script needs to be run each time your firewall boots up, so use your favorite editor to open */etc/rc.local* and add the path to the script just before the `exit 0` line. The end of */etc/rc.local* should look something like this:

```
/etc/firewall/firewallname.fw
exit 0
```

The *rc.local* file is executed after all the other startup scripts whenever Ubuntu switches to a new multiuser runlevel. Referencing your script in there ensures it will be executed after other services such as networking have started up, and that it won't be started if you boot up in single-user mode. This is handy if you need to fix configuration problems by booting into single-user mode.

Automatic Policy Installation

Once you are happy that the rules are being generated correctly for your firewall, you can save yourself some effort on subsequent updates by configuring Firewall Builder to manage installing and activating the script on your behalf. Select the firewall icon (located in User→Firewalls), right-click, select Edit, and the Firewall dialog will appear. Click Firewall Settings to re-enter the firewall options dialog. Click the Installer tab to be presented with options to execute a script to install and activate the firewall rules. The lower section of the dialog provides two text-entry fields that you can use to invoke any external script or command you like; so, for example, you can write a script that copies the script to the firewall by SCP and then executes it using SSH. Firewall Builder even comes with an sample script to do exactly what you will find installed in */usr/bin/fwb_install*. Full information on how to use *fwb_install* is available in its manpage:

```
$ man fwb_install
```

Firewall Builder also has an internal policy-installation mechanism, which is perfectly adequate for most environments.

To set up automatic policy installation, first create a group such as *fwadmin* on the firewall, and then create a user and make it a member of the group:

```
$ sudo addgroup fwadmin
$ sudo adduser fwadmin -G fwadmin
```

Set up a directory on the firewall to store the firewall configuration:

```
$ sudo mkdir -m 0770 /etc/firewall
$ sudo chown fwadmin:fwadmin /etc/firewall
```

Configure *sudo* to allow this user to execute the firewall script without entering a password by running:

```
$ sudo visudo
```

and adding a line similar to the following to the end of the */etc/sudoers* file:

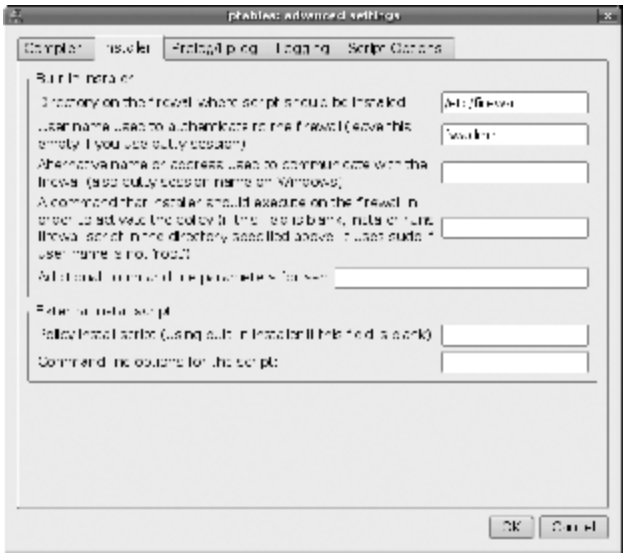
```
%fwadmin = NOPASSWD:/etc/firewall/firewallname.fw
```

The `firewallname.fw` string should be replaced by the actual name of the script generated by Firewall Builder for this firewall.

You can optionally set up public-key encryption for access to the *fwadmin* account on the firewall.

In the Installer tab of the Firewall Settings dialog, put in the path to the directory you created on the firewall and the username you created, as shown in Figure 7-6.

Figure 7-6. Policy installation options



You may not need to specify an alternate name or address for the firewall: Firewall Builder will try to automatically determine the IP address to use to communicate with the firewall, but it will use an address if you put one in.

Save and close the Firewall Settings dialog, and you should now be able to install and activate your policy simply by clicking the Install icon or by selecting Rules→Install from the menu. From now on any updates you make can be applied simply by clicking Compile and then Install.

Extensive documentation and tutorials are available on the official Firewall Builder site at http://www.fwbuilder.org if you want to learn more.

Hack 70. Use an Encrypted Filesystem to Protect Your Data

An encrypted filesystem keeps your data safe even if someone steals your computer and tries to mount the disk.

There are a number of techniques for creating encrypted filesystems, typically based on using some kind of loopback device with an encryption layer spliced in the middle. Popular systems include *cryptloop* and *loop-aes*, but a more recent development called *dm-crypt* has some interesting advantages, so that's what I use for this hack. It's faster than *cryptloop*, easier to implement than *loop-aes*, and able to operate on a wide range of block devices even when using RAID or LVM because it's built on a new feature in the 2.6 kernel called *device-mapper*. *device-mapper* is designed to provide a generic and flexible way to add extra virtual layers on top of real block devices, allowing developers to implement

special handlers for mirroring, snapshotting, concatenation, and encryption. As far as filesystem tools are concerned, they are dealing with a real block device, and any special functionality is hidden away by *device-mapper*.

dm-crypt is a *device-mapper* target that uses the kernel crypto API to provide transparent encryption, and it's even backward-compatible with the on-disk format used by *cryptloop*.

Kernel Setup

dm-crypt uses the kernel's crypto API to perform the actual encryption. The standard Dapper kernel loads crypto ciphers as modules, and this hack uses 256-bit AES encryption, so make sure that your kernel has AES support loaded:

```
$ cat /proc/crypto
```

256-bit AES is an extremely high-grade encryption algorithm that has been approved by the NSA for use in protecting classified information up to the TOP SECRET level, which is the highest security level, encompassing information that would cause exceptionally grave damage to U.S. national security if disclosed.

If AES support is loaded, you will see output like:

```
name         : aes
module       : aes
type         : cipher
blocksize    : 16
min keysize  : 16
max keysize  : 32
```

If it's not loaded, you can force it to load manually using *modprobe*:

```
$ sudo modprobe aes
```

Cryptsetup should load this module automatically when required, but it's useful to know how to check for it and load the module manually if necessary.

Install the *dmsetup* package, which will provide the tools you need to configure the *device-mapper* target:

```
$ sudo apt-get install dmsetup cryptsetup
```

Check that the *dmsetup* package has created the device mapper:

```
$ ls -l /dev/mapper/control
```

Load the *dm-crypt* kernel module:

```
$ sudo modprobe dm-crypt
```

The *dm-crypt* module registers itself automatically with *device-mapper* when it's loaded, so next check that d*evice-mapper* has recognized it and added `crypt` as an available target:

```
$ sudo dmsetup targets
```

If everything has gone to plan, you'll see `crypt` in the output:

```
crypt            v1.1.0
striped          v1.0.2
linear           v1.0.1
error            v1.0.1
```

Your system is now ready to mount encrypted devices. First, though, you need to create one!

Create an Encrypted Device

There are two options for creating a filesystem to mount as an encrypted device: create a disk image as a file and mount it as a loopback device, or use an actual block device (for example, */dev/hda3*). In both cases, the procedure is almost identical, with the exception of some extra preparation required to create and bind the loopback device.

Create a loopback disk image

If you don't have a real device, such as a memory stick or extra disk partition, that you can encrypt, you can instead use *dd* to create an empty disk image and mount it as a loopback device. In this example I call it *secret.img* and make it 100 MB in size. Just alter the `count` value if you want to make it a different size:

```
$ dd if=/dev/zero of=~/secret.img bs=1M count=
                100
```

Use *losetup* to associate the image with a loopback device:

```
$ sudo losetup /dev/loop/0 ~/secret.img
```

You will now have a virtual block device at */dev/loop/0* that you can reference just like any other block device.

Set up the block device

Now you can set up either a real block device (for example, */dev/sda1*) or a virtual block device, such as the loopback image created in the previous step, and mount it as an encrypted logical volume using *device-mapper*. Start by using *cryptsetup* to create a logical volume (I'll call it *mysecrets*) and bind the block device to it:

```
$ sudo cryptsetup -y create mysecrets
                /dev/DEVICENAME
```

The last argument must be the block device that will be used as an encrypted volume, so if you're using the loopback image created in the previous step as a virtual block device, you would instead run something like:

```
$ sudo cryptsetup -y create mysecrets /dev/loop/0
```

In either case, you will be asked for a passphrase for the logical volume, and the `-y` flag indicates that *cryptsetup* should confirm the passphrase by asking you for it twice. It's critical that you get it right, because otherwise you'll find yourself locked out of your own data!

You can then confirm that the logical volume has been created:

```
$ sudo dmsetup ls
```

You should see the logical volume listed, although the major and minor device numbers may vary:

```
mysecrets (254, 0)
```

device-mapper mounts its virtual devices under */dev/mapper*, so you should now have a virtual block device at */dev/mapper/mysecrets* that behaves just like any other block device but is transparently encrypted.

Just like a real block device, you can create a filesystem on it:

```
$ sudo mkfs.ext3 /dev/mapper/mysecrets
```

Create a mount point for the new virtual block device and mount it:

```
$ sudo mkdir /mnt/mysecrets
$ sudo mount /dev/mapper/mysecrets /mnt/mysecrets
```

You should now be able to see it mounted just like any other device:

```
$ df -h /mnt/mysecrets
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/mysecrets    97M  4.1M   88M   5% /mnt/mysecrets
```

Voil\x88 ! You now have a mounted filesystem that behaves just like any other filesystem, but any data you write to */mnt/mysecrets/* will be transparently encrypted before being written to disk, and anything you read from it will be decrypted on the fly.

Unmounting

Unmount the encrypted filesystem as usual:

```
$ sudo umount /mnt/mysecrets
```

Even after you have unmounted the block device, it's still configured in *dm-crypt* as a virtual device. You can confirm this by running `sudo dmsetup ls` again and seeing that the device is still listed. Because *dm-crypt* caches the passphrase, any other user on the computer can now remount the device without needing to know the passphrase. To prevent this, you need to explicitly remove the device from *dm-crypt* after unmounting it:

```
$ sudo cryptsetup remove mysecrets
```

After that, it will be truly gone, and remounting it will require the passphrase again.

You can simplify the process by creating a tiny script to unmount and remove the device:

```
#!/bin/sh
umount /mnt/mysecrets
cryptsetup remove mysecrets
```

Remounting

Since you'll probably want to mount the encrypted device as your normal user, it will make things easier if you add something like this to your */etc/fstab*:

```
/dev/mapper/mysecrets /mnt/mysecrets ext3 noauto,noatime 0 0
```

You can also create a script that will take care of creating the *dm-crypt* device and mounting the volume for you (replace */dev/DEVICENAME* with the actual device name or path to the file):

```
#!/bin/sh
cryptsetup create mysecrets /dev/DEVICENAME
mount /dev/mapper/mysecrets /mnt/mysecrets
```

If you're using a loopback device, you can also have the script bind the device for you:

```
#!/bin/sh
losetup /dev/loop/0 ~/secret.img
cryptsetup create mysecrets /dev/loop/0
mount /dev/mapper/mysecrets /mnt/mysecrets
```

If you get the message "ioctl: LOOP_SET_FD: Device or resource busy", it probably means that the loop device is still mounted. You can remove it with `sudo losetup -d /dev/loop/0`.

Hacking the Hack

You can even have your entire home directory encrypted if you configure the PAM (Pluggable Authentication Modules) subsystem to mount it for you when you log in. The *libpam-mount* module allows PAM to mount arbitrary devices automatically when a user logs in, so install it along with *openssl*:

```
$ sudo apt-get install libpam-mount openssl
```

Edit */etc/pam.d/common-auth* and add this line to the end:

```
auth    optional        pam_mount.so use_first_pass
```

Then edit */etc/pam.d/common-session* and add this line to the end:

```
session optional        pam_mount.so
```

You need to configure PAM to know what volume to mount and where. In this example the username is *jon* and I'm using the device */dev/sda1*, so substitute your username and device and add a line like this to */etc/security/pam_mount.conf*:

```
volume jon crypt - /dev/sda1 /home/jon cipher=aes aes-256-ecb /home/jon.key
```

If you want to use a disk image, you need to specify the loop device (such as */dev/loop/0*) here and also ensure that the system has run *losetup* before *jon* has a chance to log in (for example, you could put `losetup /dev/loop/0 /home/secret.img` into */etc/rc.local*).

Because the volume is encrypted, PAM needs access to the key to mount it. The last argument tells PAM to look in */home/jon.key* for the key, so create the key file by encrypting your passphrase using OpenSSL:

```
$ sudo sh -c "echo
            '
                YOUR PASSPHRASE
            '
            | openssl aes-256-ecb >
                /home/jon.key"
```

You will then be asked for a password, and you must enter the same password as you use to log in as that user. The reason is that when you log in, PAM will take the password you provide, use it to decrypt the key file, and then use the passphrase contained in the key file to mount your home directory using *dm-crypt*.

Note however that the step above will leave your passphrase in plain text in your *.history* file, so clear your history (with `history -c`) or at least edit it to remove the command before proceeding.

Another approach that avoids storing your passphrase in an encrypted key file is to create your encrypted filesystem with the exact same password as you use to log in. Then PAM can simply pass your password through to *dm-crypt* when you authenticate rather than extract it from the key file. In that situation you can use a line like this in */etc/security/pam_mount.conf* instead:

```
volume jon crypt - /dev/sda1 /home/jon cipher=aes - -
```

Finally, to make sure your encrypted home directory is unmounted automatically when you log out, edit */etc/login.defs* and make sure the `CLOSE_SESSIONS` option is set:

```
CLOSE_SESSIONS yes
```

Hack 71. Encrypt Your Email and Important Files

It's good to be paranoid, and the best way to protect your thoughts and data is with powerful encryption.

Encryption is a vital tool with which all Ubuntu users should familiarize themselves. One of the best ways to encrypt emails and files is with GPG, the GNU Privacy Guard. Like its inspiration PGP, GPG utilizes *public-key cryptography*. In essence, each user owns two keys: a *private key* used to decrypt messages sent to the user and a *public key* others use to encrypt messages they send to that user. In order for this system to work, therefore, folks who want to send encrypted messages to each other must exchange public—never private!—keys.

Once you have GPG up and running, there are several things you can do with it:

- Encrypt and decrypt email messages and attachments.
- Encrypt and decrypt files.
- Sign a file with your electronic signature or verify the authenticity of a file by checking its digital signature.
- Verify or sign other users' public keys.

As I said in the first paragraph, you need two keys, but where do they come from? If you've been using GPG for a while, you can move your keys from machine to machine, as needed, or you can create a new key pair from scratch.

There are GUIs for virtually all of the commands I'm going to show you in this hack. For KDE, KGpg is an excellent tool (from the package *kgpg*). GNOME users should check out Seahorse (from the package *seahorse*, and shown in GNOME menus as Encryption Key Manager).

Generate a GPG Key Pair

It's a simple matter to create your own GPG key pair (if you already have a key pair on another computer, skip to the next section):

```
$ gpg --gen-key
```

You'll be asked to choose the kind of key to use, so choose from the following items...and no, I don't know why the third and fourth options are left out:

```
1. DSA and ElGamal (default)
2. DSA (sign only)
5. RSA (sign only)
```

Press Enter to choose the default of `1`, and next you're asked about your key size. The default of `2048` is excellent, so just press Enter to accept it. Now you're asked about how long you want this key to be valid. The default is `0`, meaning that it will never expire, and normally this is just fine, as long as you plan to keep this key and use it going forward. If you instead want it only for a limited time, change it to the number of days, weeks, months, or years you want, and press Enter. After verifying that length of time, type `y` and press Enter again.

Now you need to create a user ID that is linked to the keys. First you're asked your real name, so type it in and press Enter. Following that, you're asked for an email address, and finally a comment. The comment can be anything at all, such as a URL, a company name, a location, or even nothing at all, but it's a good idea to include this data point to help distinguish between keys. Type your info in, pressing Enter after each item. You'll be asked to confirm or change your user ID when you've answered the three questions—name, email, comment—so respond accordingly and press Enter.

Now for the biggie: your secret key's passphrase. As you can probably guess, you need something really good here, not just *password* or *scooter* or *123456*. Notice that you're able to use a passphrase, not just a password, so pick something long with spaces that is still memorable to you. Stuck? Check out an article I wrote for SecurityFocus that contains tips for picking good passwords and passphrases: "Pass the Chocolate," at http://www.securityfocus.com/columnists/245. Enter your passphrase, then verify it, and then, boom! GPG begins generating your keys. You can tell because random characters will appear on your screen, and GPG will ask you to move your mouse, type on your keyboard, and generate disk activity to help improve the key by providing random data it can use. Finally, GPG will finish creating your new keys, which you can verify with this command:

```
$ gpg --list-keys
/home/scott/.gnupg/pubring.gpg
------------------------------
pub 1024D/73CA5DE6 2006-03-20
uid                  Scott Granneman (St. Louis, MO) <scott@granneman.com>
sub   2048g/675B19A0 2006-03-20
```

You've created your keys; now it's time to use them. Skip ahead to "Signing a GPG Key," or read the next section if you want to learn how to import GPG keys from another computer.

Importing GPG Keys

If you want to use the exact same public and private keys from another computer, just copy them over from the other machine to your new box. Realize that you'll overwrite anything already on the new computer, if you've created any keys at all, but this may be just fine. I've used the same keys for years, simply copying them from machine to machine, and it's worked fine. Here's how to copy keys from an old box named *eliot* using *scp*.

```
$ mkdir ~/.gnupg
$ scp
                    eliot
                :~/.gnupg/* ~/.gnupg
```

You can also import keys, which will append them onto a computer's currently existing keyring (rather than overwriting existing keys, as in the previous method). To do so, you obviously need access to them. This can be accomplished by copying the keys from another machine to yours or by grabbing the keys from a public keyserver on the Net. If the keys are on another computer, copy them to your Ubuntu box, put them on the Desktop for the time being, and then run this command:

```
$ gpg --import /home/
                    username
                /Desktop/pubring.gpg
```

You'll see the list of keys that are imported, along with totals, indicating success. Don't forget to delete the *pubring.gpg* file on your Desktop, since you no longer need it.

If the keys aren't directly available to you, but you know that the users whose keys you wish to import have uploaded them to a public keyserver, you can always import them from there. For instance, say you want to import my key. First you need to find my key's ID. Using your web browser, go to the MIT PGP Public Key Server at http://pgp.mit.edu and search for *scott granneman*. You'll get back three results, but pay attention only to the one dated 2004/08/08, which looks like this:

```
Type bits /keyID    Date      User ID
pub  1024D/6503F88C 2004/08/08 Scott Granneman <scott@granneman.com>
                               Scott Granneman (www.granneman.com) <scott@granneman.com>
```

Take note of the key ID, which is *6503F88C*. With it, you can import that specific key using the following command:

```
$ gpg --keyserver pgp.mit.edu --recv-keys 6503F88C
gpg: requesting key 6503F88C from hkp server pgp.mit.edu
gpg: key 6503F88C: public key "Scott Granneman <scott@granneman.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

Was that easy or what?

Signing a GPG Key

Before you can send someone a file or message encrypted with GPG, you have to sign the key you're going to use. By signing the key, you verify that it belongs to the correct person. In the best situation, you received it directly from the individual, perhaps face to face or perhaps through email. If you know that the key you just imported belongs to Scott Granneman, you can run this command:

```
$ gpg --sign-key www.granneman.com
```

In actuality, you can use any data that uniquely identifies a key after the `--sign-key` option. I used *www.granneman.com* because it was unique to the key you imported previously. To see the data points you can use, just run `gpg --list-keys`. For instance, the key ID—*6503F88C*—would have been a great choice, since it's obviously unique to that exact key.

When you sign a key, you're asked if you want to sign all user IDs associated with it. Type in `y` and press Enter. Verify your answer by entering `y` again and press Enter. Now you're asked for your secret passphrase—the one protecting your secret key—so that you can prove it's really you who's signing this person's public key. Enter your passphrase, press Enter, and you're finished. You've signed the key, and you can now start using it for encryption.

Encrypting Files Using GPG

You've signed my key, so now it's time to send me an encrypted file. Open your favorite text editor and enter a message like "Can't sleep...clowns will eat me." Save the file on the desktop as *test_encryption* and close it. In your terminal, enter this command:

```
$ gpg -r "Scott Granneman" --encrypt test_encryption
```

Please don't actually send me the files you're encrypting. This book is going to have a worldwide audience of billions of readers, and I can't possibly answer all of the test messages all of those readers are going to create, as well as respond to the thousands of fan mails I get every day. Thank you!

Now there's a new file on your desktop: *text_encryption.gpg*. This is a binary file that you could attach to an email and send. You don't have to create binary files that you attach, however; you can instead generate ASCII text messages suitable for pasting into an email. To do so, run this command:

```
$ gpg -r "Scott Granneman" --armor --encrypt test_encryption
```

Now your desktop contains a file named *test_encryption.asc*. If you open the file, you'll see something like this:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.4.2.2 (GNU/Linux)

hQIOAwcs+U2/zd/NEAf+IR5zIX/ggNxemqkoa4HhCDeEm5hiu1rJivzd7fi504Di
MfeE11Do7p6WMWBi+fzH5FtowwMXi/+OFt6erO65Cb3sVsBDyH01W1NrDLRNPTgN
HC88OJOiWpB16/w65rQXQC9Lgczc8kNsqMCMWUo3tHLBxFNg7d+SW62zgicX2qza
fqHq1Y5ekRPWhmmqkItbYhsXMwz8YE7jq38TP17RBcStiNEKlSAyRrFKMC2L5OI9
```

```
VCj2GV6gaKK8nDKFD87hw3V8TN29JHAcPZ5R11wJJNVRv+muDgX+bY8z/9RpTsMs
ALn3ET5J8kzquJg5tctrpXvSN4vBKpvjCH1Y30gDDQgAxRHuNLRo8muBh63N7twc
Gx5VJZlx9AfBYTTWqXPC0VA7O2DSbfgwUdDQS9mL+8ckR1XI921q9ih05IqHG40x
cibPCyg8kXRSK1PUxk4IXiOjbUs62OxMtqMvZtfSzbuOxoNRezrWXO79Q5N2KZVs
HtexYLKm9Wf4RhCQylSwosMojeg4eoTsz+R3MArG8XR/MbI/sLZKGhi3nuwpO3oK
YqaWUhob+U0ZLfCf1l3WPhO7nkJ+6xgcVKIXiycTBYeH6xfN5Gq+WHx546XKJbkB
fTutzRsqUdAr3TED77Pr6zPi8PBZL0f76KXE5KX05zClkwd97vCkIVaQkm4etsGc
4dJwAQ5fhUGqRlmMScO3UbtS3Hh43ZQ8Ndnn+Jpid7D4thZEub9lcpkGw3PLeOco
jRu9smNJHUd8J5ODapAOunqGt0whmpcePbrxEkbL1upigAxqFc9DV7g7pDgJsSLr
i+Bm7tUbhQq2vLpINDeVnzchNQ==
=yBG6
-----END PGP MESSAGE-----
```

You can attach the ASCII file to an email message, or you can copy and paste the entire text directly into a message.

For added security, you can digitally sign it so that the recipient knows that the file was in fact from you and you only. To sign and encrypt a file at the same time, use this command:

```
$ gpg -r "Scott Granneman" --armor --sign --encrypt test_encryption
```

You'll be asked for your passphrase, so go ahead and enter it. You'll still end up with *test_encryption.asc*, but it will be slightly larger, since it now contains your digital signature as well as your original message.

Although I've been focusing on encrypting files meant for another party, you can always encrypt files for yourself by simply specifying yourself as the recipient. It's a great way to protect files on your own computer that you don't want just anyone viewing.

Decrypting Files Using GPG

Let's say my buddy Jans sends me an encrypted file, and I need to decrypt it. Before proceeding, I must have imported Jans' public key into my keyring and verify his key. Now I can decrypt the file. If he sent me a binary file, I'd use this command:

```
$ gpg --output
                business_plans.odt
            --decrypt
                business_plans.odt.gpg
```

If he instead sent me an ASCII file, I'd use this:

```
$ gpg --output
                business_plans.odt
            --decrypt
```

```
business_plans.odt.asc
```

In either instance, I'd be prompted to enter my secret key's passphrase. Upon doing so, GPG would decrypt the file, leaving me with the filename I'd specified with the `--output` option. Without that option, GPG instead sends output to STDOUT, which wouldn't work very well with an OpenOffice.org document.

Of course, most good Linux email programs have integrated GPG encryption directly into their interfaces. KMail, the default for Kubuntu, makes it simple to work with encrypted emails and attachments, and while Evolution, the default for Ubuntu, isn't quite as easy, it's certainly doable. Really, there's no excuse for you not to use GPG to encrypt your emails and files. With absolutely no downsides and lots of good reasons—privacy and security come immediately to mind—you should set up your own keyring and start using GPG today.

For more details on GPG, check out `man gpg`, or head over to the GnuPG web site, at http://www.gnupg.org.

Scott Granneman

Hack 72. Surf the Web Anonymously

Firefox + the Tor Project = you accessing web sites without anyone knowing who you are.

It's still possible to use the Web with some degree of privacy and even anonymity, but you have to jump through a few hoops to do so. Even so, these hoops aren't that arduous, and, better yet, you'll get to learn about a cool program named Tor that makes all of it possible.

In answer to the question, "Why would I want to surf the Web anonymously?" here are a few reasons you might want to use Tor:

- You just don't want to be tracked as you browse the Web.
- Your ISP or network blocks your access to certain web sites or services.
- You want to access and participate in communications that you'd prefer were kept private, such as those for survivors of rape and abuse.
- You live in a country with an oppressive government that monitors or limits its citizens' Net usage.
- You want to connect to some of the cool hidden servers that anonymously publish materials available only to Tor users.

Of course, there are also some good reasons not to use Tor:

- You're a speed demon and can't stand the fact that connections using Tor will always be slower than nonproxied connections.
- You don't want to deal with the hassle that because some e-commerce sites track your browsing session using your IP address, they will fail miserably since Tor keeps changing it.
- You want to share BitTorrent or other P2P files—seriously, that slows down the entire Tor network, so please don't do it at this time.

With your choices laid out before you, let's look at how Tor does its job.

How Onion Routing Works

Tor works by utilizing a technology known as *onion routing*. You can find great explanations of onion routing at the Electronic Frontier Foundation (http://tor.eff.org/overview.html.en and http://tor.eff.org/documentation.html.en) or at Wikipedia (http://en.wikipedia.org/wiki/Onion_routing). Here's a quick-and-dirty analogy for onion routing. Imagine that Alice wants to send a message to Bob, who's on the other side of a building. Alice encodes her message and hands the first few characters of it to Carol, who decrypts it, re-encrypts it with a new key, and then hands it off to David. David decrypts it, re-encrypts it with a new key, and then hands if off to Ernest. Ernest repeats the process, handing the message off to Frank. Finally, Frank decrypts the message and hands it to Bob, the intended recipient of the message.

Remember, though, that this entire process was just for the first few characters of the message. The second set of characters don't go from Alice to Carol; instead, they go from Alice to Gus, and then from Gus to Ernest, Ernest to Paul, Paul to Scott, and from Scott to Bob. The third set of characters goes through a different route as well, as do all the other sets, until the message is completed. Some of the same people may be reused, but the progress is random.

Think about the security benefits of this process: Ernest knows that he got his chunk from Gus and that he sent his chunk along to Paul, but that's it. Ernest has no idea that the originator of the message was Alice and that the ultimate recipient is Bob. The encryption performed between each hop further protects the message. While the system isn't perfect (and the creators of Tor are the first to admit this), it's still a smart, powerful technology that goes a long way toward providing anonymity to Web users.

Installing and Using Tor

To use Tor, you actually need to install two programs on your box: Tor itself and Privoxy, an open source, powerful proxy that's needed to further protect your anonymity (for more on why this is necessary, see http://wiki.noreply.org/noreply/TheOnionRouter/TorFAQ#SOCKSAndDNS). In addition, there's a Firefox extension, SwitchProxy, that will come in very handy, so you'll need to put that in place as well. Once you've completed these tasks, you'll be able to use the Web in near-complete anonymity.

To begin, use *apt* to install Tor and Privoxy.

```
$ sudo apt-get install tor privoxy
```

You may find that *apt* wants to install some needed dependences, such as *libevent1* and *tsocks*. Go ahead and approve those packages, and a few moments later your new software will be installed. Ubuntu will start both Tor and Privoxy automatically as services, but you still need to configure them.

First set up Privoxy to use Tor. Using *sudo*, open your favorite text editor and point it to */etc/privoxy/config*. If you want to be precise, find section `5.2. forward-socks4 and forward-socks4a` and add the following line (be sure to include the dot at the end!):

AU: Just want to confirm: the section title is *"5.2. forward-socks4 and forward-socks4a"*? In other words, "and" is part of the section title?

```
forward-socks4a / localhost:9050 .
```

If you're in a hurry, or you're proudly sloppy, just add that line at the very top of the file.

Before saving and leaving *config*, you need to take care of one more detail. By default, Privoxy stores some items not conducive to total privacy. The program logs everywhere it goes, so if you want to disable this behavior, find section *1.5. logfile* and comment out the line for this setting, as follows:

```
# logfile logfile
```

In addition, Privoxy stores all intercepted cookies, so look for section *1.6. jarfile* and comment out the line for that setting, like this:

```
# jarfile jarfile
```

Save */etc/privoxy/config* and then restart Privoxy.

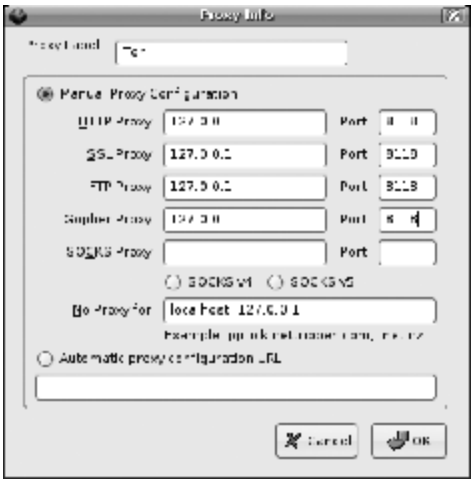
```
$ sudo /etc/init.d/privoxy restart
```

Now you need to set up Firefox to use the SwitchProxy extension. Open Firefox and head over to https://addons.mozilla.org/extensions/moreinfo.php?id=125&application=firefox (if that page is missing for some reason, search http://addons.mozilla.org for *SwitchProxy*). Click on the Install Now link on the web page, and when the Software Installation window opens, click the Install Now button that appears. Close the Extensions window and then restart Firefox. When you do, a new toolbar will appear, as shown in Figure 7-7.

Figure 7-7. After SwitchProxy is installed, it appears as a Firefox toolbar

On the Proxy toolbar, click the Add button. In the Select Proxy Type window, leave Standard chosen and click Next. In the Proxy Info window, enter `Tor` as the Proxy Label. Then, for the first four proxies (HTTP, SSL, FTP, and Gopher) enter `127.0.0.1` as the host and `8118` as the port. Your final results are pictured in Figure 7-8.

Figure 7-8. Configure SwitchProxy to use Tor



Click OK to close the Proxy Info window. Back in Firefox, choose Tor from the drop-down menu on the Proxy toolbar and click Apply. The Status on the toolbar should change from Using No Proxy to Using Tor. To make sure that Tor is working correctly, point Firefox to http://serifos.eecs.harvard.edu/cgi-bin/ipaddr.pl?tor=1, which will present a large text message about whether or not Firefox is using Tor.

Other Apps

You can use Tor as your proxy with other programs. If a program supports using an HTTP proxy, just enter in *127.0.0.1:8118*, the address for Privoxy. Other programs use SOCKS directly—like IM and IRC—and for those you use *127.0.0.1:9050*, the direct address of Tor.

In particular, you may find this implementation handy for FTP. In actuality, Privoxy works with only HTTP and HTTPS. You told Firefox to use Privoxy (and therefore Tor) for FTP and Gopher to prevent any hidden images served via FTP from giving away your IP address. Now FTP and Gopher will fail—since Privoxy doesn't work with them—but your privacy will be maintained. Find an FTP app that uses SOCKS, and you can use FTP with Tor.

For a short set of instructions that explain how to use Tor with Gaim, the excellent cross-platform IM client, see http://gentoo-wiki.com/HOWTO_Anonymity_with_Tor_and_Privoxy#How_to_use_Gaim_with_tor. On the same page are instructions for configuring *wget*, *curl*, and other terminal-based apps to use Tor; see http://gentoo-wiki.com/HOWTO_Anonymity_with_Tor_and_Privoxy#Autoconfigure_Some_Applications.

If you need to connect to a site without Tor, you can always temporarily set the SwitchProxy extension to None. Better still, with Tor acting as a proxy, click the Edit button on the SwitchProxy toolbar. When the Proxy Info window opens, add the troublesome web site to the list in the No Proxy For text box. Click OK, and now you'll browse that site proxy-free.

Tor is great software, and we should all thank the EFF for developing it. Unfortunately, the EFF doesn't have the money to pay developers to continue full-time on the project, so if you can, please consider donating a few dollars to the EFF to help fund further work. If you're able to do so, check out http://tor.eff.org/donate.

Scott Granneman

Hack 73. Keep Windows Malware off Your System

Linux users can help protect their Windows-using brethren from the myriad virus infestations out in the wild with some free anti-virus tools.

It's well known among Linux users that ours is an operating system that doesn't have to worry about viruses, unlike that *other* OS made in Washington state that's appears completely overrun by them. Even so, there are still very good open source anti-virus tools available to run on Linux, principally among them *ClamAV*. But why do people running Ubuntu need anti-virus software?

If you receive infected materials that may be sent along to Windows users, it sure would be good to prevent the transfer of virus-laden files. If you share files via Samba with people on Windows, you probably want to know if some of those are infected, just as a point of pride if nothing else. And, finally, a huge plague of Linux viruses could be unleashed upon the world sometime in the future, so it's good to be prepared (sure, it's about as likely as the dead arising from their graves to eat the brains of the living, but you never know).

And besides, it doesn't really hurt to run ClamAV on your system. The resources it takes up are miniscule, and the vast majority of the time, you'll never even know it's there. So why not?

To install ClamAV, run the following command (you'll need the universe repository enabled [Hack #60]):

```
$ sudo apt-get install clamav
```

This will prompt *apt* to download some dependencies required by *clamav*, including *clamav-base* and *clamav-freshclam*. Go ahead and accept them, and let *apt* download and install the software. You may find that configuration problems occur due to one of the packages needing to be configured before another can be configured, so you may have to run the following command to straighten things out:

```
$ sudo apt-get -f install
```

You should see the various *clamav* packages successfully set up, and then *apt* will start *freshclam* for you automatically.

You've installed the base packages, but you haven't installed a GUI yet. ClamAV doesn't need a GUI—the whole thing can be run from the command line, and in fact, that's best for scripting—but if you want a GUI, you have a choice. If you use Kubuntu, you can install *klamav*, a KDE-based front end for ClamAV; if you use Ubuntu, *avscan*, a GTK-based frontend is what you want. Or you can install both, as it really doesn't hurt anything. I'll install both, even though I'm basically a Kubuntu user:

```
$ sudo apt-get install avscan klamav
```

A lot of dependencies are required for these two packages, so go ahead and accept them. I'm going to focus the rest of this chapter on the command line, but here's a quick glimpse of the frontends. Ubuntu's GUI, AntiVirus Scanner, is shown in Figure 7-9. Kubuntu's GUI, KlamAV, is shown in Figure 7-10.

Figure 7-9. Ubuntu's virus scanner

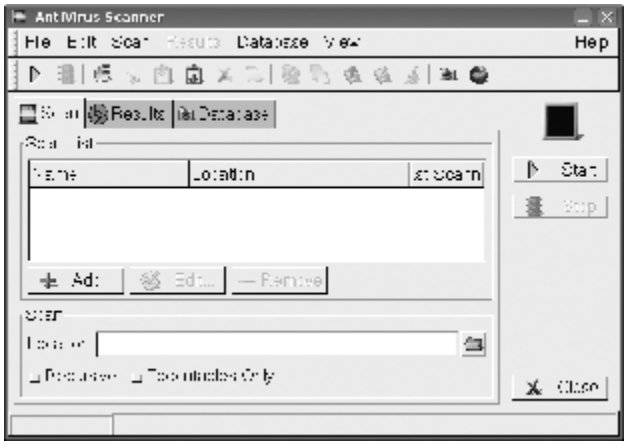
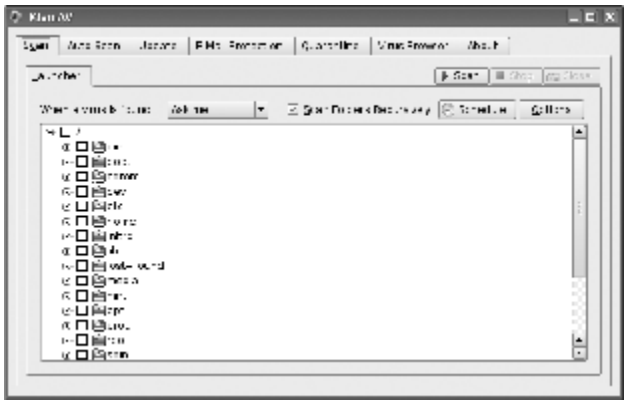


Figure 7-10. Kubuntu's virus scanner



By the way: even if you're using Ubuntu, you can still run KlamAV if you have the KDE libraries installed. I recommend using KlamAV over AntiVirus Scanner if at all possible, since it's a much nicer program.

Now, back to the command line. To check for regular virus-definition updates—one of the key features of an anti-virus program—ClamAV uses *freshclam*. The config file that handles update checking is */etc/clamav/freshclam.conf*, but you really don't need to do anything to it, since it automatically checks for new virus definitions once an hour, every day. Now that's timeliness!

To actually scan your system for nasties, ClamAV uses *clamscan*. If you look at `man clamscan`, you'll see the full list of options available to you. Two that you'll want to use most of the time are `-r`, which orders ClamAV to scan recursively through the directory you specify as well as all its subdirectories, and `-i`, which has ClamAV print only the names of infected files, as opposed to every single file that ClamAV scans.

In addition, you probably want to quarantine infected files. To do so, you'll want to use an option like `--move=/tmp/virus`, but */tmp/virus* must exist before *clamscan* runs.

Putting everything together, you'd run the following command if you wanted to scan your home directory on demand:

```
$ mkdir /tmp/virus
$ clamscan -ri --move=/tmp/virus

                /home/yourusername
```

It's a better idea to set up a cron job that automatically scans your entire hard drive in the middle of the night. To do so, add the following line to */etc/crontab* as *root*:

```
11 3 * * * root mkdir /tmp/virus ; clamscan -ri --log=/var/log/clamscan.log --move=/tmp/virus /
```

At 3:11 A.M. every morning, a directory in */tmp/virus* will be created, and then *clamscan* will run. Any infected objects will be quarantined, and a log of activity will be created as well.

ClamAV probably integrates with your email program as well. KMail and Sylpheed Claws (http://claws.sylpheed.org/), for instance, allow you to select ClamAV directly as your anti-virus program of choice. Evolution and many others adopt a much less user-friendly method by forcing users to manually create a filter that pipes mail through ClamAV. That's pretty bad, and I hope that Evolution and other apps like it come to their senses and allow users to specify ClamAV—or other anti-virus programs—directly. It's the only reasonable thing to do.

Who says you have to pay $50 for expensive, bloated anti-virus software? ClamAV does the job, efficiently and easily...and for free!

For further information about ClamAV, see the Clam AV web site at http://www.clamav.net, and especially the Clam AntiVirus User Manual at http://www.clamav.net/doc/latest/html/.

Scott Granneman

Chapter 8. Administration

Yes, Ubuntu runs pretty smoothly out of the box, and if you've got a system that one person uses, you might not have to do much in the way of system configuration. However, the time may come when you need to add new users or provide tech support to friends and family you've turned onto Ubuntu. If so, then you'll find hacks in this chapter to help you out.

There are plenty of other cases where you may need to wear a system administrator's hat, and there are hacks in this chapter to help with those, too. Perhaps you need to manage and mount external drives, or mount directories from remote servers. And once in a while, things will go wrong: in many cases, you can peek into the system logs to figure out what happened, but sometimes you'll need to resort to a rescue disc.

And there's one system administration task no one can avoid: backups. You'll find hacks in this chapter for making traditional backups and keeping files between two or more computers in sync.

Hack 74. Edit Configuration Files

When you need to edit a configuration file from the command line in a pinch, use these tips for the ins and (especially) the outs of vim.

If one thing is for sure about Linux, it's that it has no shortage of text editors. This reflects the largely text-based nature of Linux, from the command line, to source code, to the configuration files that programs refer to. When you are in a desktop environment, you can use one of many graphical tools to edit text files; however, in an emergency situation, or when logged into a machine remotely, you may need to fall back on a command-line text editor. Under Ubuntu, the default text editor is *vim*, and this hack provides you with the basic information you need to make changes to configuration files using *vim*.

The *vi* editor has a rather mixed reputation. Some people love how quickly you can edit files with it, and others hate its steep learning curve. The *vim* editor (short for "Vi IMproved") takes *vi* and adds not only a number of powerful improvements, but also a number of user-friendly changes such as a complete, integrated help environment. Ubuntu includes *vim* by default, and even if you run *vi*, you are actually running *vim* in a *vi*-like emulation mode.

So, to edit a file, run *vim* on the command line followed by the filename as an argument. For instance, to edit the filesystem table (*/etc/fstab*), type:

```
$ sudo vim /etc/fstab
```

Unlike most other editors you may be used to, *vim* operates via a number of *modes*. Different keys perform different functions depending on which mode you are in. (This can be rather confusing the first time you use *vim*, since you will start typing only to see all sorts of strange and undesirable behavior.) For starters, there are two main modes you will be operating in: Normal mode (for navigating the file) and Insert mode (for editing text).

Navigate Through the File

Normal mode is the main mode you default to when you start *vim*. In this mode, you use a number of different keys to move throughout the file. Table 8-1 lists some of the main navigation keys and their functions.

Table VIM Normal Mode keybindings

Navigation key	Function
`h`, `j`, `k`, `l`	Move left, up, down, or right, respectively. The arrow keys will also move the cursor.
`w`, `b`	Move forward or backword one word at a time, respectively. Useful when you want to skip through a file quickly.
`^`, `$`	Move to the beginning or the end of the current line, respectively.
`gg`, `G`	Move to the beginning or the end of the file, respectively.
`:` *`number`*	Move to the specified line number in the file.
`/` *`keyword`*	Search for the specified keyword; hit `n` to move to the next keyword.

Edit a Line

Once you have moved through to where you wish to edit, you have a number of different options to edit a line. If you simply need to delete some text, the x key will delete the character under the current cursor. To add or change text, you will need to switch to Insert mode. Once in Insert mode, *vim* behaves a lot like other text editors; when you type a particular key, it will appear in the file. There are a number of ways to get into Insert mode, depending on where you want to start editing, but the simplest is to hit i, which will let you edit at the current position of the cursor.

Once in Insert mode, you can make any changes you need to make to a particular file. You can hit the Backspace or Delete key to delete characters and can still use the arrow keys to move around in the file. Once you are finished, hit the Esc key to exit out of Insert mode.

Copy and Paste

Sometimes when editing a file it is useful to copy and paste one line or a section of a line to another line. There are a number of ways to do so, some faster than others, but for new *vim* users, one of the easiest ways is through *vim*'s Visual mode. To select a portion of text, type Esc until you are in Normal mode, move to the beginning of the text you want to copy, and then hit the v key to switch to Visual mode. Now move the cursor as normal, and you will notice that characters will be selected as it moves. When you have selected all of the characters you want, hit the y key to copy (or yank) the text. Then move to where you want to paste it and hit p.

If you know you want to copy full lines at a time, enter Visual mode with capital V instead of v. Now move up or down to select a number of lines and type y to copy and then p to paste your lines.

Undo

Once nice feature that *vim* adds over *vi* is unlimited undos. If you made a mistake and want to undo it, hit Esc until you are in Normal mode, and then simply type the u key to step back through each edit. If you find you need to redo a change, type Ctrl-R.

Save and Exit

After you have made your changes and are ready to save, you need to switch to Command mode. Command mode lets you run special *vim* commands, including the commands you need to save changes and exit. To save (or write) your changes, type `:w` and press Enter. When you are ready to quit, type `:q` and press Enter. You can also combine these two commands into a single command, and type `:wq` and press Enter.

Search and Replace

A common need when editing files is to substitute one word for another throughout the file. To do so in *vim*, type:

```
:s/word/replacement/g
```

to change all instances of the word on a particular line. To change all instances of the word within a file, type:

```
:%s/word/replacement/g
```

If you want to give confirmation before making each change, add a c to the end of the command, typing:

```
:%s/word/replacement/gc
```

and *vim* will ask you before it makes any changes.

One particularly handy use of the search-and-replace feature when editing configuration files is for commenting or uncommenting multiple lines in the file. To do so, enter Visual mode, select each line you want to comment, and then type:

```
:s/^/#/
```

to comment the lines out with the # character. Replace # with whatever character that file uses for comments. If you instead want to uncomment a number of lines, select them in Visual mode and then type:

```
:s/^#//
```

or replace # with the character your file uses for comments.

Help

If you find you are lost, or you just want some help with *vim*'s options, type

```
:help
```

to enter *vim*'s help screen. You can also follow `:help` by a particular keyword, so if you want to find out what the w key does, for instance, type

```
:help w
```

Where to Learn More

This hack is by no means an exhaustive tutorial on how to use *vim*. *vim* has a number of advanced and powerful features that, once learned, allow you to make rapid changes to files. If you want to know more about *vim*'s features and how to use them, type:

```
$ vimtutor
```

at the command line to try *vim*'s built-in interactive tutorial program.

Hack 75. Manage Users and Groups

Use GUI and command-line tools to add, remove, and modify user and groups on the system.

So you have used your Ubuntu desktop for some time, and it has started to attract attention from the other members of your household. They all want in on the action, but you may not necessarily want them to be able to edit your files. The solution is to create new accounts for each person that wants to access your system. That way, everyone will have their own login, password, and home directory to store files. Under Ubuntu, you have a number of options to edit users and groups, both with GUI and command-line tools.

Use the User Administration Tool

Ubuntu provides a GUI tool that makes user and group management relatively easy. Click System→Administration→Users and Groups to start the user administration tool (Figure 8-1). The interface is split up into two tabs, the first for users and the second for groups, and has buttons on the right side to add, edit, and delete users and groups. By default, only the users and groups you are mostly likely to want to edit are displayed, but you can toggle the "Show all users and groups" checkbox at the bottom of the window to see everything.

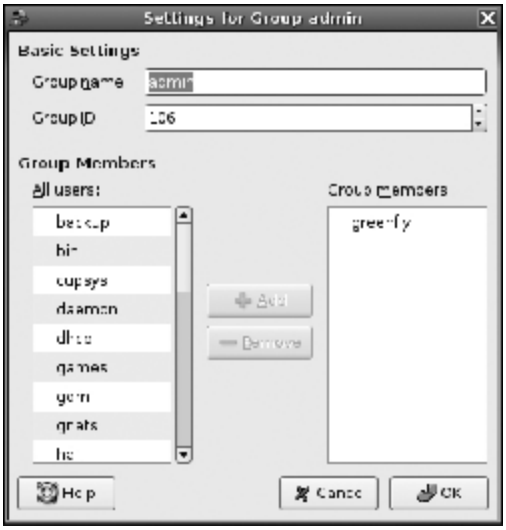
Figure 8-1. Ubuntu User Administration Tool

To add a new user, click the Add User button. A new window will appear with a number of fields to fill out (Figure 8-2). The Username and Password fields are the most crucial to fill out. You can choose a password or you can have Ubuntu generate a random password for you. If you are a more advanced administrator, click on the Advanced tab to change the default group, shell, home directory, and user ID. Ubuntu also provides three user profiles to choose from. The Default profile has no extra privileges; the Desktop profile will give the user access to audio, video, and other peripheral devices; and the Administrator profile will allow the user to become *root* (despite its name, Default is not the default choice; Desktop is). All of these settings, including user access, can be changed later on; just select a particular user and click Properties. To remove a user, select it and click Delete. Note that none of these changes will be applied until you click OK and close the program.

Figure 8-2. Add Users Configuration Window

The Groups tab lets you configure group membership on the system. Group membership dictates who can *sudo* to *root* (members of the admin group) and who can administer printers, scanners, and other devices. To see the members of a particular group, select it and then click Properties (Figure 8-3). You can select new users to add to the group from the list on the left side of the window. To create a new group, click the Add Group button and fill out the fields in the window that appears.

Figure 8-3. Ubuntu Group Account Editor



Manage Users and Groups from the Command Line

You can also add, change, and remove users and groups with command-line tools. To add a new user to the system, type:

```
$ sudo adduser
                username
```

You will be prompted for the user's real name, along with other optional information, and finally you will be asked to enter the user's password. The *adduser* script will automatically create the user's home directory and a group for the user. You can change the user ID, default home directory, and shell with the `--uid`, `--home`, and `--shell` options, respectively. Later on, to tweak the user's settings, open the */etc/passwd* file with your preferred text editor as *root* (read "Edit Configuration Files" [Hack #74] for a primer on editing text files from the command line).

If you create a user at the command line, the new user will not be assigned to the correct groups (such as *audio*) needed for the full desktop experience. You can use *sudo* to edit the */etc/group* file and give the new user the appropriate group memberships, using the first user you created as a guide.

To delete a user, use the *deluser* command:

```
$ sudo deluser
                username
```

By default, this will remove only the user from the system, not any of the user's files. To remove the user's home directory and mail spool, add the `--remove-home` option. To remove all files on the system owned by the user no matter where they are, use the `--remove-all-files` option.

There are similar command-line commands to add and remove groups from the system. To add a group to the system, use the *addgroup* command:

```
$ sudo addgroup
                groupname
```

You can use the `--gid` option to specify which group ID number to use. All system groups are found in the */etc/group* file, so you can edit this file as *root* if you want to tweak group membership or other settings. If you want to add a current user to a current group, instead of editing */etc/group*, you can type:

```
$ sudo addgroup
                username
```

```
                groupname
```

To remove a group from the system, use *delgroup*:

```
$ sudo
                delgroup

                groupname
```

Be careful when deleting users or groups, particularly any users or groups you did not explicitly create. Accounts created by Ubuntu when it was installed, or when certain packages were installed, should generally stay as they are. Leave them be unless you know what you are doing.

Hack 76. Mount Any Filesystem

Tweak /etc/fstab to control which filesystems are mounted at boot.

By default, Ubuntu will automatically detect and configure *mount points* for any partitions it finds when Ubuntu is installed. However, if you add a new disk to the system, or you want to automatically mount an NFS or SMB share at boot time, you must resort to the tried-and-true Linux method: editing */etc/fstab*.

The */etc/fstab* file (short for *filesystem table*) keeps track of filesystems that you want to mount in static locations. Here is a standard Ubuntu *fstab* file, which provides a good example for what each field in the file stands for:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/hda1        /              ext3    defaults,errors=remount-ro 0       1
/dev/hda5       none            swap    sw              0       0
/dev/hdc        /media/cdrom0   udf,iso9660 user,noauto 0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto  0       0
```

Here you see a list of partitions Ubuntu will mount, where it will mount them, what type of filesystem they use, any special options that might need to be passed to the filesystem, whether the partitions should be included in partitions the dump utility backs up, and what order filesystems are checked at reboot time. It's a lot of information, to be sure, but when you want to add a new filesystem to the mix, there are only a few fields that are crucial.

Add a Local Partition

One of the simpler partitions to add to the *fstab* file is a local partition. Suppose, for instance, that I added a second IDE hard drive to the system (*hdb*), and it had a Windows FAT32 partition as the first partition (*/dev/hdb1*) and a Linux EXT3 partition as its second partition (*/dev/hdb2*). First I need to create a place these filesystems will mount, so I create two new directories under */mnt* called *windows1* and *linux1*:

```
$ sudo mkdir /mnt/windows1 /mnt/linux1
```

Now I can add two lines to my */etc/fstab* file, one for each of the new partitions:

```
/dev/hdb1      /mnt/windows1   vfat    defaults      0       0
/dev/hdb2      /mnt/linux1     ext3    defaults      0       2
```

Notice the differences between these two lines. Besides the fact that the first column in each line lists a different partition and the second column has a different mount point, notice that in the third column I specified `vfat` for the filesystem type for the Windows partition and `ext3` for Linux. The value that goes here varies depending on what type of partition you are mounting, and Table 8-2 displays some of the common partition types and what value to put in */etc/fstab*.

Table Filesystems and their fstab labels

Filesystem	fstab label
Windows FAT, FAT32	vfat
Windows NTFS	ntfs
Linux EXT3	ext3
Linux EXT2	ext2
ReiserFS	reiserfs
XFS	xfs
JFS	jfs
SMB Network File Share	smb
NFS Network File Share	nfs
CD-ROM / DVD-ROM	udf,iso9660
RAM Filesystem	ramfs
Compressed RAM Filesystem	cramfs

In the fourth column, both fields list `defaults`, which mounts the filesystem with default settings. Different filesystems support different mount options, such as whether to mount the filesystem as read-only or read/write, what sorts of ownership files in a partition might have, and so forth. You can get more information about special options that a filesystem supports from the *mount* manpage:

```
$ man 8 mount
```

In the fifth column, I left the values at 0 because I am not planning on using the *dump* utility to back up these partitions. This is a safe value for most partitions you might want to mount.

In the final column, notice I set the Windows partition to 0 and the Linux partition to 2. I did so because I don't want Linux to automatically run *fsck* on my Windows partition, yet I do want the Linux partition to be among the drives that are checked. I set the Linux partition's value to 2, because the standard is to set the root filesystem (*/*) to 1 so that it is checked first, then all other Linux filesystems to 2.

Mount the Filesystems

Now that these two file systems are configured, I can use the *mount* command to mount them:

```
$ sudo mount /mnt/windows1
$ sudo mount /mnt/linux1
```

While I could have used the *mount* command to mount these filesystems manually, without bothering to add any entries to *fstab*, when I set up the filesystems in *fstab*, I don't have to list a lot of options—just the device or mount point I want to mount. After mounting your devices, you can type either `mount` or `df` to see all of your mounted filesystems.

Mount Network Filesystems

Network filesystems use slightly different syntax than ordinary partitions, so they get a separate section here. Specifically, the syntax you use to describe the filesystem (the first column) is different. For this example, I have an NFS share on host *file1* at */mnt/shares*, and a SMB file share on host *file2* called *data*. I want to mount the NFS share at */mnt/share1* and the SMB share at */mnt/data1*. To mount both of these partitions at boot time, I would add the following lines to my */etc/fstab*:

```
file1:/mnt/shares      /mnt/share1   nfs    defaults     0       0
//file2/data           /mnt/data1    smb    defaults     0       0
```

Notice the difference in syntax. NFS shares follow the *`hostname:/path/to/share`* syntax, while SMB shares follow the *`//hostname/share`* syntax. Other than that, the remaining fields are similar to the first examples, and I could pass special NFS or SMB options in the options field if I needed to.

If you don't want a partition to mount at boot time (like floppies and CD-ROM drives), add the `noauto` option to the list of options.

To mount these shares, I would make sure that the */mnt/share1* and */mnt/data1* directories existed, and then I would type:

```
$ sudo mount /mnt/share1
$ sudo mount /mnt/data1
```

As with local partitions, you can use the *mount* or *df* commands to check the status of these mount points.

For more information on the syntax for the */etc/fstab* file, and what sorts of mount options you can use for particular filesystems as well as what filesystems Linux can mount, see the *fstab(5)* and *mount(8)* manpages, respectively.

Hack 77. Control Startup Services

Use GUI and command-line tools to start, stop, disable, and enable services.

As Ubuntu boots, you might notice text scrolling by, detailing all of the different things Ubuntu is doing. Among these things are a number of services that Ubuntu enables at boot time, such as the cron scheduling service, the system logger, and the graphical login manager. If you have installed other services on your system such as a web server, those services will also be enabled at boot. Sometimes, though, you may want to either stop or temporarily disable these services, and Ubuntu provides a number of ways to do this, both graphically and through the command line. This hack shows some of the more common ways to control startup services.

Services Administration Tool

Ubuntu provides a graphical tool to manage services that start up at boot time. Click System→Administration→Services, or type:

```
$ sudo services-admin
```

to start the Services Administration Tool. Figure 8-4 shows how simple the program is—you have a list of services with a checkbox next to each of them. To disable a service, just deselect its checkbox and click OK. Currently, this application supports changing only whether a service starts at boot, so to manually start, stop, or restart a service, you will need to refer to the command-line method.

Figure 8-4. Ubuntu Services Administration Tool

Command-Line Method

Before learning how to start, stop, and disable services from the command line, it's important to understand Ubuntu's startup process and how Ubuntu determines which programs to run when it starts. For most Linux distributions (including Ubuntu), *System V init scripts* govern which programs start at boot and which programs don't. All System V init scripts that could potentially be run at boot are typically located in the */etc/init.d/* directory for Ubuntu. Not every script in */etc/init.d* is executed at boot, however. Linux organizes which scripts to run for different circumstances into runlevels; most Linux systems have seven runlevels, ranging from 0 to 6. Think of a runlevel as a checklist of programs for Ubuntu to start before it presents a login.

A few of these runlevels are set aside for special states in a Linux system:

Runlevel 0
: Halts the system.

Runlevel 1
: Sets up single-user mode.

Runlevels 2–5
: Set up different multiuser modes. Although, typically, only one or two of these are used by a distribution.

Runlevel 6
: Reboots the system.

Each runlevel has a directory that stores symlinks to certain init scripts in */etc/init.d*, which are started when that runlevel is selected and stopped when it is exited. Ubuntu puts these symlinks under */etc/rc* `runlevel` *.d*—for example, all runlevel 2 scripts are located in */etc/rc2.d/*.

If you look in one of these runlevel directories, you'll notice that many of the symlinks to scripts in */etc/init.d* have odd names that begin with an *S*, *K*, or *D*; then a number; and finally the name of the script. Ubuntu defaults to runlevel 2, so here is a sample */etc/rc2.d* directory:

```
greenfly@ubuntu:~$ ls -l /etc/rc2.d/
total 0
lrwxrwxrwx 1 root root 20 2006-01-21 14:48 K77ntp-server -> ../init.d/ntp-serverlrwxrwxrwx 1 root root 17 2006-
lrwxrwxrwx 1 root root 15 2006-01-07 08:19 S10acpid -> ../init.d/acpid
lrwxrwxrwx 1 root root 18 2006-01-07 08:17 S10sysklogd -> ../init.d/sysklogd
lrwxrwxrwx 1 root root 15 2006-01-07 08:17 S11klogd -> ../init.d/klogd
lrwxrwxrwx 1 root root 14 2006-01-07 08:47 S12dbus -> ../init.d/dbus
lrwxrwxrwx 1 root root 13 2006-01-07 08:50 S13gdm -> ../init.d/gdm
lrwxrwxrwx 1 root root 13 2006-01-07 08:46 S14ppp -> ../init.d/ppp
lrwxrwxrwx 1 root root 16 2006-01-07 08:48 S19cupsys -> ../init.d/cupsys
lrwxrwxrwx 1 root root 15 2006-01-07 08:52 S19hplip -> ../init.d/hplip
lrwxrwxrwx 1 root root 14 2006-01-07 08:47 S20apmd -> ../init.d/apmd
lrwxrwxrwx 1 root root 22 2006-01-07 08:48 S20hotkey-setup -> ../init.d/hotkey-setup
lrwxrwxrwx 1 root root 21 2006-01-07 08:46 S20laptop-mode -> ../init.d/laptop-mode
lrwxrwxrwx 1 root root 17 2006-01-07 08:16 S20makedev -> ../init.d/makedev
lrwxrwxrwx 1 root root 23 2006-01-07 08:18 S20nvidia-kernel -> ../init.d/nvidia-kernel
lrwxrwxrwx 1 root root 19 2006-01-07 08:48 S20powernowd -> ../init.d/powernowd
lrwxrwxrwx 1 root root 15 2006-01-07 08:46 S20rsync -> ../init.d/rsync
lrwxrwxrwx 1 root root 21 2006-01-07 08:47 S25bluez-utils -> ../init.d/bluez-utils
lrwxrwxrwx 1 root root 15 2006-01-07 08:17 S25mdadm -> ../init.d/mdadm
lrwxrwxrwx 1 root root 17 2006-01-07 08:46 S89anacron -> ../init.d/anacron
lrwxrwxrwx 1 root root 13 2006-01-07 08:46 S89atd -> ../init.d/atd
lrwxrwxrwx 1 root root 14 2006-01-07 08:46 S89cron -> ../init.d/cron
lrwxrwxrwx 1 root root 17 2006-01-07 08:18 S98usplash -> ../init.d/usplash
lrwxrwxrwx 1 root root 22 2006-01-07 08:46 S99acpi-support -> ../init.d/acpi-support
```

```
lrwxrwxrwx 1 root root 19 2006-01-07 08:16 S99rmnologin -> ../init.d/rmnologin
lrwxrwxrwx 1 root root 23 2006-01-07 08:16 S99stop-bootlogd -> ../init.d/stop-bootlogd
```

As you can see, this directory is full of symlinks that point to a script in the *init.d* directory. The letter at the beginning of each filename tells *init* when to execute this script. If the script begins with an *S*, then *init* starts the script when it goes through the runlevel. If the script begins with a *K*, then *init* stops (or kills) the script when it changes to a different runlevel. If the script begins with a *D*, then that script is disabled for the time being, and *init* ignores it. *init* runs the scripts in numerical order, so the numbers in each script let you know in which order they are to be run. This is useful to ensure that dependent services start after the service they are dependent on.

When Linux boots and starts the *init* process, it reads its configuration from */etc/inittab*, which configures each available runlevel, the default runlevel to use, and some other settings. Next, *init* loads any system scripts from a special system runlevel directory at */etc/rcS.d*. These scripts load daemons and services that are vital to the boot process. Lastly, *init* runs any startup scripts for the default runlevel in alphabetical order.

Scripts in */etc/rcS.d* are run in runlevels 1 through 5, so you should generally leave them alone unless you know what you are doing. If you accidentally disable a crucial service, you may have to resort to a rescue disc to undo the mistake.

Change the Runlevel

You can change the runlevel yourself on the command line with the init command. To switch to single-user mode from the command line, type:

```
$ sudo init 1
```

If you're running X11 when you issue this command, beware, since it will kill X and your desktop environment!

This command runs all of the shutdown scripts for your current runlevel and then any startup scripts for single-user mode. To change back to the default multiuser runlevel for Ubuntu, type:

```
$ sudo init 2
```

You can also use *init* to halt or reboot a machine: just change to runlevel 0 and runlevel 6, respectively.

Manually Start and Stop Services

You can start and stop scripts manually by running the script with the start or stop argument. For example, to stop the CUPS service from the command line, type:

```
$ sudo /etc/init.d/cupsys stop
```

To start the service back up, type:

```
$ sudo /etc/init.d/cupsys start
```

Most scripts also support a restart argument which will run *stop*, then *start* for you. Most *init* scripts are also configured to output the list of arguments they support when you execute them without any options:

```
greenfly@ubuntu:~$ sudo /etc/init.d/cupsys
Password:
Usage: /etc/init.d/cupsys {start|stop|restart|force-reload|status}
```

Disable Scripts from Starting

To disable a script, you must know your default runlevel. On Ubuntu, the default runlevel is usually set to 2, but you may want to double-check your default runlevel before you start disabling services. All runlevels are defined in */etc/inittab*, so to check the default runlevel, type:

```
greenfly@ubuntu:~$ grep initdefault /etc/inittab
id:2:initdefault:
```

As you can see, in this example, the default runlevel is in fact 2. Now change to the directory containing all of the scripts for that runlevel (*/etc/rc2.d*) and find the script you want to disable. To disable a service, just rename its script by changing the *S* to a *D*. For example, to disable the cupsys service, type:

```
greenfly@ubuntu:~$ cd /etc/rc2.d
greenfly@ubuntu:/etc/rc2.d$ sudo mv S19cupsys D19cupsys
```

To enable it again, rename it back by changing the *D* to an *S*:

```
greenfly@ubuntu:~$ cd /etc/rc2.d
greenfly@ubuntu:/etc/rc2.d$ sudo mv D19cupsys S19cupsys
```

You'll still need to stop the service as shown earlier if you want to shut it down right away, but renaming it will control whether it's started the next time you reboot (or change runlevels).

Hack 78. Build Kernels the Ubuntu Way

Special tools take the pain out of compiling and installing custom kernels.

Debian boasts some very useful tools to make the job of building and installing kernels much easier, and of course those same tools are available in Ubuntu as well. They streamline the process of compiling a custom kernel and building a *.deb* package around it, allowing you to install a new kernel in the same way as you install any other package. This makes it possible to build a kernel on one machine and then just install the package on other machines, without having to recompile or track down all the loose ends—great if you need to upgrade a number of similar machines!

To build a custom kernel "the Ubuntu way" you will need to get hold of the kernel source and a variety of tools to configure, compile, and package it.

Get the Source

You can get the official kernel source yourself directly from http://ftp.kernel.org/pub/linux/kernel/, or a mirror if you like, but of course there are even packages of the kernel source. A quick:

```
$ apt-cache search linux-source
```

will get you a list of Linux kernel source packages all ready to go. Picking one as an example, you could just run:

```
$ sudo apt-get install linux-source-2.6.15
```

to download the source to your system. You'll then end up with an archive sitting in */usr/src*, which is where most kernel work is done. Now you can extract the source package:

```
$ cd /usr/src
$ sudo tar -xjf linux-source-2.6.15.tar.bz2
```

That will leave you with a */usr/src/linux-source-2.6.15* directory, decompressed and ready to configure. First, though, create a symlink to it called *linux*, like this:

```
$ sudo ln -sf linux-source-2.6.15 linux
```

The `f` option forces creation of the new symlink even if there's already an existing link to an old kernel source. Now you can easily get to your kernel source tree just by typing

```
$ cd /usr/src/linux
```

Tools You Will Need

To install the main tools, you will need to run:

```
$ sudo apt-get install kernel-package libncurses5-dev
```

This will also cause your computer to pull down a big list of supporting packages that are dependencies.

Configuration Methods

There are a number of ways to configure a kernel prior to compiling it, and all but the first are invoked as arguments to the *make* command. Make sure you have a shell open in the kernel source directory (which should be */usr/src/linux* if you followed my directions earlier) before you try any of these. In increasing order of sophistication, they are:

editing .config

The actual configuration is saved in the source directory as a text file called *.config*. If you really want to (or if you are trying to find an option some *README* has listed by name, such as `CONFIG_PACKET`) you can edit this file directly using a text editor such as *vim* or Anjuta. This should need to be done very rarely, though.

`make config`

The most basic approach: this will just ask you a whole heap of questions one after another. Personally I hate configuring the kernel this way, because it's sequential and you have to go through a heap of stuff you probably couldn't care less about. Only bother with this as a last resort, such as if other options aren't available to you for some reason.

`make menuconfig`

This is the most common way to do the configuration. It displays a nice, keyboard-driven menu that you can navigate using arrows, Enter and the spacebar. However, one catch that can trip you up is that you need to have the development libraries for Ncurses installed, not just Ncurses itself. That's why I had you install the *libncurses5-dev* package a few paragraphs ago!

`make xconfig`

The preferred method for machines with X, this option is very similar to *menuconfig*, except it's all done in a nice point-and-click GUI with mouse navigation. The possible gotcha to getting this working is you'll need *tk8.3*. If you get errors saying it can't find "wish," run:

```
$ sudo apt-get install tk8.3
```

and all should be well.

Configuring the Kernel

Whichever configuration method you choose, the next thing to do is to examine the settings for the different kernel options and make changes as required.

There are a huge number of kernel options, so to help you find things, they are grouped together in a logical way in different major and minor sections. The different configuration methods have their own ways to represent this, but they should all be fairly straightforward to follow.

I won't go into details of how to do the kernel configuration here, because what you'll need to configure will vary dramatically, and there are plenty of tutorials on the Net, including the Kernel-HowTo, that explain this in detail. Basically, though, you need to work through each menu in turn to find options and modules you want to enable, and set them to either *off* (not available), *on* (compiled straight into the kernel), or *module* (compiled separately so it can be loaded when needed).

Once you've finished going through the options, quit and save to have your choices written out to a configuration file to be used by the compiler.

One little trick to note is that the actual configuration details are stored in a file called *.config* in the kernel source directory. Once you've gone to all the trouble of setting up a kernel the way you want it, moving to a new kernel can be painful if you have to go through the configuration process from scratch, and you'd be almost certain to miss something crucial—I know I usually do! To make things easier for yourself, you can copy the *.config* file into the new kernel source directory to have all your options carried over automatically. Then you can run the configuration again just to check things over, save, and go on as before. Provided you're not changing to a totally new kernel type, this trick can save you a lot of time. Beware if you're moving to a totally different kernel, though, such as from 2.4.x to 2.6.x, because many of the options will be totally different and while the old config will still work, it may not have some options activated that appear only in the new kernel. You can also run `make oldconfig` after copying in the old *.config* file, which will then prompt you only for options that are new or have changed since you configured your previous kernel.

Compiling and Packaging

This is the point when your kernel is actually compiled and placed in a package. Normally, that would be a lot of work, but thanks to the tools provided by *kernel-package*, it's now one of the easiest steps.

If you have read a traditional Kernel How-To, you've probably seen a sequence of commands such as `make dep && make clean && make bzImage` used to build the dependencies and then the kernel itself. You don't need to do any of that; instead just type:

```
$ sudo make-kpkg kernel-image
```

while in the */usr/src/linux* directory, and those steps will all be taken care of. The kernel-packaging tools first compile your kernel and modules according to the configuration you just generated and then build a Debian package called *kernel-image-* `version_architecture` *.deb* outside the current source directory—i.e., in */usr/src*.

This stage can take a while depending on your machine speed and which modules you selected. Expect anything from 10 minutes to a half hour, during which time you'll see a very long stream of debugging information that probably won't be of any interest to you at all, unless something goes seriously wrong.

Install Your Kernel Package

You're almost there! What you have now is a kernel, custom-compiled according to your requirements and set up as a Debian package ready to install.

Installing your new kernel is now just a matter of using *dpkg* to install the package as you would any other Debian package:

```
$ sudo dpkg -i
                kernel-image-2.6.15_10.00.Custom_i386.deb
```

(replacing *kernel-image-2.6.15_10.00.Custom_i386.deb* with whatever your package is called). It may ask you if you want to create a boot floppy using that image; the dozens of machines I manage don't have a single floppy drive among them, so I always say no, but if you want to, you can say yes to have it create one for you.

dpkg will also take care of updating your bootloader configuration so your new kernel will be available next time you boot. Both GRUB and LILO are managed automatically, but if you use a different bootloader, you may need to update it yourself at this point.

What happens when you install the package is that *dpkg* puts your new kernel image in the */boot* directory where kernels are normally stored, creates a symlink from */vmlinuz* to your new kernel (you can verify this by typing `ls -l /vmlinuz`), modifies your */boot/grub/menu.lst* or */etc/lilo.conf* as appropriate so your bootloader can find your new kernel, and moves and renames the previous kernel so it's available as an emergency fallback in case your new kernel borks.

Note that at this point, you will not actually be running the new kernel; it's just set up, ready to go for next time you reboot. Installing a new kernel is one of the very few things that actually requires you to reboot a Linux system.

Rebooting and Testing

Time for the big test! If you are running X, exit the session and reboot. If not, just type `sudo reboot`. When GRUB loads, you will have the option of pressing Esc to see a list of available kernels, including your new one. Select it and hit Enter.

Once your machine has booted, you can use *uname* to check which kernel you are running:

```
$ uname -a
```

For more information about which modules were loaded and whether the new kernel correctly detected your hardware, you can look through the dmesg log:

```
$ dmesg | less
```

If everything worked as expected, congratulations! You've just compiled and installed a custom kernel "the [Debian|Ubuntu] way."

Installing on Other Machines

This is where the convenience of building kernels as packages becomes most obvious. If you want to install your custom kernel on other machines, the process is very simple: just copy the *.deb* package you created to the target machine and install it using *dpkg* exactly as before. Simple!

There's no need to compile the kernel on each machine, copy the source code to them, or even have a compiler installed on them. All the hard work was done once on one machine and doesn't need to be repeated.

If you run a server farm or computer lab and have a lot of local machines to install your kernel on, the best way to do so is probably to put the kernel package on a local package repository and have all your computers fetch it over the network. You can learn more about how to do so in "Create Your Own Package Repository" [Hack #65].

Install Multiple Copies of One Kernel Version

Often you'll want to install multiple copies of the same kernel version while doing testing, but that can cause problems: modules are installed in a directory based on the kernel version number, so if you want to try out different configuration options, you'll run into problems. Luckily, the kernel package tools allow you to work around that limitation by appending an arbitrary version string to the regular version string. This means you can install multiple builds of the same kernel version and keep everything neatly organized on disk, as well as select which one you want to use at startup.

To modify the version string, just pass the `--append-to-version` option when creating the package, like so:

```
$ fakeroot make-kpkg --append_to_version=-jon17
                kernel-package
```

This would generate a kernel package something like *kernel-image-2.6.15-jon17_10.00.Custom_i386.deb*.

Hack 79. Back Up Your System

Use Ubuntu's Simple Backup utility to easily set up standalone or recurring backups of your important files.

Backups always seem to be the thing we think about after it's too late. Just about every computer user I know has been bitten at least once by failing to make a backup of an important file or directory. Yet what should you back up? How often? Out of the hundreds of backup tools out there, which should you use? Ubuntu has answered all of these questions with the aptly named Simple Backup tool. This tool allows you to scheduling automatic recurring backups of predefined important files on the system, or, if you want more control, it allows you to set fine-grained backup options that better suit your needs. In this hack, I discuss some of the main options in the Simple Backup tool and how to set up a backup-and-restore solution for your computer.

Get Simple Backup

The first step is to install Simple Backup. Its package is called *sbackup* and is available from the *universe* repository, so if you haven't yet enabled *universe*, you will need to do so. If you need more instructions on how to enable repositories, refer to "Modify the List of Package Repositories" [Hack #60] or individual hacks corresponding to your preferred package-management tool [Hacks #54, #55, and #56].

Once the program is installed, two new menu entries will appear under System→Administration: one called Simple Backup Config and a second called Simple Backup Restore. Select Simple Backup Config to see the fairly straightforward main window shown in Figure 8-5.

Figure 8-5. The main Simple Backup window

Configure Simple Backup

The General tab display three main options for you to choose from:

Use recommended backup settings

If you are new to Ubuntu or aren't quite sure what you should backup, but you do want to set up recurring backups, select this option. It will automatically set up daily incremental and weekly full backups of important directories on your system and store them under */var/backup*. Note, however, to prevent the backup from filling up your disk, this option excludes files larger than 100 MB as well as a number of multimedia files, so if you want to back up either of those, move to the next options.

Use custom backup settings

If you have specific backup needs, or the default choices from the recommended backup don't suit you, choose this option to access the fine-grained configuration options in the following tabs (more on them shortly).

Manual backups only

If you don't want to set up a recurring backup schedule, but you do want to be able to manually back up some files from time to time, select this option and read on for more information on how to configure what to back up in the following tabs.

Use recommended backup settings

If you are new to Ubuntu, don't know what exactly to back up, or don't want to fuss with a configuration, check the "Use recommended backup settings" in the General tab and then click Save. Ubuntu will schedule a nightly incremental backup of important files on the system, along with a weekly full backup. If you later find you need to restore a file, see the "Restore from Backup" section later in this hack.

Use custom backup settings

There are a number of reasons why the default backup settings might not suit you. For instance:

- You may have a number of important files that are larger than 100 MB, and you need to be able to back those up.
- You want to keep a backup of your multimedia files.
- You want to save the backups somewhere other than */var/backup* (either another directory on the system or possibly a directory on a remote system).
- You want to include or exclude other files or directories from the backup.
- You want to schedule your backups to run at different times or different frequencies from the default.

If you want to enable custom recurring backups, select "Use custom backup settings." If you want to configure a custom manual backup instead, select "Manual backups only." Apart from the Time tab being disabled in the manual backup option, the configuration ability will be identical.

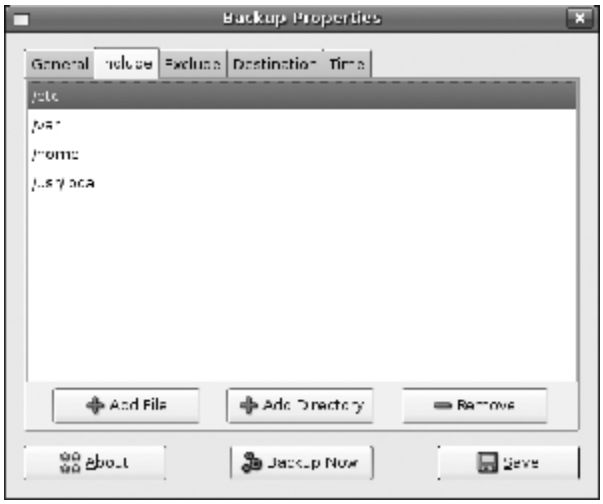
Configuring Custom Backup Settings

Here are some settings you can modify to customize the backup of your system.

Configure files to include

By default, Ubuntu includes the major important directories you might want to backup on a system, such as */etc*, */var*, */home*, and */usr/local*. It's possible you might have an extra directory or file that you want included in addition to these (possibly an extra mount point under */mnt*). If so, click the Include tab to see the directories and files currently included in the backup (Figure 8-6).

Figure 8-6. Define which directories to include in the backup

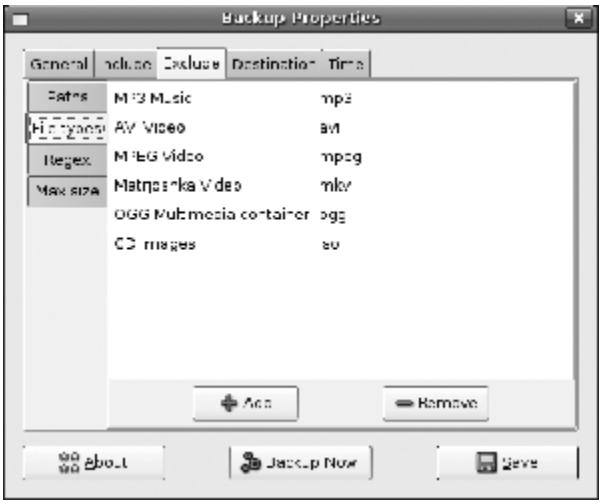


Configure files to exclude

There are a number of different files that Simple Backup excludes by default. Click the Exclude tab to configure these exclusion options. Figure 8-7 shows an example of some multimedia file types that are excluded by default. If you do want to backup your MP3s for instance, select "mp3" from the list of excluded files and click Remove. If you want to add a particular file type to the exclusion list, click the Add

button and choose a predefined file type from the drop-down menu, or enter the file's extension and then click OK. This particular exclusion list depends on actual filename extensions, so it will work only on files with the correct extensions in their filenames.

Figure 8-7. Define which file types to exclude

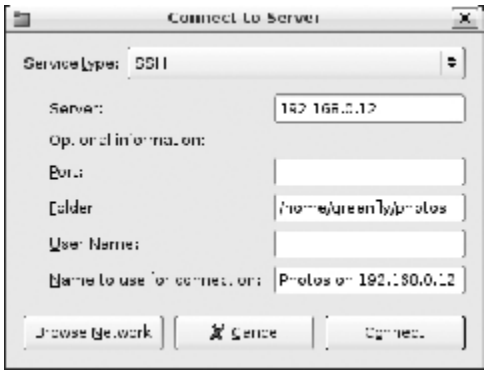


A number of different exclusion properties are allowed via the tabs along the left side of the window. In addition to File types, click Paths to exclude by individual files or directory paths. If you are familiar with defining patterns using regular expressions, click the Regex tab and you will be able to define an exact pattern to match files to exclude. The final tab in this window, "Max size," is important because it will greatly influence the size of your backups. If you choose the custom configuration, it will default to excluding files larger than 10 MB. Depending on the size of the files you want to back up, you may want to raise or completely disable this option, but note that in doing so you risk filling up your backup directory if it doesn't have adequate space.

Configure the backup destination

The Destination tab allows you to change where Simple Backup will store backups. You can either select the default of */var/backup*, choose a custom directory somewhere on your system, or you can even select a remote FTP or SSH share. To back up to a remote share, use the "Connect to Server" dialog shown in Figure 8-8 or type `ssh://` or `ftp://` followed by the `username:password`, then @, then the remote host to connect to and the remote directory.

Figure 8-8. Backup to a remote share



ED/AUs: How do you access the Connect to Server dialog mentioned in the previous paragraph? Any further explanation needed?

Configure recurring backups

If you want to have Simple Backup automatically back up your system periodically, click the Time tab to configure how frequently to back up (Figure 8-9). You can choose anything from never backing up to backing up hourly, daily, weekly, or monthly. You can also configure the time, day of week, or day of month to back up, depending on how frequently you want to back up the system. Finally, you can configure how often to do a full backup. By default, a full backup is performed weekly, with incremental backups being performed otherwise to save on space.

Figure 8-9. Schedule your backups in the Time tab

When you are finished with your backup configuration, click Save to save the configuration and exit, or click Backup Now to start the backup right this instant based on your configuration.

Restore from Backup

So you have accidentally deleted a file, or you need to roll back to a previous version of it. If you have already configured recurring backups, or have manually performed backups using Simple Backup, click System→Administration→Simple Backup Restore. Figure 8-10 shows the main window with a particular backup set selected. You can choose from any previous backups from the "Available backups" drop-down menu, and you can then browse through the directory structure of the backup to find the file or directories you want to restore. Once you have found the files or directories you want to restore, click Restore to overwrite any current version of the file with the backed-up version, or click Restore As to restore the file or directory to a new location with a new name so you don't overwrite the current version.

Figure 8-10. The Simple Backup Restore window

Hack 80. Clone an Installation

Export a list of installed packages on one Ubuntu system, and import them into another to build a duplicate system.

I've installed many different Debian-based distributions over the years, and one thing I've found is handy to have is a complete list of packages you have installed. If you want to create a system that is similar to a different system you have already set up, it can be difficult to remember each and every package you had installed. In this hack, I cover a method to export your current package list and import it into a new system.

This method works best when you are exporting to and importing from the same distribution and, specifically, the same release—for example, exporting from Ubuntu Dapper to Ubuntu Dapper. Because of the differences in package versions and dependencies across releases, and especially across distributions, you will have substantially more headaches with conflicting packages if you try to export, say, from Ubuntu Breezy to Ubuntu Dapper or, worse, vice versa.

Export the List of Installed Packages

The first step in cloning an installation is to grab the complete list of installed packages from the first system. To do so, you basically instruct *dpkg* to dump its entire list of packages, filter out any packages that aren't currently installed, and then redirect the output to a file:

```
$ sudo dpkg --get-selections | grep '[[:space:]]install$' | \\
              awk '{print $1}' > package_list
```

Next, copy this text file to the destination system over the network, via a USB key or whatever method you prefer. You may also want to copy over the */etc/apt/sources.list* file from the base system, since the new system may not have all of the same repositories enabled (if the repositories aren't the same, the destination system may not be able to find some of the packages in the list).

Prepare the Destination System

Now you need to prepare the destination system. If both systems are running the same release of Ubuntu, this may be as simple as just copying the initial system's */etc/apt/sources.list* on top of the one on the current system. Otherwise, if the systems are from different releases, you will want to compare the initial system's *sources.list* with the */etc/apt/sources.list* file on your destination system and see if there are any extra third-party repositories or repository subcategories that need to be enabled. (Read "Modify the List of Package Repositories" [Hack #60] for more information on Ubuntu's repositories and how to edit *sources.list*.)

Once your *sources.list* file is settled, update your package list to make sure you get the latest version of the packages:

```
$ sudo apt-get update
```

Import the Package List

To import the package list, pipe the entire list to *xargs*, which then splits it up into manageable chunks for the *apt-get* command:

```
$ cat package_list | xargs sudo apt-get install
```

If you are migrating to a different Ubuntu release, this will require a bit of trial and error, since you will likely get complaints about packages no longer existing. The simple fix for this is to edit the package list and remove the package from the list, save your changes, and then rerun the command. You will likely need to do this a number of times, but eventually you will have a list of valid packages. A more complicated but thorough fix would involve checking the latest list of packages for a potential update or replacement for the package that no longer exists and installing them one by one.

Once *apt-get* completes, you are finished. All of the files from the package list will have imported into the new system. Now keep in mind that this doesn't mean that all of the settings have transferred over. To do that, you will likely need to copy settings from the */etc* directory or possibly other directories, depending on the package.

Hacking the Hack

It can be rather handy to have a complete list of installed packages for backup and restore purposes. An easy way to automatically generate a complete list is to have a cron job update the list of installed packages every night. To do so, create the following script and save it in */etc/cron.daily/package_list*:

```
$!/bin/sh

dpkg --get-selections | grep '[[:space:]]install$' | \\
  awk '{print $1}' > /etc/package_list
```

Now make the script executable:

```
$ sudo chmod a+x /etc/cron.daily/package_list
```

This */etc/package_list* file can then be backed up [Hack #79] with the rest of your system settings.

Hack 81. Rescue an Unbootable System

When you've had a little too much fun with your new Ubuntu system and it no longer boots, here are a few options to get your computer back in running order.

If you are reading this hack, I want to offer you my condolences. It can be stressful and upsetting when your system won't boot. I'm sure you are thinking right now about all of those files you wished you backed up (I know this is a bad time, but once your system is back up and running you might want to check out "Back Up Your System" [Hack #79]). I've certainly been in the same situation far too many times, but so far I've also been able to bring my system back to life. While it would be impossible to cover every possible scenario that might cause a system to no longer boot, I'm going to go over how to use the Ubuntu install CD in rescue mode to fix the problem and describe some common rescue scenarios.

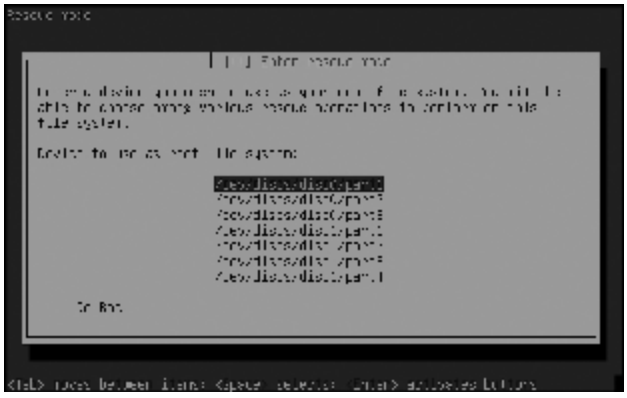
Boot into Rescue Mode

First, find your trusty Ubuntu install CD and reboot your system to its initial boot screen. Among the number of options is "Recover a broken system." Select this option, and Ubuntu will start what appears at first to be the default installation program. You will be prompted for some standard language and network settings as you are during the installation, but these steps are only to set up the initial rescue environment. Notice that in the top-left corner of the screen, "Rescue mode" appears.

Continue through these dialogs until you are prompted to choose your root device. In Figure 8-11, you see a sample of this dialog with a lot of different drives. How do you pick the right one? If you aren't too familiar with the partition layout of your drives, it might require a bit of trial and error, but here's a basic rule of thumb:

- If you installed Ubuntu as a standalone system and had Ubuntu overwrite everything on the current drive, your root filesystem is probably the very first partition on the list.
- If you installed Ubuntu in a dual-boot configuration with Windows, your root filesystem is probably the second partition in the list.

Figure 8-11. The rescue-disc root-device dialog

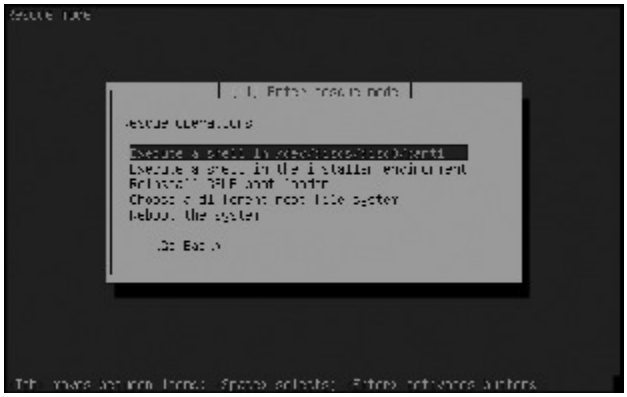


After you choose your root filesystem, the Ubuntu CD will attempt to mount it, and it will continue to the next dialog if it was successful. If it wasn't successful, you might have chosen the wrong partition, so pick another and try again.

Common Rescue Scenarios

The next dialog will present you with a number of different rescue options (see Figure 8-12). The operation you choose here will vary greatly depending on the symptoms of your unbootable system, so instead of describing everything possible with these options, I'm going to describe a few common rescue scenarios and what to do from this dialog to fix the problem.

Figure 8-12. The rescue-mode chooser



The system won't go past the boot menu, or the menu is missing

This is a common symptom for an unbootable system. Perhaps you reinstalled Windows or some other operating system as a dual-boot system. Whatever the reason, your default GRUB boot menu is now missing and needs to be restored.

From the rescue operations menu, select the "Reinstall GRUB boot loader" option. The next screen that appears is the same one you might remember being prompted with when you first installed Ubuntu. It asks you where you would like to install the GRUB boot loader. Unless you remember setting up GRUB somewhere in particular, chances are you probably installed it to the Master Boot Record on your first hard drive. If so, type `(hd0)` and then continue. The rescue mode will reinstall GRUB and drop you back to the main "Rescue operations" menu. Select "Reboot the system," and you should hopefully be presented with your standard boot menu.

The system goes past the boot menu but can't access the root filesystem

This can be a tricky symptom, as a number of different problems can be the cause, from corrupted filesystems to new problematic kernels. Before you go through any steps in the rescue mode, first try to boot and select a different kernel option in the boot menu. If that kernel boots, then there is likely some sort of hardware support problem for your latest kernel. Check for updates to your kernel package, and in the meantime, boot from the working kernel. If you experience the same problem with the different kernels in your boot menu, continue to the following steps.

For this sort of problem, the troubleshooting will need to be done directly from the rescue CD, so select the "Execute a shell in the installer environment" option. You will then be dropped to a basic BusyBox shell where you can perform some basic diagnostics and repairs.

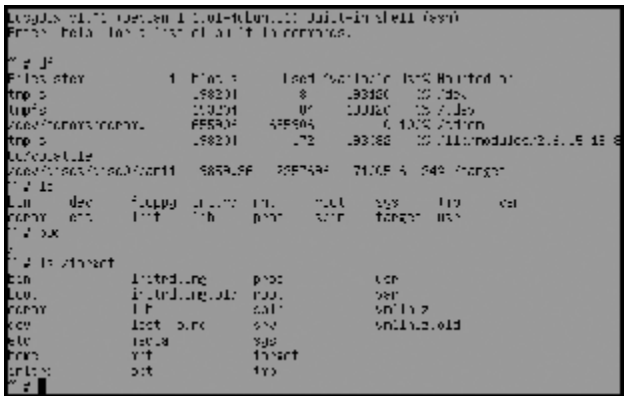
Test the root file system

First, test and see if you can read your default root filesystem. The rescue-mode CD mounted this filesystem at */target* earlier in the boot process. Type:

```
~ # ls /target
```

and see if you get a list of files somewhat similar to what you see in Figure 8-13. If you do, then your root filesystem is in fact mountable so you can skip past the filesystem repair steps to "The system boots and continues through part of the boot process, but hangs at a particular service" at the end of this hack. Otherwise, there's a chance your root filesystem was not unmounted cleanly and has suffered some damage requiring a manual filesystem check. Run *df* to see which partition your root device is on (the one mounted on */target*). Figure 8-13 has sample *df* output you can reference. On a default Ubuntu install, your root filesystem will likely be on */dev/discs/disc0/part1*.

Figure 8-13. Some sample diagnostics from the BusyBox shell



Repair the filesystem

Now use the *fsck* tool to attempt to repair your file system. First, you will need to unmount the */target* partition if it is currently mounted:

```
~ # umount /target
```

The rescue-mode environment has *fsck* utilities only for *ext2*, *ext3*, and *jfs*, so if you chose another filesystem during the install, you will have to resort to a different rescue disc. Ubuntu defaults to *ext3*, so if you went with the default, type the following to check and repair your drive:

```
~ # /sbin/
                    fsck.ext3
                -f
                    /dev/discs/disc0/part1
```

Replace `fsck.ext3` with `fsck.ext2` or `fsck.jfs` if you have an *ext2* or *jfs* partition, respectively, and replace the partition with your particular root partition. Once the *fsck* completes, attempt to mount the partition under */target* again and try to access it with *ls*:

```
~ # mount
                    /dev/discs/disc0/part1
                 /target
~ # ls /target
```

If you see a number of files and directories, congratulations, your drive is repaired! Type `exit` to leave the BusyBox shell and reboot, and see if you can boot into your system successsfully.

The system boots and continues through part of the boot process, but hangs at a particular service

Whether it is due to a poorly written *init* script or some other bug, on some rare occaisions, an *init* script might stall and refuse to continue through the boot process. When this happens, you can't drop to a shell to repair the problem, so you must resort to a rescue CD. Make a note of the service that is stalling, and boot into the rescue mode of your Ubuntu install CD.

To fix this sort of issue, you will need to actually drop to a shell within your root filesystem's environment, so select "Execute a shell in /dev/discs/disc0/part1" from the Ubuntu rescue operations menu. (This option might be named something slightly different if you picked a different partition as your root filesystem.)

This option will drop you to a shell from within your own root filesystem. You can navigate this filesystem and run commands as though you had actually booted into it. At this point, you will want to identify the *init* service that is causing you problems and disable it. "Control Startup Services" [Hack #77] has instructions on how to identify and disable startup services from the command line, so read through that hack for the specifics. In a default Ubuntu install, you will likely want to look for your *init* service in either */etc/rc2.d* or */etc/rcS.d*. Once you find the offending *init* script, rename it, changing the *S* at the beginning of the filename to a *D*.

Once the init script has been disabled, type `exit` to exit out of the shell and then reboot and attempt to boot back into your Ubuntu system. The disabled init script should not stall you this time, so you should be able to get to your default login screen.

Conclusion

It's true that these are only a few of the possible problems that could render a system unbootable. If you have tried all of these rescue options and still can't fix your system, don't give up! Read "Get Help" [Hack #2] for some information on how to enlist the help of some fellow Ubuntu users.

Hack 82. Check the Captain's Log

Find out where Ubuntu logs important system information so you can track down the cause of startup and system errors.

When you use a system long enough, eventually you find you need to put on your detective hat. Maybe you've added a piece of hardware or plugged in a new device, and you aren't sure whether the system is recognizing it. Maybe you upgraded some software, and now it's not working quite right. Whatever the reason, when you want to track down what's really going on under the hood of your Ubuntu system, it's time to look through logs.

A normal desktop system generates a surprising amount of logs in a day, even if nothing is wrong. When you connect to a network, plug in a new device, log in, or do any number of things, the system generates logs. The majority of system logs reside in the */var/log* directory. Some of these logs overlap; for instance, logs from daemons will show up both in the *daemon.log* and in *syslog*. Here are some of the main logs you will find under */var/log*, along with their uses:

syslog

: *syslog* is the primary system log and contains log output from daemons and other programs running on the system, such as *dhclient*, *cron*, *init*, *xscreensaver*, and some kernel logs. This log is the first place to look when trying to track down general system errors.

dmesg

: This log traditionally lists all of the boot-time kernel logging for a system, along with any other kernel logs related to devices and module loading. Check here to see what sorts of devices the kernel detected at boot time, as well as to track down any errors the kernel might have had when loading a module.

kern.log

: Like *dmesg*, this log contains kernel log output, however it has the advantage of being timestamped.

Xorg.?.log

The Xorg logs contain very detailed output from Xorg and are numbered according to which display it is running. The default Xorg session runs on display 0, so for the most part you would look through *Xorg.0.log* to trace down errors in Xorg.

messages

This log contains some kernel log messages along with log output from certain system programs. For instance, *gconfd* and some other programs log here.

daemon.log

Here, you will find the same daemon output that you would normally find in *syslog*, but without log output from kernels or other systems that log to *syslog*.

auth.log

Inside *auth.log*, you will find information about user authentication including every time a user logs in to the system or executes *sudo*.

mail.log

If you use your system as an email server, this log will contain information about incoming and outgoing messages, along with any errors.

apache/

If you have installed the Apache web server on your system, */var/log/apache* contains *access_log*, *error_log*, and all of the other primary Apache logs.

cups/

CUPS is the printing system for Ubuntu. This directory contains all of the different logs for the CUPS service, so look here to debug any issues with printing on the system.

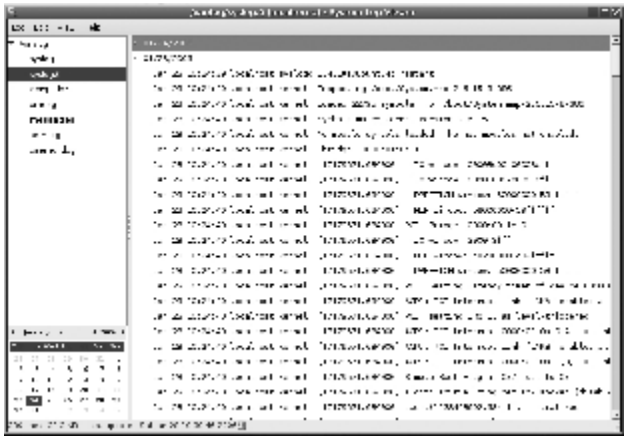
gdm/

GDM is the graphical login manager for GNOME. If you notice any problems with the main login screen, look at the logs in this directory for clues.

View the Logs

Now that you have an idea where to look for information, there are a number of different methods you can use to actually look through the logs. Ubuntu provides a nice graphical tool for log viewing called the System Log Viewer. To start this program, click System→Administration→System Log. The default window displays a sidebar to the left listing various system logs you have open and a main window listing the contents of the selected log. By default, the System Log Viewer lists only *syslog*, but you can click Log→Open to add a logfile to the sidebar (Figure 8-14).

Figure 8-14. Default System Log Viewer window



A nice feature of the System Log Viewer is that it splits logs up according to date. Along the bottom-lefthand side of the window is a calendar with days displayed in bold if the current log contains entries for that day. Select a day from the calendar, and it will appear in the main window. You can also collapse a particular day from view in the main window by clicking the arrow next to the date.

You can also view logs the old-fashioned way from a terminal. The program less is a great, no-frills way to open up a logfile and page through it. Just type:

```
$ less /var/log/syslog
```

to open up the syslog (you can also replace that file with the path of the file you wish to open). You can use the arrow keys to navigate up and down the file, or you can hit G to scroll down to the very bottom of the file, or g to scroll to the top. To search within the logfile for a keyword, type / followed by the keyword to find. The keyword will be highlighted within the terminal, and you can type n to advance to the next match, or N to go back to the previous match. Hit F to have *less* continue to update the log as new lines are appended (a lot like running `tail -f` on the file).

Some people might be tempted to just use *vi* when opening logfiles; however be warned that *vi* caches the entire file to disk when it opens it. This might not be a problem for small logfiles, but when you open a 2 GB Apache log, you might possibly fill up the remaining space on your disk! *less* does not cache the entire file like *vi*, so it is the safer choice for large logfiles.

grep is also a very useful tool for logs. *grep* accepts a pattern as an argument and will look for that pattern within a file. For instance:

```
$ grep dhclient /var/log/syslog
```

will return all of the log entries containing the word “dhclient”. Grep is particularly useful for Xorg logs, as the warnings and errors in these logs are preceded by `WW` and `EE`, respectively. To grep out all of the warnings and errors from *Xorg.0.log*, type:

```
$ grep -E '(WW|EE)' /var/log/Xorg.0.log
```

Note that the `-E` option turned on extended regular expressions so that I could use a more advanced pattern.

Some logfiles you want to view might be *gzip*ped due to log rotation. In this case, use *zless* and *zgrep* just as you would use *less* and *grep*, respectively.

Hack 83. Mount Removable Devices with Persistent Names

Mount your USB drive so it appears as /media/music every time. Mount your FireWire drive so it appears as /media/data every time.

My storage setup on my Ubuntu box at home is a bit unusual. I have an external drive dedicated to music and another external drive that backs up the music drive. I also have an external drive to hold all my personal documents, pictures, and movies, with another external drive to back up that drive. Those four drives are all connected to my desktop via FireWire. Finally, I have an external drive that's a temp drive: when I download a new movie or set of pictures, or rip a new CD, I keep those files on the temp drive until I can properly place them on the music or personal data drive. Unlike the other four, the temp drive uses USB.

Five drives sounds cool, but there's a major annoyance associated with them. When I reboot Ubuntu (a rare occasion, to be sure, but it does happen), their mount points shift around. Sometimes FireWire drive number one gets */media/sdb1*, and sometimes */media/sdc1*. The same thing happens to the other drives as well, which wreaks havoc with my backup scripts and my attempts to SSH in from other machines, to name just two problems I've experienced. I want persistent naming of those drives, so that FireWire number one is always */media/ `something`*, FireWire number two is always */media/ `somethingelse`*, and so on.

After much searching, I found the answer on the Ubuntu forums, at http://ubuntuforums.org/showthread.php?t=91948, and in an excellent overview by Daniel Drake titled "Writing udev rules" (http://www.reactivated.net/writing_udev_rules.html). Basically, Ubuntu includes a technology called *udev*, which manages dynamic devices. I can tell *udev* how I want it to label each separate drive when I plug it in, a far more efficient and usable method than the default. Here's how to do so.

You're going to be editing a file found at */etc/udev/rules.d/10-local.rules*. Check to see if that file already exists on your hard drive:

```
$ ls /etc/udev/rules.d/10-local.rules
```

If it's already there, then you're going to edit it; if it's not there currently, go ahead and create a blank file with the proper name and permissions:

```
$ sudo touch /etc/udev/rules.d/10-local.rules
$ sudo chmod 777 /etc/udev/rules.d/10-local.rules
```

At the end of the process described in this hack, the devices will be listed in the *10-local.rules* file. The USB drive, for instance, will look like this:

```
BUS=="usb", SYSFS{serial}=="6R0000065086", NAME{all_partitions}=="temp"
```

The first part—`BUS=="usb"`—is obvious; that's how that drive connects to the machine. The third part—`NAME{all_partitions}=="temp"`—makes sense too, since `"temp"` is the name I want the drive to use in the future as its mount point, instead of */media/sdb1* or whatever else Ubuntu decides to use. But what about that second part? Where is `"6R0000065086"` coming from?

In order for *udev* to work, it has to know which specific drive I want to label `"temp"`. When I plug in a drive (or any dynamic device), the drive identifies itself in a whole host of ways. I need to pick a unique data point that I can pass to *udev* to use for identification. To see what *udev* sees when you plug in a drive, I run the following command (notice that the path given is the current path that Ubuntu has assigned to the drive—in this case, */dev/sda*):

```
$ udevinfo -a -p $(udevinfo -q path -n /dev/sda)
```

Once I press Enter, an enormous amount of data streams by, looking something like this (this listing has been greatly truncated for length):

```
device '/sys/block/sda' has major:minor 8:0
  looking at class device '/sys/block/sda':
    SUBSYSTEM=="block"
    SYSFS{dev}=="8:0"
    SYSFS{range}=="16"
  SYSFS{removable}=="0"
 SYSFS{size}=="488397168"
    SYSFS{stat}==" 2159 102 16586 1871738 28 17 360 204 0 14257  1871942"

...

  looking at the device chain at
'/sys/devices/pci0000:00/0000:00:1e.0/0000:05:09.2/usb4/4-1/4-1.2/4-1.2:1.0':
 BUS=="usb"
    ID=="4-1.2:1.0"
    DRIVER=="usb-storage"
    SYSFS{bAlternateSetting}==" 0"
    SYSFS{bInterfaceClass}=="08"
    SYSFS{bInterfaceNumber}=="00"
    SYSFS{bInterfaceProtocol}=="50"
    SYSFS{bInterfaceSubClass}=="06"
    SYSFS{bNumEndpoints}=="02"
    SYSFS{modalias}=="usb:v04B4p6830d0001dc00dsc00dp00ic08isc06ip50"

  looking at the device chain at
'/sys/devices/pci0000:00/0000:00:1e.0/0000:05:09.2/usb4/4-1/4-1.2':
    BUS=="usb"
    ID=="4-1.2"
    DRIVER=="usb"
    SYSFS{bConfigurationValue}=="1"
    SYSFS{bDeviceClass}=="00"
    SYSFS{bDeviceProtocol}=="00"
    SYSFS{bDeviceSubClass}=="00"
    SYSFS{bMaxPower}==" 0mA"
    SYSFS{bNumConfigurations}=="1"
    SYSFS{bNumInterfaces}==" 1"
    SYSFS{bcdDevice}=="0001"
    SYSFS{bmAttributes}=="c0"
    SYSFS{configuration}==""
    SYSFS{devnum}=="3"
    SYSFS{idProduct}=="6830"
    SYSFS{idVendor}=="04b4"
    SYSFS{manufacturer}=="Cypress Semiconductor"
    SYSFS{maxchild}=="0"
```

```
    SYSFS{product}=="USB2.0 Storage Device"
    SYSFS{serial}=="6R0000065086"
    SYSFS{speed}=="480"
    SYSFS{version}==" 2.00"

...
```

See the line in **bold** that says `SYSFS{serial}=="6R0000065086"`? That's the unique serial number for this device, so that's exactly what I'll use in the line I'm adding to *10-local.rules*. Notice also that the line `BUS=="usb"` is also present, in case I don't know how */dev/sda* is connected to my box (hey, with five drives, it could happen!). Once again, then, the new line in */etc/udev/rules.d/10-local.rules* is:

```
BUS=="usb", SYSFS{serial}=="6R0000065086", NAME{all_partitions}=="temp"
```

I need to repeat the process for any other drives as well. Although the next four drives are all connected with FireWire, the process is the same; it's just the listings that are different:

```
$ udevinfo -a -p $(udevinfo -q path -n /dev/sdb)

device '/sys/block/sdb' has major:minor 8:16
  looking at class device '/sys/block/sdb':
    SUBSYSTEM=="block"
    SYSFS{dev}=="8:16"
    SYSFS{range}=="16"
 SYSFS{removable}=="0"
 SYSFS{size}=="234441648"
    SYSFS{stat}=="1166 102 8386 739882 27 16 344 52 0 6575 739934"

...

  looking at the device chain at
'/sys/devices/pci0000:00/0000:00:1e.0/0000:05:04.0/fw-host0/0030e0f4e020dca0/0030e0f4e020dca0-0':
 BUS=="ieee1394"
    ID=="0030e0f4e020dca0-0"
    DRIVER=="sbp2"
  SYSFS{address}=="0x0000fffff0000830"
    SYSFS{ignore_driver}=="0"
    SYSFS{length}=="0"
    SYSFS{model_id}=="0x000001"
    SYSFS{model_name_kv}=="OXFORD IDE Device LUN 0 "
    SYSFS{specifier_id}=="0x00609e"
    SYSFS{version}=="0x010483"

  looking at the device chain at
'/sys/devices/pci0000:00/0000:00:1e.0/0000:05:04.0/fw-host0/0030e0f4e020dca0':
 BUS=="ieee1394"
    ID=="0030e0f4e020dca0"
    DRIVER=="unknown"
 SYSFS{bus_options}=="IRMC_0_ CMC_0_ ISC_0_ BMC_0_ PMC_0_ GEN_0_
   LSPD_2_ MAX_REC_64_ MAX_ROM_0_ CYC_CLK_ACC_255_"
    SYSFS{capabilities}=="0x0083c0"
    SYSFS{guid_vendor_id}=="0x0030e0"
 SYSFS{guid}=="0x0030e0f4e020dca0"
    SYSFS{nodeid}=="0xffc0"
    SYSFS{tlabels_allocations}=="1262"
    SYSFS{tlabels_free}=="64"
    SYSFS{tlabels_mask}=="0x0000000000000000"
    SYSFS{vendor_id}=="0x0030e0"
    SYSFS{vendor_name_kv}=="Oxford Semiconductor Ltd.   "
```

```
...
```

In this instance, *udevinfo* tells me how this drive is connected to my Ubuntu desktop: `BUS=="ieee1394"` (remember that IEEE1394 is the official name for FireWire). FireWire drives don't include `SYSFS{serial}` like USB drives do; instead, their unique identifiers are `SYSFS{guid}`. This particular drive holds my personal files, so I want it to show up as */media/data*. Put all that together, and I add this line to */etc/udev/rules.d/10-local.rules*:

```
BUS=="ieee1394", SYSFS{guid}=="0x0030e0f4e020e229", NAME{all_partitions}=="data"
```

After performing the same task three more times, `/etc/udev/rules.d/10-local.rules` looks like this:

```
BUS=="usb", SYSFS{serial}=="6R0000065086", NAME{all_partitions}=="temp"
BUS=="ieee1394", SYSFS{guid}=="0x0030e0f4e020e229",
NAME{all_partitions}=="data"
BUS=="ieee1394", SYSFS{guid}=="0x0030e0f4e020dca0",
NAME{all_partitions}=="data_copy"
BUS=="ieee1394", SYSFS{guid}=="0x0030e0f4e020912c",
NAME{all_partitions}=="music"
BUS=="ieee1394", SYSFS{guid}=="0x0030e0f4e020c727",
NAME{all_partitions}=="music_copy"
```

Now I need to test my changes to make sure they'll actually work. First I need to restart *udev*, the subsystem that is managing all this craziness:

```
$ sudo /etc/init.d/udev restart
```

Now I'll test the rules for the USB external drive to make sure everything will work. To test, I use *udevtest*. The syntax of the command is:

```
$ sudo udevtest
                ***sysfs_device_path***


                ***subsystem***
```

How do I know what the *sysfs_device_path* and *subsystem* are? Look back when I ran `udevinfo -a -p $(udevinfo -q path -n /dev/sda)` and notice the first things reported back to me:

```
looking at class device '/sys/block/sda':
  SUBSYSTEM=="block"
```

My *udevtest* command therefore looks like this:

```
$ sudo udevtest /sys/block/sda block
```

The results appear immediately below it (some error messages about other items in *udev* that aren't important here have been removed):

```
udevtest.c: looking at device '/block/sda' from subsystem 'block'
...
udevtest.c: opened class_dev->name='sda'
udev_rules.c: configured rule in '/etc/udev/rules.d/10-local.rules:1' applied, 'sda' becomes 'temp'
udev_add.c: creating device node '/dev/temp', major = '8', minor = '0', mode = '0640', uid = '0', gid = '46'
udev_add.c: creating device partition nodes '/dev/temp[1-15]'
```

Excellent! Now I know that *udev* would in fact create a device name */dev/temp*—and therefore a mount point at */media/temp*—instead of */dev/sda* and */media/sda*, and will always use that in the future. By the way, don't worry—even though the output says it was creating nodes, it really wasn't. It was just showing me what it would do when it runs for real.

So I know the USB drive works, but what about one of the FireWire drives?

```
$ sudo udevtest /sys/block/sdd block
```

Here are the results for that drive (again, some unnecessary data has been removed):

```
udevtest.c: looking at device '/block/sdd' from subsystem 'block'
...
udevtest.c: opened class_dev->name='sdd'
udev_rules.c: configured rule in '/etc/udev/rules.d/10-local.rules:5' applied, 'sdd' becomes 'music_copy'
udev_add.c: creating device node '/dev/music_copy', major = '8', minor = '48', mode = '0640', uid = '0', gid = '
udev_add.c: creating device partition nodes '/dev/music_copy[1-15]'
```

Exactly what I wanted, so it looks like everything will work as intended. Now I restart the machine, and lo and behold, my drives appear, just as I wanted them:

```
%(path)/media/temp%
%(path)/media/data%
%(path)/media/data_copy%
%(path)/media/music%
%(path)/media/music_copy%
```

Of course, there's nothing stopping you from setting up *udev* so that your digital camera always shows up as */media/camera*. Or so your iRiver music jukebox consistently mounts as */media/iriver*. And so on. Basically, if your system sees it as a dynamic device, you can use *udev* to map it to a specific mount point. Thanks to *udev*, Ubuntu just got a lot easier to use.

Scott Granneman

Hack 84. Mount Remote Directories Securely and Easily

If you can access it via SSH, you can mount it on your filesystem, giving you secure access to your favorite stuff.

Samba has transformed how Penguinistas interact on networks with Windows machines and even other Linux boxes. In particular, I often use *smbfs* (short for "Samba filesystem") to mount a Samba share on my box for easy access. For example, I have two computers on my network: a server named *sophocles* and a laptop named *euripedes*. A huge collection of music can be found on *sophocles* at */var/music*, but I want easy

access to those goodies from *euripedes*. Using Samba on *sophocles*, I share */var/music*, and using *smbfs* on *euripedes*, I mount */var/music* on my server *sophocles* to */home/scott/tunes* on my laptop *euripedes*. By doing so, it appears to me while I'm using *euripedes* as though *tunes* is just another local folder on my laptop, so I can read files from it, save to it, and do anything else I could if that folder were in fact on my machine.

AU: Euripides is spelled incorrectly in the hard drive name throughout the hack (as "euripedes). Do you want to change it throughout, or leave as is?

This is great, except that there are some issues. Setting up Samba can be a royal pain, so any time I can use something simpler, I jump at the chance. Second, Samba shares aren't secure by default. Call me paranoid, but I don't like anything flowing over a network that isn't encrypted. Yes, it's possible to tunnel Samba using SSH, but that just adds more time and trouble on top of the royal pain that Samba sometimes causes. Finally, Samba was designed for LANs, not the wild and wooly Internet, so accessing shares remotely is out of the question (yes, there are ways to do it, but it's just not a good idea on today's Net...and it causes yet more complication!).

But I'm here to tell you that there's a better way: *sshfs*, the SSH filesystem. Instead of Samba, it uses SSH. All the problems I listed previously are obviated when you switch to SSH. SSH is a breeze to set up and use. All SSH traffic is encrypted, hence the name Secure SHell. And SSH was designed for use on LANs as well as the big, bad Internet, so it's a great choice no matter where you are.

To go back to my original example, I can SSH from *euripedes* to *sophocles* (and vice versa, for that matter), so I now use *sshfs* to mount */var/music* on *sophocles* to */home/scott/tunes* on *euripedes*. It's easy, it's encrypted, and if this is a connection I'm going to need all the time, I can set things up in */etc/fstab* to automate the whole process. Ready to try it out?

Before doing anything else, make sure that you can *ssh* from the client to the server:

```
$ ssh scott@sophocles
Linux sophocles 2.6.15-18-386 #1 PREEMPT Thu Mar 9 14:41:49 UTC 2006 i686 GNU/Linux

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
Last login: Sun Mar 12 13:59:45 2006
$ exit
```

If you can't SSH in, you need to set that up:

- On Windows, install Cygwin, available from http://www.cygwin.com, and install OpenSSH as a service.
- On the Mac, enable Remote Login in System Preferences→Sharing→Services.

If you can successfully SSH in to the server, you're finished with that machine (see how much simpler this is than it would be if you were using Samba?).

Now you need to focus on the local computer. Make sure you've enabled the *universe* archive in your */etc/apt/sources.list* file, as detailed in "Modify the List of Package Repositories" [Hack #60]. Once you've done so, install *sshfs*:

```
$ sudo apt-get install sshfs
```

You'll be told that *apt* is going to have to install some additional packages in addition to *sshfs* (*fuse-utils* and *libfuse2*), so go ahead and approve *apt*'s request.

When *apt* finishes installing your new software, it's time to fix some permissions so that normal users can mount and unmount using *sshfs*. If you leave this step out, mounting and unmounting will require the use of *sudo* and knowledge of the *root* password, which you probably don't want. Better to go ahead and run this command, which grants all users on your system execute permission for the *fusermount* command, used by *sshfs* to perform its magic:

```
$ sudo chmod +x /usr/bin/fusermount
```

Now you need to make the mount point you plan to use on your local machine. In my case, this would be */home/scott/tunes*:

```
$ mkdir /home/scott/tunes
```

In order for *sshfs* to work, you have to load the *fuse* module. For now, go ahead and run this command:

```
$ sudo modprobe fuse
```

I know you don't want to have to run that command every time you want to mount with *sshfs*, so use the following command to tell Ubuntu to automatically load the *fuse* module when you boot:

```
$ sudo sh -c "echo 'fuse' >> /etc/modules"
```

Be very careful and use >> instead of > (in other words, append instead of overwrite)! If you accidentally use >, you will hose your box and none of your modules will load on boot. For that reason alone, you really should back up up */etc/modules* first, with `sudo cp /etc/modules /etc/modules_bak`.

Now for the final step: mount */var/music* on *sophocles* to */home/scott/tunes* on *euripedes*. The syntax of the command is as follows:

```
$ sshfs
        user@[IP|HOSTNAME of remote machine]:/shared/path /local/mount/point
```

If your username is the same on the remote and local machines, you can leave it off. DNS will work either on your LAN, if you have an entry for the server in your */etc/hosts* file, or over the Net to a machine with a registered address. Otherwise, use the computer's IP address. If you don't specify a path after the colon, *sshfs* mounts the home directory for the SSH user, which may be all that you want. In my case, my username is the same on both boxes, I have an entry for *sophocles* in the */etc/hosts* file, and I want to mount a specific directory on the computer, so my command looks like this:

```
$ sshfs sophocles:/var/music /home/scott/tunes
```

If you get the error "fusermount: failed to open /dev/fuse: Permission denied", add yourself to the `fuse` group with `sudo addgroup username fuse`, log out, log back in again, and try the command again.

Now I can open my favorite music player, point it to */home/scott/tunes*, and start enjoying jazz, rock, country, or whatever it is that floats my boat. Remember, it's all secure, so I don't have to worry about anyone sniffing packets and finding out what's moving around on my network.

To unmount, you have a choice. You can use the *fusermount* command with the `-u` option (for "unmount"):

```
$ fusermount -u /home/scott/tunes
```

Or it might be simpler just to use the good ol' *umount* command:

```
$ sudo umount /home/scott/tunes
```

If I knew that I wanted to make my connection to *sophocles* whenever I booted, I could add the following to */etc/fstab*:

```
sophocles:/var/music /home/scott/tunes fuse defaults 0 0
```

The first three items are different from the usual *fstab* entries, with the first two exactly like the *sshfs* command you just used, and `fuse` indicating to your computer the kind of filesystem it's going to mount. If you want to change the other entries and you know what you're doing (run `man fstab` to learn more), go right ahead.

Once you discover *sshfs*, you're going to use it all the time. There's no reason not to: it works beautifully, and the added benefit of security is icing on the cake. Thank you, *sshfs*!

Scott Granneman

Hack 85. Make Videos of Your Tech-Support Questions

Have a friend or relative with tech-support troubles? Show them how to record a movie of their Ubuntu problems and send it to you, you ever-helpful, all-knowing Ubuntu guru, you.

We've all been there: Uncle Gussy is on the phone to you trying to explain a computer problem he's having, but his description isn't helping in the slightest: "Wellllll, I click on the thingie, and then this doohickey opens, so I click on that, and then the computer does something weird." Gee, thanks, Uncle Gussy!

If you were smart enough to set Uncle Gussy up with Ubuntu (and being a good niece or nephew, that's what you did, right?), you could just tell him to use Istanbul to record what he's doing on his computer into a patent-free, open source Ogg Theora video file. Uncle Gussy could then email you the file, which would vastly simplify your job as family tech support.

To find out more about Istanbul, or to keep up with its development, check out http://live.gnome.org/Istanbul. Don't know what Ogg Theora is, or are only familiar with Ogg Vorbis? Then go read Wikipedia's article on the subject, at http://en.wikipedia.org/wiki/Ogg_Theora.

When you first set up Uncle Gussy's Ubuntu box, make sure Istanbul is installed. First look for it, to see if it's already on the machine:

```
$ whereis istanbul
istanbul:
```

Nope. OK, you need to put it on there. Make sure you have the universe repository enabled [Hack #60], and then install Istanbul:

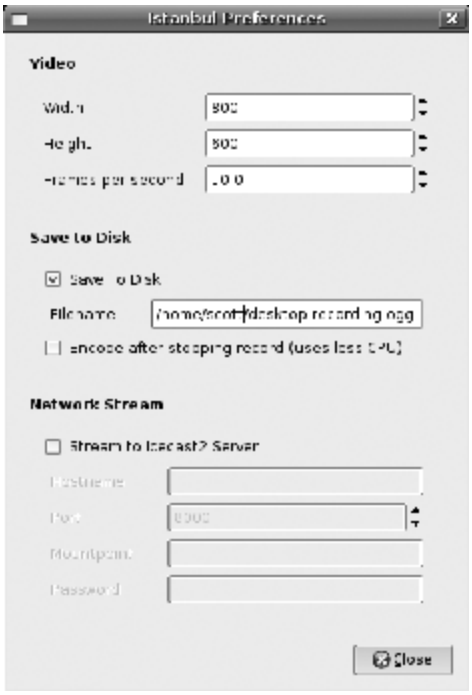
```
$ sudo apt-get install istanbul
```

You may find that *apt* wants to install several other packages as well, chiefly related to *gstreamer*, so go ahead and indicate your approval. Once *apt* has finished its business, you need to start Istanbul so you can configure it, so select Applications→Sound & Video→Istanbul Desktop Session Recorder. You'll know Istanbul has loaded when you see a large red dot to the left of the clock on the top panel, as shown in Figure 8-15.

Figure 8-15. Istanbul on the top-edge panel

Right-click on Istanbul's icon and choose Preferences, which brings up the window shown in Figure 8-16.

Figure 8-16. Istanbul preferences



It's a good idea to check the box next to "Encode after stopping record (uses less CPU)" for the reason given in the setting's name. Also, to reduce file size, change Frames Per Second from `10.0` to `1.0`. Of course, this also means that Uncle Gussy will have to move slowly and deliberately as he records his movie, but that can only help you when you're viewing it. As for the Network Stream section, that would more than likely be far too complicated for the likes of Uncle Gussy, so just leave those prefs alone. Click Close, head on home, and then wait patiently until the next phone call from Uncle Gussy.

You should test Istanbul before Uncle Gussy uses it, to make sure it will work correctly. If Istanbul doesn't quite work, try playing with the Preferences. In particular, uncheck "Encode after stopping record," which sometimes fixes otherwise inexplicable problems with Istanbul. This will take up more CPU, since Istanbul will encode the file on the fly if you uncheck this box.

The Ogg Theora encoder loves to gobble up CPU cycles, so whatever you can do to reduce that load would be good. Checking the box next to "Encode after stopping record" was a good start, but you should also walk Uncle Gussy through temporarily changing his screen resolution to 800×600 if at all possible.

Now for Uncle Gussy's debut as Cecil B. DeMille. Tell him to start Istanbul by going to Applications→Sound & Video→Istanbul Desktop Session Recorder. To start recording, he clicks on the large red circle, which changes to a stop box to indicate that recording is in progress. He then performs the actions he wants you to watch and decipher. Tell him to go slow and make all his actions deliberate, so that you can easily comprehend what he's doing.

When he's finished, he clicks on the stop box, and the icon changes once again, to a gray icon that GNOME uses to represent a hard drive, to let him know that Istanbul is encoding. If your uncle has a fast computer, or his movie is short, he may not even see that last icon, in which case he'll be back to the red circle. If Uncle Gussy is done with Istanbul, he can close the program by right-clicking on the icon and choosing Quit.

There should now be a new file named *desktop-recording.ogg* in Uncle Gussy's home directory. Tell him to email that to you, and shortly thereafter you'll be able to see exactly what's going on with his Ubuntu machine. It's the next best thing to sitting next to him while he screws up—I mean, *uses*—his computer.

You may find that Istanbul is still a bit buggy, but it's getting a lot of attention from developers, so keep an eye on it. More and more, developers of various Linux distros are going to rely on Istanbul when users complain about various bugs and problems.

Istanbul is supposed to work with KDE, and therefore Kubuntu, but it's impossible at this time to access the Preferences menu by right-clicking on the Istanbul icon. Other than that, the program works, so make any changes to Preferences while in GNOME and then switch over to KDE to actually use the program.

Scott Granneman

Hack 86. Synchronize Files Across Machines

If you switch between multiple machines, it's a real hassle to keep track of which system has the latest version of a file. Use the free Unison File Synchronizer to keep all your systems in sync, even Windows and Mac machines.

There's an old saying that everyone should take to heart: "There are two kinds of computer users: those who have lost data and those who are about to." I belong in that first camp, but it only took one nasty experience with an accidentally deleted term paper to force me to resolve never to let that happen again. That night long ago, when I lost the fruits of my labors analyzing the great film *Raising Arizona*, taught me the hard way that backing up is essential. And while traditional backups [Hack #79] are vital, keeping two or more machines in sync not only gives you some extra security, but it also makes it easy to switch from system to system without having to remember which one has the latest copy of a file.

If a user new to Linux starts asking more experienced users about a good way to sync his data, he will soon hear about a wonderful tool named *rsync*. *rsync* was developed by Andrew "Tridge" Tridgell, the same man behind Samba; in fact, Tridge has stated that he believes that he'll be remembered through posterity for *rsync* far more than for Samba, and he just may be right. However, *rsync*, while absolutely excellent, is not necessarily the best software to use for syncing two repositories that you're actively working with.

The Problem with rsync

rsync is truly awesome software, and it really is worth your time to check it out. However, I don't use *rsync* for my day-to-day sync needs. Instead, I use Unison, which uses *rsync* as its base. Why Unison instead of *rsync*? Because Unison synchronizes in two directions at one time, while *rsync* goes only in one direction.

Let me give another example, from my own setup. At home I have a desktop machine, but I also use a laptop. I work a lot outside of my home, so my laptop travels with me constantly; when I'm home, though, I sometimes leave my laptop in my backpack and use my desktop instead. I have a large amount of data that I need to have available to me at all times: web pages I've read, instruction manuals, articles I'm working on, photos, and so on. All told, it's about 10 GB worth of stuff. I keep all of this work on my desktop, and I keep a mirror of the same thing on my laptop.

Obviously, I need to keep all 10 GB in sync between my two machines. If I delete a file on the desktop, I need to delete it on the laptop. If I change a file on the laptop, it wouldn't do to open the original, unchanged file on the desktop and make changes there. That way lies madness.

So why can't I use *rsync*? Because my goal is to synchronize between two machines, each of which acts as a source and a destination at the same time. Here's one example: on my laptop, I move the file *widgets* from directory *foo* to the directory *bar*. On the desktop, however, *widgets* is still in *foo*. If I run rsync with the laptop as the source and the desktop as the destination, so far so good: *widgets* is now in *bar* on both the laptop and the desktop. However, *widgets* is still in *foo* on the desktop. The next time I run rsync with the desktop as the source and the laptop as the destination, *widgets* gets copied over from *foo* on the desktop back into *foo* on the laptop, leaving me with *widgets* in both *foo* and *bar*, which produces a real mess.

Of course, I could run rsync with the `--delete` option set, so that files deleted on the source are also deleted on the destination, but that can be a very dangerous practice. What if I delete *foo* on the laptop but *change foo* on the desktop? If I run rsync with the laptop as source, then *foo* is deleted on the desktop, which is not what I want. Instead, I have to remember to run rsync with the desktop as source so that the changed *foo* is copied over to the laptop. But what if there were files I changed on the laptop and deleted on the desktop? Then I need to run rsync with the laptop as source...and on and on, ad infinitum, back and forth.

Unison solves this problem. You define a source and a destination, and then you run Unison. After some length of time, Unison starts asking you questions about the files it has found: copy this file from laptop to desktop? Copy this other file from desktop to laptop? Delete this file on the laptop since it was deleted on the desktop? You can accept Unison's guesses, specify a direction for the copy to go, or even tell Unison to skip the file entirely until another day. With Unison, synchronizing two directories, each on a different machine, becomes a far simpler task. And, as icing on the cake, Unison is cross-platform, as it runs on Linux, Mac OS X (with a few caveats; see "Further Information About Unison" at the end of this hack), Unix, and Windows machines.

Synchronize Files on Two Machines with Unison Using SSH

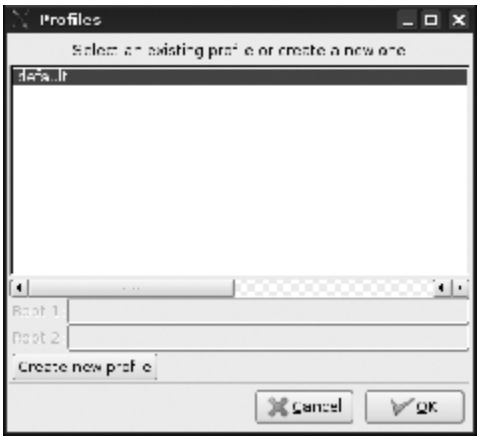
Installing Unison is easy as pie (however, make sure you have the universe repository enabled [Hack #60]). Just run the following command, and you're done:

```
$ sudo apt-get install unison unison-gtk
```

Now you have both the basic, command-line-only Unison program and the GTK-based GUI.

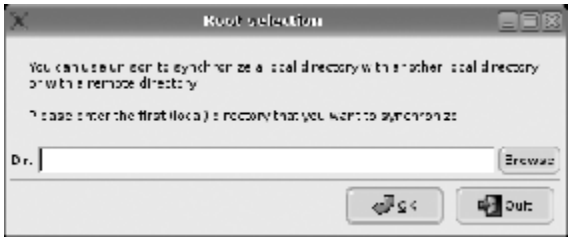
It's time to sync a group of files with another copy of those same files on another machine. For added safety, I'm going to use SSH for the transport, a capability built into Unison. Unfortunately, Unison doesn't create entries in the GNOME Applications menu or the KDE K menu, so you'll need to add those yourself. In the meantime, you can type `unison-gtk` on the command line. When the program opens, you'll see Figure 8-17, which asks you to either select or create a profile.

Figure 8-17. Select or create a Unison profile



Click Create New Profile, and you're asked to give a name for the new profile. Since I'm going to sync up my folder that contains the work I've done on this book, I'm going to cleverly name this profile *ubuntu_hacks* and click OK. Now both profiles, *default* and *ubuntu_hacks*, appear. I double-click on *ubuntu_hacks*, and a small window opens, shown in Figure 8-18, asking me to enter the local directory that I want to sync.

Figure 8-18. Select the local directory you want to sync



Since I don't like to type when I can avoid it, I instead click Browse and navigate to the folder:

```
/home/rsgranne/documents/clientele/aaa_current/OReilly/ubuntu_hacks
```

I click OK, and another window opens, shown in Figure 8-19. This one wants to know about the remote folder.

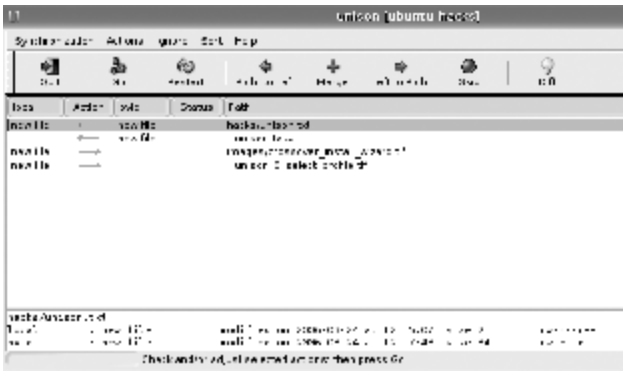
Figure 8-19. Select the remote directory you want to sync

Notice that I can sync to a local folder (which could also be a folder accessed through Samba or NFS that I've mounted on this machine), or to another machine via SSH, RSH, or a socket. In this case, I'm using SSH, so I enter the address of the remote host—in this case, 192.168.0.14—and the username, which is *rsgranne*. Finally, I type in the remote directory I want to sync:

```
/media/data/zzz_rsg/documents/clientele/aaa_current/OReilly/ubuntu_hacks
```

I click OK, and since this is the first time I've synched these two folders, Unison presents me with a warning that it is going to have to create archives for those two folders from scratch. No problem, so I click OK. The main Unison window opens, and Unison goes to work, comparing the two directories (and all their subdirectories as well, since the program automatically recurses). Be patient during this process. Unison may appear to be locked up, but it's just taking its time to digest all the files and folders. The more data, the longer Unison will take. Eventually Figure 8-20 appears.

Figure 8-20. You can sync files in a number of directions



Unison's interface is pretty self-explanatory, with the arrows showing the direction in which copying will occur. If I want to switch the direction in which a file is copied, I simply select it and then click the "Right to Left" or "Left to Right" button. If I want Unison to ignore a file, I click Skip. If I want to invoke a *diff*-type program to actually compare the contents of the two files (this obviously works only with text files and the like) and then selectively merge the differences, I click Diff and then Merge. When I'm sure that everything is the way I want it, I click the Go button to tell Unison to copy files and folders. There are actually lots of other options available through the Actions menu, so it's a good idea to check it out.

Tweaking Your Profiles

When I first started Unison, I had to create a profile, which keeps track of the directories I want to compare and the transport method, among other configurations. Profiles live in a hidden directory (*.unison*), which itself is in your home directory, and they're just text files that end with the *.prf* extension. Before I finish looking at Unison, I want to mention a few other tweaks that you should add manually to those files in order to make your sync processes work a little more smoothly.

In order to add these tweaks, you're going to need to open your profiles with your favorite text editor. In my case, I'm going to open and change *ubuntu_hacks.prf*, located in */home/rsgranne/.unison*.

To start, I'll add the following to the file (it doesn't matter where you put these preferences, but I like to put them at the top of the file, after my two roots and before any "ignore" lines):

```
times = true
```

If you use Unison for any length of time, you're going to notice that a lot of files need their "props" synchronized. Complaining about "props" is just Unison's way of telling you that the date- and timestamps for your files—which indicate when the files were modified—are not exact. By setting `times` to `true`, you are copying over not just the contents of the files, but their modification dates and times as well, which will vastly reduce any "props" complaints that you'll see.

This setting copies over modification timestamps for files only, not directories.

In a similar vein, you may want to add the following:

```
owner = true
group = true
```

When you add these two settings to your profile, you order Unison to synchronize the owners and groups of files, along with their contents, which can really help to keep things straight as you move files around between machines.

If you want to synchronize users and groups by the real numbers that your Ubuntu system uses instead of the names that we humans use (in other words, by *1000* instead of *rsgranne*; if you don't know what I'm talking about, take a look at your */etc/passwd* file and run the command `man 5 passwd`), then add this preference to your profile as well:

```
numericids = true
```

If you're using Unison to sync only Linux boxes, then you can skip this next preference. But if you're involving Windows machines in your sync—say, from a Windows workstation to a Linux machine, or from a Windows PC to a Linux workstation—then you probably want to use `fastcheck` (but read the following paragraphs carefully):

```
fastcheck = yes
```

You'll probably want to use `fastcheck = yes` most of the time, but every once in a while you'll want to comment out that line with a pound sign in front of it (`#`), run Unison, and then remove the pound sign. Let me explain why.

You may already have asked yourself a pretty important question: how the heck does Unison know that a file has changed? For files on a Linux box, Unison looks at their inode numbers and modtimes; if either of those is different, then Unison thinks that the file has changed. Windows boxes, of course, don't have inode numbers, so Unison's default for Windows is to scan the contents of each file to see if anything has changed. This is certainly a safe method, but if you have a lot of files to scan on a Windows machine, it's going to take a long, long time. Failing to set the `fastcheck` preference then, or setting it to `fastcheck = default`, results in this behavior, which probably is not what you want to do.

Don't know what an inode number is? Here's a handy explanation: http://en.wikipedia.org/wiki/Inode.

If you set `fastcheck` to `yes`, however, then Unison acts differently on your Windows machines: it looks at the file modification times instead of scanning the contents of each file (Linux files still get the same treatment: their inode numbers and modtimes are checked). This results in a much faster scan of your Windows machine, but in very rare circumstances (a file has been changed, but you have somehow managed not to change the create time, the modification time, and the length of the file—pretty hard to do), Unison may refuse to make an update of some of the files on the Windows machine.

Now, practically speaking, you don't have to worry about this very much. First, you'd have to really jump through hoops to fool Unison, and second, even if Unison thought that it should possibly overwrite such a file, it won't, as the Unison manual explains: "Unison will never overwrite such an update with a change from the other replica, since it always does a safe check for updates just before propagating a change."

Since this is the case, I would recommend leaving `fastcheck` set to `yes` most of the time, but, every once in a while, comment out that preference with a pound sign, so that the full contents of your Windows files will be checked on the next run. Then, after you've satisfied yourself that things are copacetic, uncomment `fastcheck = yes` and go back to the faster method.

Changing the Location of Your Logfile Directory

When you run Unison, the program generates a logfile, which can be a very useful thing. The default, however, is to create this logfile—named *unison.log*—in your home directory, which I find annoying since I try to keep my home directory neat and tidy. If you don't mind this, then leave the logfile preference alone. But if you, like me, want to change the location in which the logfile goes, then set this preference:

```
logfile = $HOME/.unison/unison.log
```

I like the logfile to go into the hidden *.unison* directory, but you can change this any way you like.

Further Information About Unison

The main site for all things Unison can be found at http://www.cis.upenn.edu/~bcpierce/unison/. Hopefully, it'll get a cooler URL someday.

On the main Unison site, you can read the very complete and very informative Unison documentation (if you really get into Unison, then you should look into running it sans GUI; the documentation will tell you everything you need to know, although the GUI actually provides a good way to learn the basics). If that doesn't help, there is a good FAQ that covers several important questions, especially one that I was wondering myself: "Does Unison work on Mac OS X?" Be sure to read that answer in full before trying to use Unison on that OS!

For further support, there are a few listservs you can join—or just search, if you don't want to join—at Yahoo! Groups. *Unison-users* is just that: a list for users of Unison. It currently has over 600 members, and they produce in the neighborhood of 100 messages a month, so it's not too overwhelming. You can get more information about the list at http://groups.yahoo.com/group/unison-users/. If you're a developer, then you might want to check out *Unison-hackers*, at http://lists.seas.upenn.edu/mailman/listinfo/unison-hackers. If all you want are the announcements when a new version is released, then you should subscribe to the low volume (one message per month) *Unison-announce*, which you can find at http://groups.yahoo.com/group/unison-announce—or just send a blank email to unison-announce-subscribe@groups.yahoo.com.

Finally, Open magazine, a "weekly e-zine for Linux and Open Source computing in the enterprise," has a nice article about Unison available at http://www.open-mag.com/features/Vol_53/synch/synch.htm. They focus on getting Unison to work with Windows as well as Linux, and provide some technical details about fail-safe provisions that Unison provides.

Scott Granneman

Chapter 9. Virtualization and Emulation

ED: Need Introduction for this chapter

Hack 87. Run Windows Applications

If the Windows applications you need to run just happen to be supported by Wine, you won't need to dual-boot or run an emulator. Just run the installers and launch the programs as if they were any other Linux app.

Novell, the company behind the SUSE distribution, recently ran a survey asking people which Windows applications they would most like ported to Linux. Adobe's Photoshop (number 1), Dreamweaver (3rd), and Flash (5th) all ranked very high (you can see the complete poll results here: http://www.novell.com/coolsolutions/feature/16917.html). Will Adobe be porting these much-requested apps to Linux soon, in response to the survey results? Probably not. But that doesn't mean you can't run them on Linux.

You can run all three of these applications, along with many more, thanks to Wine, an open source Windows compatibility layer. *W*ine *i*s *n*ot an *e*mulator (hence the name, in true GNU recursive style), but it does provide an alternative, 100-percent-non-Microsoft implementation of the DLLs that Windows programs use. (Wine can also use actual Windows DLLs as well.) The software has been in development for more than 12 years and just reached beta status in the fall of 2005. But Wine has been in widespread use for quite some time, and it's included in most distributions, including Ubuntu (in the universe repository [Hack #60]).

Install Wine

The Wine software included with Ubuntu is frequently at least a step behind the current version, so to run the latest version you'll want to edit your */etc/apt/sources.list* file and add Wine's own *apt* repository. You can do so manually, or with the Synaptic Package Manager.

To add the line yourself, open Terminal and enter this command:

```
$ sudo nano /etc/apt/sources.list
```

After you furnish your password, the *nano* editor will open *sources.list*. Enter this line at the end of the file:

```
deb http://wine.sourceforge.net/apt binary/
```

Save the file (press Ctrl-O), then open Terminal again and run:

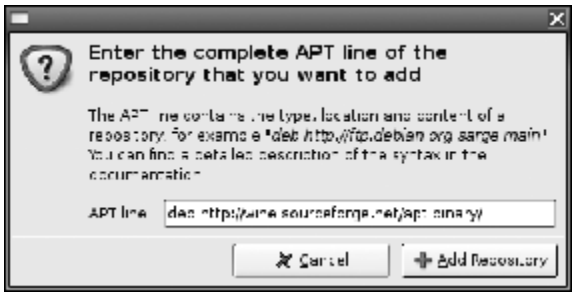
```
$ sudo apt-get update
```

This will update the package cache. Now you can install Wine with the command:

```
$ sudo apt-get install wine
```

If you prefer to add the Wine repository and install Wine with Synaptic [Hack #56], click System→Administration→Synaptic Package Manager and select Settings→Repositories. Then click Add, select Custom, and enter `deb http://wine.sourceforge.net/apt binary/` in the APT Line field, as shown in Figure 9-1.

Figure 9-1. Adding the Wine repository with Synaptic



Next, click Add Repository, and then click Reload to update the package cache. Now search for *wine*, highlight it, right-click the highlighted entry, and select Mark for Installation. Then click Apply, review the proposed changes, and click Apply again. Wine will proceed to download and install the latest version (0.9.10 as of this writing).

Wine is very much a work in progress, and each new version shows improvement, even if it sometimes temporarily breaks compatibility with Windows programs that formerly worked. The newer Wine is, the better it is, unlike its vintage namesake. Overall, each new version tends to be more stable and run more Windows software than its predecessor, so it's definitely worth running the latest version.

Configure Wine

The first thing you will want to do with Wine is configure it. You do so by running the Wine configuration utility, *winecfg*. In Terminal, enter:

```
$ winecfg
```

Running this command will create the Wine configuration directory in */home/username/.wine*. It will also bring up the tabbed Wine configuration interface shown in Figure 9-2, which lets you adjust various parameters of your Wine installation, including the version of Windows that you want Wine to behave like. It's generally fine to accept the defaults, but you may find it helpful to add a new Windows drive (via the Drives tab) that explicitly maps to your CD-ROM drive. To do so, click the Add button to create a new Windows D drive, and then click Browse to select the path to your CD-ROM (such as */media/cdrom0/*). Finally, click the Apply button to finish.

Figure 9-2. The Wine configuration interface

There's one more thing you should attend to before you begin to install Windows apps, and that is to install Microsoft True Type fonts (so applications running in Wine can render text correctly) and *cabextract*, a useful tool for extracting the contents of Microsoft *.cab* archives. You can install both packages via *apt* by issuing this command in Terminal:

```
$ sudo apt-get install msttcorefonts cabextract
```

You can also choose to install with Synaptic by searching for the *msttcorefonts* package, and then installing it. Synaptic will install the *cabextract* package along with *msttcorefonts*.

Install a Windows Application

OK, now that you have Wine, `msttcorefonts`, and `cabextract` installed, you can install your first Windows application. If you have it, you may as well choose Photoshop, since it is currently the most requested app for porting to Linux (actually, I think the GIMP is more than fine for most web graphics work, but I'll defer to popular opinion here).

Wine won't always run the current version of a Windows application; often, there will be some catching up to do. So, for example, the current version of Photoshop is not likely to work with the current version of Wine (although, at some point in their respective development cycles, it may). Usually you will run Windows apps that are a generation or two behind. This is what I did with Photoshop. Since I tend to run that application more on OS X, my Windows version (Photoshop 6.0) is older. Photoshop 6 is no slouch in terms of functionality,

though, and you can use these instructions to install a more recent version. Photoshop 7 is widely reported to work well, and it's possible you'll be able to get Photoshop CS working, too.

Pop the installation CD for your Windows app in your drive. Open Terminal and type the command `wine`, followed by the path to the installer. For example:

```
$ wine "/media/cdrom0/Adobe Photoshop 6/Setup.exe"
```

That's it. If you're using an application that works with Wine, the installer will launch and install the application into the mock Windows filesystem in your *.wine* directory, as shown in Figure 9-3.

Figure 9-3. Installing Photoshop on Ubuntu

Run Windows Applications

Now that you've installed a program, you can run it—but you have to find it first. For example, my Photoshop installation is located in the */home/username/.wine/drive_c/ProgramFiles/Adobe/Photoshop6.0/* directory. To launch Photoshop, I simply type the following command in Terminal:

```
$ wine "c:\\Program Files\\Adobe\\Photoshop 6.0\\Photoshp.exe"
```

Photoshop will launch almost instantly (see Figure 9-4). And this old version runs *fast*—probably faster than it did/does on Windows. It also runs very well; I have yet to encounter any glitches.

Whenever you have questions as to whether a particular Windows application will work with Wine, you should consult the Wine Application Database at http://appdb.winehq.org/. Frank's Corner (http://www.frankscorner.org/) is another good resource for checking whether your application is likely to work.

Figure 9-4. Photoshop running on Ubuntu

Working with Wine

When Wine works really well, there's no better way to run a Windows application in a Linux environment. Many people dual-boot, of course, but that is really cumbersome when all you want to do is use a particular application. Other people run Windows itself using emulation software, but that is a less well-integrated solution, more resource-intensive, and generally not as fast. Wine enables Windows apps to run at native speed (at least) and to interact seamlessly with Ubuntu. Photoshop (along with other Windows apps) can open and save files anywhere on your computer, with no special effort required (there's no need for Samba-based file sharing between Ubuntu and your Windows environment, for example). In addition to the mock C drive installed by Wine, there is a Z drive that maps to your Linux filesystem.

Running Wine on your Ubuntu system is very straightforward. You've already seen how to install and then launch Windows applications. Some other basic Wine commands to keep in mind include:

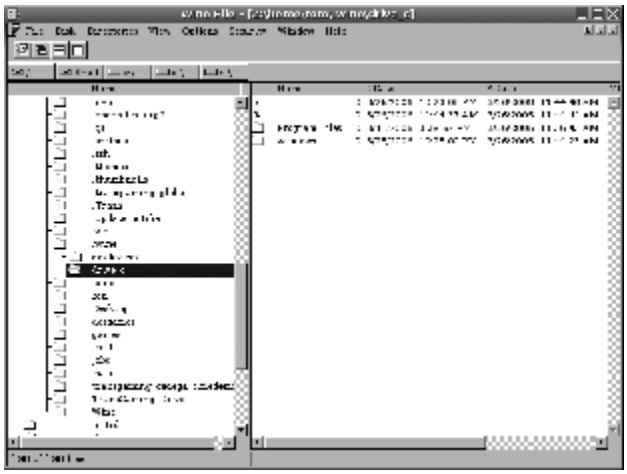
`wine control`

Some programs (such as QuickTime) install associated control-panel applets; `wine control` will let you access them.

`winefile`

Running the `winefile` command will bring up a graphical interface, shown in Figure 9-5, that enables you to view your drive configuration and locate files. You can also run programs directly from this graphical environment; simply double-click on an *.exe*, just as in Windows.

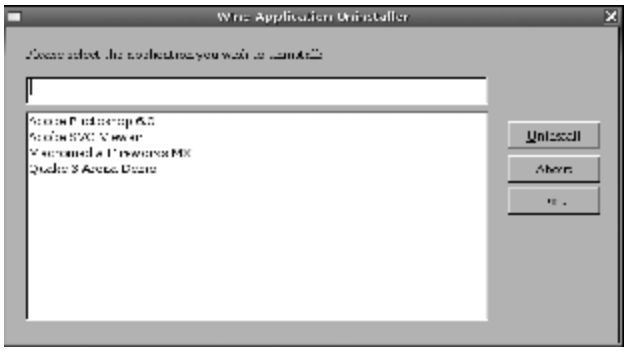
Figure 9-5. Wine File, an Explorer-like graphical environment



`uninstaller`

The `uninstaller` program, shown in Figure 9-6, is similar to the Windows Add/Remove Programs control-panel applet. Simply highlight the program you want to remove and click Uninstall. It's that straightforward.

Figure 9-6. Uninstaller gives you an easy way to remove programs



You can access a complete set of Wine documentation at the Wine HQ site: http://www.winehq.org/site/docs/wineusr-guide/index.

Running Other Windows Components

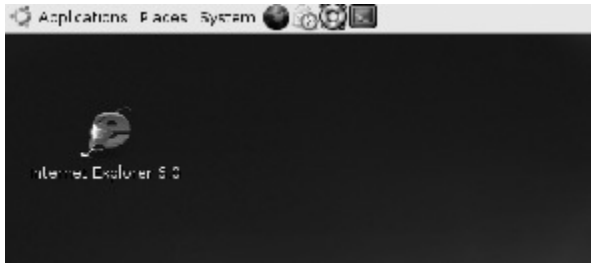
In addition to various Windows applications, you may find it interesting to run various Windows components as well—notably, Internet Explorer, the infamous standards-shy browser that is nevertheless used by over 80 percent of all web surfers. There are a number of ways you can install IE.

The easiest way, by far, is to use one of the utilities that have sprung up around Wine to accomplish this very thing. The three major such utilities are WineTools (http://www.von-thadden.de/Joachim/WineTools/), Sidenet (http://sidenet.ddo.jp/winetips/config.html) and the awkwardly named IEs 4 Linux (so named because it will let you install IE 6, IE 5.5, and IE 5 all at once, if you so choose). Of the three, I recommend IEs 4 Linux (http://www.tatanka.com.br/ies4linux/). It is the only one that doesn't radically modify your Wine filesystem and that seems to work smoothly with the latest version of Wine.

To install IEs 4 Linux, first download it from here: http://www.tatanka.com.br/ies4linux/en/instructions/. Then open the *tar.gz* file and run the *ies4linux* program that's inside. You'll be given a choice of which version(s) of IE to install; I chose IE 6 only. IEs 4 Linux will then proceed to download the version(s) you selected; IE 6 alone is an 80 MB download. The Microsoft browser(s) will be installed in a separate, *.ies4linux* directory in your home directory; your *.wine* directory is left untouched.

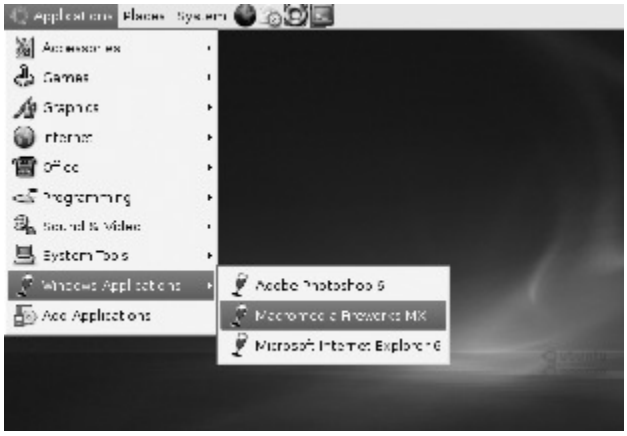
A nice, if rather bizarre touch: an Internet Explorer icon (customized with Wine glass and shown in Figure 9-7) is placed on your Ubuntu desktop. You can simply double-click it to launch the security-challenged browser, which defaults, for safety's sake, to an *about:blank* home page.

Figure 9-7. Internet Explorer on the Ubuntu desktop



Perhaps you don't want an IE icon on your desktop—I certainly didn't. So, I removed it and used Ubuntu's Applications Menu Editor to add Internet Explorer and some other Windows apps to the Ubuntu Applications menu. The result is shown in Figure 9-8 (you'll find the Wine glass in */usr/share/icons*).

Figure 9-8. Adding Windows programs to the Applications menu



Note that Wine's commercial cousins, CodeWeavers' CrossOver Office [Hack #31] (http://www.codeweavers.com/) and TransGaming's Cedega [Hack #88] (http://www.transgaming.com/), can add the menu items for you.

Thomas Pletcher

Hack 88. Play Windows Games

Those Windows games don't have to be trapped in that partition you rarely boot into. Instead, you can use Wine or Cedega to play them.

Wine (http://www.winehq.com), the open source Windows compatibility layer [Hack #87], is well known for its ability to run many popular Windows applications, and even certain Windows components such as Internet Explorer. Wine can also run some popular Windows games, though gaming is not its primary focus. Still, if a game runs successfully in Wine, then Wine is probably the best way to play it. It is usually a fairly straightforward matter to install and play games in Wine, not to mention the fact that Wine itself is free and open source software.

Still, there are other game-playing solutions for Linux which may often produce better results. That is certainly the case when it comes to classic, DOS-based games. For these, DOSBox (http://dosbox.sourceforge.net) is your best bet. DOSBox is a free and open source x86 and DOS emulator that runs on multiple platforms (including modern versions of Windows and Mac OS X, as well as Linux). It is especially good with older games.

Run Blasts from the Past

DOSBox is included in Ubuntu's universe repository [Hack #60], and it is generally up-to-date. So you can simply install it by running:

```
$ sudo apt-get install dosbox
```

or using Synaptic [Hack #55] or Adept [Hack #56]. In either case, the SDL multimedia library dependencies will also be installed. After you've installed DOSBox, you'll need to create a new directory in your home directory and name it something like *dosgames*. This is, of course, where you'll install those DOS classics you remember so fondly. Let's download and install a game to use with DOSBox; you'll find almost all the classic DOS games at the Abandonia site (http://www.abandonia.com).

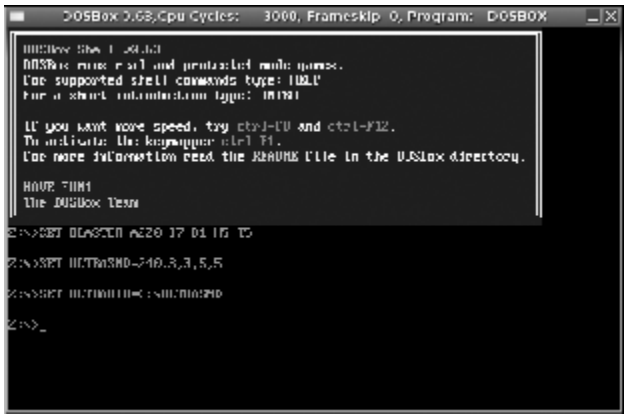
I chose a true all-time classic, the original Tetris (http://www.abandonia.com/games/en/69/Tetris.htm), and made a directory for it (*dosgames/Tetris*). Then I unzipped the Tetris download into the *dosgames/Tetris* directory. To launch DOSBox, you just type `dosbox` in Terminal.

Doing so brings up the DOSBox shell (shown in Figure 9-9), where you can enter various DOS commands and launch your games. The DOSBox shell opens in the Z directory by default, so you will need to mount the C directory, which contains your games. You can do so within the DOSBox shell by typing:

```
Z:> mount
          c
          /home

              /username/dosgames
```

where `username` is your username and `dosgames` is whatever you named the directory that contains your DOSBox games. After you've mounted the C directory, you can switch to it simply by typing `c:` within the shell.

Figure 9-9. The DOSBox shell



Now that you're in the C directory, type `dir` to have a look at its contents. (See the DOSBox Wiki at http://dosbox.sourceforge.net/wiki/index.php?page=DOSBoxWiki for a DOS command reference and other useful information.) If you've installed Tetris, you should see a line like this:

```
TETRIS    <DIR>    15-03-2006    14:14
```

which simply tells you that *TETRIS* is a directory, and gives you the time and date of its installation. Inside the *TETRIS* directory (change to it by typing `cd tetris`, and then look inside with `dir`), you'll see one file: *Tetris.com*. (DOS executable files have one of three extensions —*.exe*, *.com*, or *.bat*) You can launch the game by typing `tetris.com`. (You can also launch the game by typing `TETRIS.COM`—don't worry about case sensitivity.) Figure 9-10 shows the game in action.

Figure 9-10. Playing the original Tetris, with DOSBox

There are a few basic commands you can access if you need them:

- Ctrl-F7 decreases frameskip.
- Ctrl-F8 increases frameskip.
- Ctrl-F11 slows down the game.
- Ctrl-F12 speeds up the game.

When you've finished your game, type `z:` at the prompt and press Enter. This takes you back to the `Z:>` prompt, where you can type `exit` to quit the DOSBox shell.

You're not limited to antique classics like Tetris when you use DOSBox; many more graphically advanced games are available as well. Abandonia, in addition to its very wide selection of games, is also valuable for its compatibility ratings; you can see at a glance whether an older game is compatible with DOSBox (most are). And many collections of DOS games are available on CD-ROM, including classic role playing games such as Wizardry and Might and Magic, as well as classic text adventures from Infocom. Some of these collections are easy to find in the $9.99 bin at your local computer store. Others are a bit harder to find, so you might have to check eBay.

Run Current Windows Games with Cedega

Cedega (http://www.transgaming.com) is a commercial version of Wine—commercial to a fault, some would say. Cedega, formerly known as WineX, is basically a fork of Wine that has gone proprietary. This has generated some resistance among free software fans, and you won't find Cedega in the Ubuntu repositories. But if you're a gamer, you'll want Cedega anyway; it's generally regarded as the best way to run recent Windows titles on Linux.

Cedega costs $5 per month, and you have to pay the first three months up front. However, there is a fully functional, two-week trial version available. I recommend you start with that; if you like it, you can convert to the paid version from the trial.

You can download the Cedega Time Demo from http://www.transgaming.com/products_linux.php. You'll find the download link near the bottom of the page, and you'll find system requirements and installation information at http://downloads.transgaming.com/files/timedemo_howto.html. Pay particular attention to the video-card requirements: Cedega won't be worth your subscription cost if your system isn't up to handling current games. NVIDIA GeForce or ATI Radeon class cards are pretty much the minimum requirement; pixel shaders are only available on NVIDIA FX class cards and above. ATI Radeon 8500 or better cards will also work, but only with proprietary ATI drivers. Intel and other cards may be able to drive some games, but probably not most.

To install the Cedega Time Demo, run the installation script with this command:

```
$ sh
                /PATH_TO_INSTALLER/
            cedega_timedemo_installer
```

Note that the preceding command will install the Time Demo in your user directory. If you'd prefer a system-wide installation, then run the preceding command as *root*.

The installer will check for updates and then ask whether you want to install the Microsoft Core Fonts and the MozControl package. You must agree to the licenses for each before they are downloaded as part of the install.

The TransGaming people will email you a registration key, which you'll need to enter during the process of installing Cedega. Figure 9-11 shows the Cedega registration process.

Figure 9-11. The Cedego demo must be registered during installation

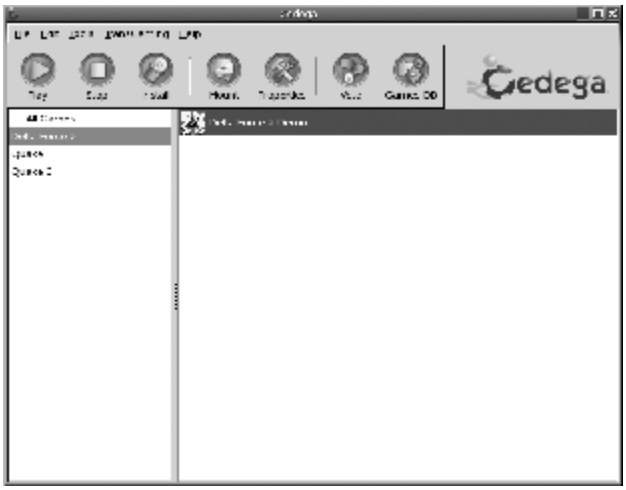
Finally, before you play your first game with Cedega, a wizard will walk you through tests of various facets of your system. If your computer can't pass the OpenGL Direct Rendering and 3D Acceleration tests, in particular, Cedega is not going to run many games.

After installation, you can launch the Cedega Time Demo by entering:

```
$ ~/cedega_timedemo
```

in a Terminal. This brings up Cedega's graphical interface, shown in Figure 9-12.

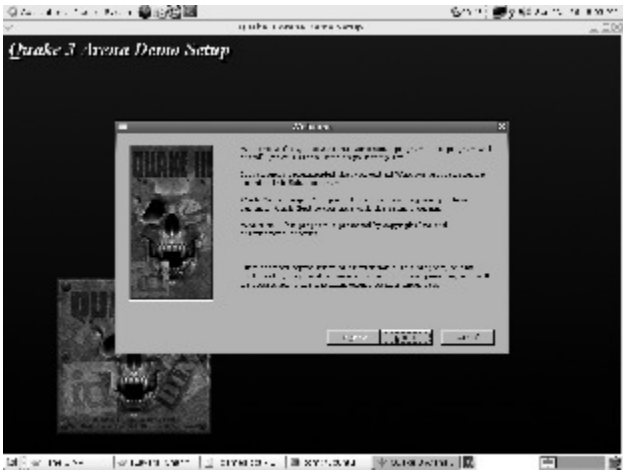
Figure 9-12. The Cedega interface



In Cedega, almost everything can be done via the GUI. There are command-line options too, though. You can read about these options and other Cedega usage tips on the Cedega How-To Guide page at http://downloads.transgaming.com/files/Cedega-How-To-5.0.0-2.html. Note that the Cedega Time Demo is actually the same as the full subscription version, currently 5.1.

To install a game with Cedega, just click the Install button and then browse to and run the installer. You can download games from a wide variety of sources (e.g., Download.com), and you can even mix and match the original Windows *.pak* files from CDs with open source game binaries for games like Quake I, II, or III. You can also install demos to try a game before installing the full version (see Figure 9-13), but note that there can often be significant differences between the demo and the full game; both may not run. TransGaming has an extensive games database (http://transgaming.org/gamesdb/) you can consult.

Figure 9-13. Installing the Quake 3 demo



As noted earlier, you'll want the best graphics setup possible to take maximum advantage of Cedega and today's Windows titles. But Cedega (and Wine as well) can run some games even on underpowered machines—a title such as Delta Force 2 being a case in point (see Figure 9-14).

Figure 9-14. Running Delta Force 2

Cedega, in spite of its licensing and cost issues, is regarded by many as the best way to run current Windows games on Linux. For hardcore gamers, that alone is enough to recommend it. Just be sure your system is suitably equipped, and also make sure you have the time and inclination to tweak your system to get the most out of Cedega, and today's games. Sometimes a game will run "out of the box," but sometimes it won't. For many gamers, getting such games to work is part of the fun.

Run Google Earth with Wine

Modern games aren't the only visually advanced Windows applications that can be challenging to run on Linux. Because it relies on DirectX and OpenGL, a sophisticated application like Google Earth poses similar issues, and there are similar rewards for those who can get it to run.

A how-to for Gentoo users for installing Google Earth with Wine was published at http://gentoo-wiki.com/HOWTO_Install_GoogleEarth_with_wine, and I was able to adapt these instructions for an Ubuntu PC. Your mileage may vary: chances are some additional tweaking will be required, depending on your computer's exact configuration. But you can use the following instructions as a starting point, and if you're lucky, they'll be all that is needed. First, of course, you must install Wine [Hack #87].

Next, you need to install the Microsoft DCOM98 (Distributed Component Object Model for Windows 98). First, run *winecfg* and set the Windows version to Windows 98. Then you can download DCOM98 from here: http:// download.microsoft.com/download/d/1/3/d13cd456-f0cf-4fb2-a17f-20afc79f8a51/ DCOM98.EXE (or just search for the link on Google; it's probably faster!). Now, install DCOM98 with a `WINEDLLOVERRIDE`, specifically:

```
$ WINEDLLOVERRIDES="ole32=n" wine DCOM98.EXE
```

After you have DCOM98 installed, download the *psapi.dll* from http://www.dll-files.com/dllindex/pop.php?psapi and place the DLL in your *~/.wine/drive_c/windows/system32* directory.

Downloading these DLLs from untrusted sources can be quite dangerous, since you have no way of knowing that they have not been compromised (for example, they could be harboring a Trojan horse). The safest option is to copy them over from a working Windows system that has recently been checked for spyware, viruses, and other malware.

Next, download Google Earth from http://earth.google.com. Before you do so, though, run *winecfg* again and set the Windows version to Windows XP. Also before you install Google Earth, make sure you download the *usp10.dll* (Google for the download location); place this in the *~/.wine/drive_c/windows/system32* directory as well.

Now, install Google Earth like so:

```
$ WINEDLLOVERRIDES="ole32,oleaut32,rpcrt4=n" wine GoogleEarth.exe
```

After you've managed to install Google Earth, you reach the moment of truth: it's time to run it. Here's the command I used:

```
$ WINEDLLOVERRIDES="ole32,usp10,msvcrt=n" \\
             wine "c:\\Program Files\\Google\\Google Earth\\GoogleEarth.exe"
```

This *should* work (see Figure 9-15). However, it might not work on *your* system, and therein lies the challenge. (If you do run into trouble getting it to work, be sure to check the HOWTO mentioned earlier in this section.) Getting Google Earth to run on Ubuntu or any Linux system is a worthy hack, and tools like Wine (along with DOSBox and Cedega, for vintage and state-of-the-art games) make it possible.

Figure 9-15. Google Earth launching via Wine

Thomas Pletcher

Hack 89. Run Ubuntu Inside Windows

If you want to avoid the confines of the emulator sandbox, give coLinux a whirl and run Ubuntu as a cooperative process alongside of Windows.

coLinux (Cooperative Linux), available from http://www.colinux.org/, takes a unique approach to solving the same problems that emulators do. Instead of providing a completely enclosed sandbox like Virtual PC or VMware, it's a port of the Linux kernel that lets you run Linux alongside Windows. This means that you don't get a fully emulated and/or virtualized PC environment. But you do get enough to launch a Linux distribution in console mode with networking support, and from there you can use *ssh* to tunnel into the coLinux system and run applications on your Windows X11 server. You can get a great, free X11 setup with Cygwin (http://www.cygwin.com), a Windows port of many Linux tools and applications. With the combination of coLinux and Cygwin, you can get Ubuntu running alongside Windows, and without the overhead and complexity of emulation and virtualization, it runs quite fast.

To get Ubuntu up and running with coLinux, you'll first need to install coLinux itself and choose the minimal Debian disk image that comes along with it. With some *apt-get* trickery, you'll be able to turn that Debian installation into an Ubuntu system.

Install coLinux

Download the latest installer from the coLinux web site, and run it under Windows. If you don't have WinPcap (http://winpcap.mirror.ethereal.com/install/default.htm) installed, you should download and install it (don't worry, the coLinux installer will remind you about this, too).

When you run the installer, it will try to install coLinux into *C:\\ProgramFiles\\coLinux*. You should change this to *C:\\coLinux*, because the default configuration files expect to find it there. Plus, paths without spaces make everyone happier.

When you're prompted to choose a Linux distribution, select the Debian installer. This will download a small compressed file that you'll later expand to about 1 GB. Sometime during the install, you'll get a Windows Logo compatibility-testing error message about the TAP-Win32 adapter, which is necessary if you want to use networking, so you'll need to click Continue Anyway and let Windows install it.

When installation is finished, you'll be offered the opportunity to look at the *README* file. I know we never do this, but with software that's this complicated, I think you should.

Set Up the Debian Image

Now that coLinux is installed, there are a few more steps to get it configured. By now, I hope you've installed Cygwin, since you'll need to use some of the utilities from it to get things set up nicely. If not, go install it now and, when you're ready, open a Windows command prompt, then `cd` to the *C:\\coLinux* directory. You'll need to expand the filesystem image you downloaded. It's probably compressed with *bzip*, so you should use the *bunzip2* utility from Cygwin to decompress it:

```
C:\\coLinux>\\cygwin\\bin\\bunzip2 Debian-3.0r2.ext3-mit-backports.1gb.bz2
```

Consider backing up this file before you unzip it, just in case you make a horrible error while you are hacking it and want to start out with a fresh image.

Next, you'll need to create a swapfile. Cygwin's *dd* is fine for this purpose. Windows has a utility called *fsutil* that can also create it, so either of these commands will create a 512 MB swapfile:

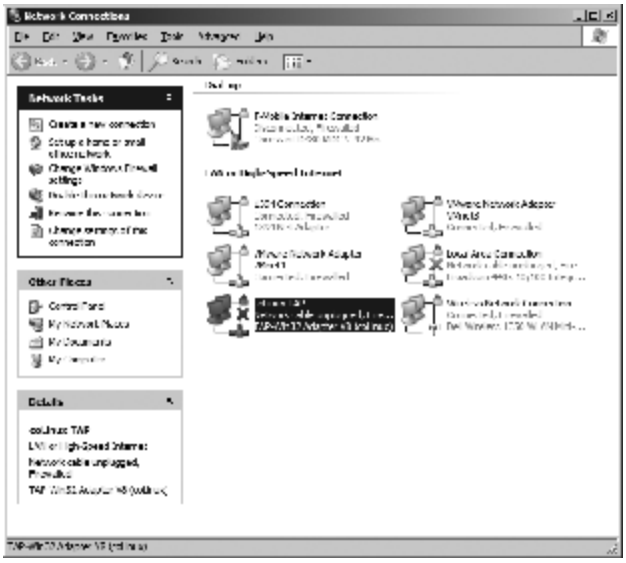
```
> fsutil file createnew swap_device 536870912
> c:\\cygwin\\bin\\dd if=/dev/zero of=swap_device bs=1M count=512
```

Now you need to work on the configuration file. Copy *default.colinux.xml* to *colinux.xml*, and open *colinux.xml* in your favorite text editor.

Edit this file and make sure the path to your filesystem image (default is *root_fs*) and swap device (default is *swap_device*) are correct. Bump your memory size up to whatever you feel comfortable setting aside (256 MB is good, assuming you have more than 512 MB of memory and don't plan to do something memory-intensive while your coLinux system is running).

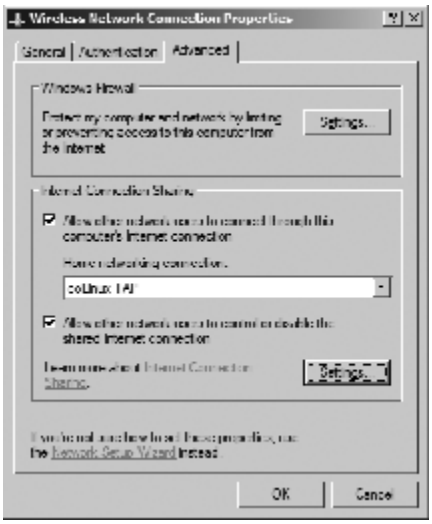
Now it's time to get the network connection set up. Go to Control Panel→Network Connections and locate the TAP-Win32 adapter, highlighted in Figure 9-16. Change its name if you'd like (the default will be something like "Local Area Connection 5"), and add its name to the network entry in *colinux.xml*.

Figure 9-16. Finding the TAP adapter in Windows



Now you need to share your Internet connection. Locate your Internet connection in the Network Connections window, right-click on it, and select Properties. Go to the Advanced tab, click "Allow other network users to connect through this computer's Internet connection," and select the TAP adapter in the "Home networking connection" drop-down menu, as shown in Figure 9-17.

Figure 9-17. Sharing your Internet connection



Next, return to the Command Prompt and start colinux as a standalone process. You can run it as a daemon, but it's easier to debug this way. The `-t NT` option uses the Windows command prompt as the console, for easier cutting and pasting, and the `-c` option specifies the configuration filename:

```
C:\\coLinux>colinux-daemon.exe -t NT -c colinux.xml
Cooperative Linux Daemon, 0.6.3
Compiled on Sun Feb  5 20:25:03 2006

Linux version 2.6.11-co-0.6.3 (george@CoDebianDevel) (gcc version
3.4.4 20050314
 (prerelease) (Debian 3.4.3-13)) #1 Sun 384MB LOWMEM available.
 initrd enabled: start: 0xd7e10000  size: 0x001ef78a)
 On node 0 totalpages: 98304
```

You'll see the usual messages scroll by until you get a login prompt. Log in as *root*, with the password `root`. At this point, networking won't be working, since there are some hardcoded (and incorrect) values in */etc/network/interfaces*. Run the command `pump -i eth0` and check the output of `ifconfig` to see if you got an IP address from Windows connection sharing. If so, you can edit (you may want to do an `apt-get update` and then `apt-get install vim` or `apt-get install emacs`) */etc/network/interfaces* and replace:

```
iface eth0 inet static
 address 192.168.254.205
 netmask 255.255.255.0
 gateway 192.168.254.1
```

with:

```
iface eth0 inet dhcp
```

If you want the same address each time, you should instead replace 192.168.254.205 with the address you got from *pump*. As long as you're not running multiple coLinux instances, you'll never need to worry about IP address collisions. If you are running multiple instances, you can always choose IP addresses in the 192.168.0.0/24 that don't conflict.

At this point, you can issue the command `halt` inside of the coLinux distribution to shut down, start up coLinux again, and verify that networking is configured correctly on boot. Now you're ready to install Ubuntu.

If your real network uses the same address range as that used by Windows (192.268.0.0/24), you'll have problems. If this happens, you'll need to change your physical network to use a different range of addresses.

Install Ubuntu

It's pretty easy to turn Debian into Ubuntu, but for the smoothest transition, you should make sure you've got the latest and greatest Debian system, so do an `apt-get update; apt-get upgrade`. You also need to update a few things to make sure your system has the necessary versions to install Ubuntu. This should do the trick:

```
$ sudo apt-get install sed locales
```

For another (not quite as quick-and-dirty) approach to converting Debian to Ubuntu, see "Convert Debian to Ubuntu" [Hack #66].

Next, edit */etc/apt/sources.list*, delete everything in there, and insert the following lines:

```
deb http://archive.ubuntu.com/ubuntu dapper main restricted
deb-src http://archive.ubuntu.com/ubuntu dapper main restricted

deb http://archive.ubuntu.com/ubuntu dapper-updates main restricted
deb-src http://archive.ubuntu.com/ubuntu dapper-updates main restricted

deb http://security.ubuntu.com/ubuntu dapper-security main restricted
deb-src http://security.ubuntu.com/ubuntu dapper-security main restricted
```

Now, it should theoretically be smooth sailing from here, but it probably won't be. To kick off the update, run `apt-get update` and then `apt-get dist-upgrade`.

If you see a message about configuring LILO, you can ignore it. coLinux doesn't use it.

During the upgrade, you will almost certainly get this error:

```
E: This installation run will require temporarily removing the
essential package e2fsprogs due to a Conflicts/Pre-Depends loop.
This is often bad, but if you re ally want to do it, activate the
APT::Force-LoopBreak option.
E: Internal Error, Could not early remove e2fsprogs
```

If so, add `-o "APT::Force-Loopbreak=true"` to the `apt-get dist-upgrade` command line, as in:

```
$ sudo apt-get -o "APT::Force-Loopbreak=true" dist-upgrade
```

If you continue to get errors, run this command to clear things up:

```
$ sudo apt-get -o "APT::Force-Loopbreak=true" -f install
```

Then, run the `apt-get dist-upgrade` command again, and it should run to completion.

When it's done, you must `apt-get install ubuntu-keyring` so that *apt-get* can verify package signatures. Now you can start *apt-get*ting what you need. Pick out some high-level applications, since this will pull in a lot of other dependencies. I'd suggest something like:

```
$ sudo apt-get install firefox openoffice.org2 gnome-games
```

Do what you can to prevent Ubuntu from installing an X server, since that can only complicate things.

Enabling SSH

If you'd rather not work in the coLinux console, you can *ssh* into your coLinux system. First, run `apt-get install ssh`, then use `ifconfig eth0` to determine your IP address, which will be on the private network that Windows assigned for Internet connection sharing, such as 192.168.0.143. Fire up PuTTY or openssh (available in Cygwin) to connect to that IP address. If you're using Cygwin, you may need to set your `TERM` to something other than `cygwin`, such as `ansi`:

```
bjepson@thlon ~
$ ssh root@192.168.0.16
Password:
Last login: Fri Mar  3 23:00:31 2006 from thlon.mshome.net
Linux colinux 2.6.10-co-0.6.2 #5 Sat Feb 5 10:19:16 IST 2005 i686 GNU/Linux

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
```

```
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
colinux:~#
```

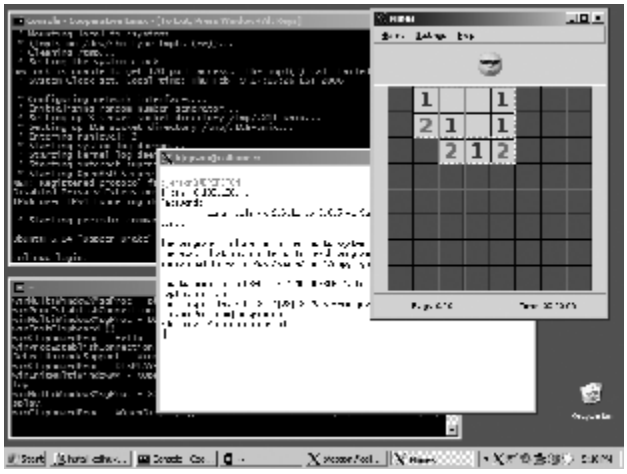
Running X11 Applications

To run any X11 applications, you'll need to `ssh -X` into your coLinux system from within the Cygwin X11 environment and launch applications from there. (You will probably need to enable X11Forwarding in Ubuntu's */etc/ssh/sshd_config* and restart *sshd* with `/etc/init.d/sshd restart` to get this working. You'll need to `apt-get xauth` as well.)

Now you're ready to start playing. Run the `adduser` *`username`* command to add a mortal user, and then you can log out as *root*.

Next, start up a Cygwin shell on the Windows system, and run the Cygwin X server with the command `startxwin.sh`. When the xterm appears, use `ssh -X` *`IP_ADDRESS`* to log in to your coLinux system. Replace *`IP_ADDRESS`* with either the fixed private network address you assigned it or the latest one it got from DHCP (use *ifconfig* in the Ubuntu system to find this out). Once you're logged in, you can start running your favorite programs (see Figure 9-18)!

Figure 9-18. Windows and Linux in perfect harmony



If you get the error"xterm Xt error: Can't open display: xterm: DISPLAY is not set," you should log out, and try `ssh -X` again, but with the `-v` option, which gives you verbose details on your connection attempt and should shed light on the situation. For example, if you see an error related to xauth, that means you need to `apt-get xauth` on the coLinux Ubuntu system.

Brian Jepson

Hack 90. Use Xen to Host Virtual Machines

Use Ubuntu as a deployment platform for multiple virtual machines.

A *virtual machine* is a simulated computer-inside-a-computer, allowing you to boot an entire extra operating system inside your primary environment. You may already be familiar with the concept of emulation: booting Microsoft Windows within your Linux machine using VMWare [Hack #92], running an arcade-game emulator such as MAME on your computer so you can play old-style console games on your PC, or using Virtual PC on a Macintosh to allow it to run Windows programs.

These systems generally work by using both software that pretends to be hardware (emulation) and software that encapsulates and grants access to physical devices (virtualization). So, in practice, what we commonly think of as an emulator is actually a combination of emulation and virtualization. For example, if you're running an x86 emulator on an x86 system, the CPU certainly doesn't need to be emulated (so it's virtualized), but other devices, such as network adapters, may be emulated. For performance reasons, a virtualization environment such as VMWare will emulate as little as possible and virtualize everything it can.

This scheme allows an application designed for the target hardware to run unmodified. The application itself probably won't be able to tell the difference: it just thinks it's running on whatever hardware is being emulated and virtualized by the host system.

While this approach can be applied to single applications, it can even be applied to entire operating systems: it's possible to boot an entire extra copy of Ubuntu, for example, right inside the copy you already have running. The extra copy is a complete, self-contained virtual computer with its own IP address, kernel, users, and applications. Mainframe systems have been doing this for decades, with many virtual machines running simultaneously under the management of a *hypervisor*. By taking this approach, the primary operating system running directly on the hardware can be relatively simple since it doesn't need to provide any end-user services or applications itself. All it needs to do is run the hypervisor to manage the child operating systems and provide a stable environment on which they can run.

The benefit of this approach is that the core system itself can be extremely stable, secure, and reliable, allowing mainframes to run for years or decades without any downtime at all. Once the parent OS has been installed and configured, it can be left running indefinitely and be almost immune to hackers and other external threats by being completely segregated from the network. Then any applications or services that need to be provided to end users can be installed within a virtual machine that can be backed up easily and "rebooted" at any time without affecting the underlying hardware or any other virtual machines that happen to be running alongside it. Separating services into different virtual machines is a great way to limit the impact of security problems, since a compromised web server running on one virtual machine does not give the attackers any access to mail services running in a different virtual machine, even if they're on the same physical server.

Other advantages of this setup include rapid backups, versioning and rollbacks; live migration of running services to another host; rapid provisioning of new services; segregated system administration; and support for multiple runtime environments simultaneously on one server.

This technology has now started to filter down to Linux in a number of forms, including UML (User Mode Linux), VMWare, and Xen. This hack focuses on Xen, a project to implement a mainframe-style hypervisor within a Linux environment, giving Linux much of the power and flexibility of the older Unix systems that inspired it. Xen's approach to virtualization is to provide a virtual architecture to which the Linux kernel can be ported. In just the same way that Linux is supported on architectures such as ia32 (386), ia64, PowerPC, AMD64, and many others, Xen defines an architecture for which the kernel can be compiled. This virtual architecture is provided by a special kernel that runs on the host machine, and virtual machines that execute on it run an otherwise unmodified kernel that has been compiled for the Xen architecture.

The host machine is generally referred to as *dom0*, and guest virtual machines that run on it are referred to as *domU* machines.

Initial Host Installation

The whole idea of virtualization is to run the *dom0* base system in a totally cut-down form with as little installed as possible. The less you have in the base system, the less there is to go wrong. With an extremely stable base system, you can then run all your services in virtual machines.

Start by doing a clean install of Ubuntu in server mode [Hack #93] so you have as little cruft on your system as possible.

Base Packages Required by Xen

To run Xen and provide networking services to virtual machines, you will need to install a number of packages:

```
$ sudo apt-get install grub bridge-utils iproute python \\
             python-twisted gcc libcurl3 libcurl3-dev zlib1g zlib1g-dev
```

Obtain Xen3 Binary Distribution

While you can apply the Xen patches and build a kernel [Hack #78], the simpler approach for now is to just grab the latest prebuilt binaries from the Xen project and install them using the provided installer. To reduce load on the Xen project infrastructure, the developers request that P2P systems such as BitTorrent be used to fetch the binaries where possible, but if you submit your email address, they will also provide you with a direct download link if you prefer. Visit http://www.xensource.com/xen/downloads for more information. For Ubuntu, the item you need to download is the Xen 3.0 Tarball. (Even if you're on a single processor system, it's OK to download the one marked 32-bit SMP.)

Extract Binary Distribution

Once you've downloaded the binary distribution, just unpack it and run the provided install script:

```
$ tar zxf xen-3.0.1-install-[arch].tgz
$ cd xen-3.0.1-install
$ sudo ./install.sh
```

The install script places a number of special kernels and *initrd* images in */boot*, so assuming all went well you should now have quite a few extra items in there.

Configure GRUB Bootloader

GRUB is designed to automatically detect new kernels and autoconfigure them, but unfortunately it doesn't manage the Xen kernels in quite the right way, so you will need to make some manual adjustments. Start by updating the autogenerated kernel list:

```
$ sudo update-grub
```

You should see GRUB report your new Xen kernels in addition to any that were previously installed. Now open the GRUB configuration file (*/boot/grub/menu.lst*) in your favorite editor and look down near the bottom for all the autogenerated kernel definitions. You should find one that looks something like this:

```
title           Ubuntu, kernel 2.6-xen0
root            (hd0,0)
kernel          /boot/vmlinuz-2.6-xen0 root=/dev/hda1 ro quiet splash
boot
```

To allow the Xen kernel to boot properly, the configuration stanza needs to be restructured. It's also a good idea to put the new stanza up above the list of automagically detected kernels, so look for a line that reads:

```
### BEGIN AUTOMAGIC KERNELS LIST
```

and just before it, put in an entry using the same values as the autogenerated stanza but restructured to use a `module` line like the one shown here, and with the `kernel` line modified as shown:

```
title           Ubuntu, kernel 2.6-xen0 static
root           (hd0,0)
kernel          /boot/xen-3.0.1.gz console=vga
            module          /boot/vmlinuz-2.6-xen0 root=/dev/hda1 ro console=tty0
```

The `static` entry in the title is just a label so that when GRUB displays a list of available kernels, you can tell which ones were added automagically and which one is your statically defined kernel.

Now tell GRUB to update its kernel list again, just to make sure you haven't made any mistakes:

```
$ sudo update-grub
```

Xen Services

Xen needs to start up a number of services at bootup, so set appropriate runlevel symlinks:

```
$ sudo update-rc.d xend defaults 20 21
$ sudo update-rc.d xendomains defaults 21 20
```

Enable Networking

Virtual machines you run on your host will most likely need some way to make network connections. Obviously, they don't have actual Ethernet cards installed in their virtual motherboards, so the usual solution is to have the host machine provide a virtual "bridge" from its own networking stack to the guest system. Each guest virtual machine can then be assigned its own IP address and even a virtual MAC address,

with packets passing through the virtual bridge, into the host networking stack, and finally out the host's Ethernet card and onto the real network.

To keep everything as secure as possible, it's also a good idea to implement firewalling within the *dom0* host to prevent external machines attacking the *domU* guests.

Install the packages required to provide network connectivity and network-management services to the guest operating systems:

```
$ sudo apt-get install iproute bridge-utils screen ssh
```

Disable TLS Libraries

The standard Thread Local Storage (TLS) library is incompatible with the Xen kernel, so it's necessary to either disable the library or replace it with a special Xen-friendly version before attempting to reboot with the Xen kernel. Otherwise, an emulation mode is used within Xen that reduces performance considerably.

The simplest solution for now is to move the library aside so it's not available when the host reboots:

```
$ sudo mv /lib/tls /lib/tls.disabled
```

If necessary, you can always move it back later if you need it for a non-Xen kernel.

For running heavily threaded code, you may prefer to install a *nosegneg* (no segmentation negotiation) version of the library.

Reboot

You should now be ready to reboot your *dom0* host with the Xen kernel, so `sudo reboot` and make sure GRUB loads the correct kernel when it restarts. You can also double-check the kernel version once the machine has finished booting:

```
$ uname -r
2.6.12.6-xen0
```

Your *dom0* machine is now ready to host guest *domU* virtual machines, so either download a pre-prepared image or create your own [Hack #91].

Hack 91. Create An Ubuntu/Xen Virtual Machine

Install Ubuntu into a virtual-machine image and boot it using Xen.

The previous hack showed you how to set up a Xen server [Hack #90]. To make use of your newly Xen-enabled host, you need to create some *domU* virtual machines to run on it. In this hack, you'll set up a basic Dapper virtual machine (VM).

Prepare VM Filesystems

The VM needs both root and swap filesystems. In a production environment, these would most likely be stored on some form of shared-access filesystem, such as a SAN or similar, but to keep things simple for now you should create them as loopback disk images on the local disk. So start by making a couple of directories to store them in:

```
$ sudo mkdir -p /vm/vm_base
$ sudo mkdir /vm/images
```

Root filesystem

The root filesystem will be a 2 GB image, so first use *dd* to create a disk image and then use *mkfs* to set up a filesystem in it. 2 GB should be enough space for initial testing, but if you want to vary the size, just change the `count` argument, which, when multiplied by the `bs` (block size) argument, determines the total disk size.

While *mkfs* is running, it will complain that *vm_base.img* is not a special block device and ask if you want to proceed anyway. Say `y`:

```
$ sudo dd if=/dev/zero of=/vm/images/vm_base.img bs=1024k count=2000
$ sudo mkfs.ext3 /vm/images/vm_base.img
```

Swap filesystem

You can create a 200 MB swap image in a similar way:

```
$ sudo dd if=/dev/zero of=/vm/images/vm_base-swap.img bs=1024k count=200
$ sudo mkswap /vm/images/vm_base-swap.img
```

Mount the root filesystem image

Mount the root filesystem image as a loopback device, allowing it to appear as a separate volume, even though it's only an image stored on the local disk:

```
$ sudo mount -o loop /vm/images/vm_base.img /vm/vm_base
```

Install Ubuntu into the Root Filesystem

Now you have the equivalent of a blank disk just waiting to have Linux installed on it. The system you install will become the *domU* virtual machine that is booted by Xen. Installing into a different volume can be a tricky exercise, but luckily Debian provides an extremely useful tool called *debootstrap* that takes care of all the hard work for you. First grab *debootstrap*:

```
$ sudo apt-get install debootstrap
```

Then use it to build a new Breezy base system into the root filesystem. The installation process has been radically overhauled for Dapper, so for now it's easier to install Breezy first and *dist-upgrade* to Dapper afterwards:

```
$ sudo debootstrap --arch i386 breezy /vm/vm_base/ \\
             http://archive.ubuntu.com/ubuntu
```

If you're wondering why you're installing Breezy, it's because *base-config* no longer exists in Dapper (the install process has been largely rewritten into a single-stage system, and I haven't yet figured out exactly how the initial configuration takes place). A simple workaround is to do a Breezy bootstrap and then update it to Dapper.

The preceding example specifies the architecture as `i386` (32-bit Intel). You may need to specify an alternative, such as `powerpc` or `amd64`, depending on your system architecture.

Running *debootstrap* can take quite a long time because it needs to download all the packages, and if you need to go through this process several times, you may want to set up a local package cache [Hack #61] or replace the mirror address with a reference to *file:///media/cdrom* and use an Ubuntu install CD. If you have a local *apt-cacher* running already, you can include the cache address like so:

```
$ sudo debootstrap --arch i386 breezy /vm/vm_base/  \\
             http://localhost/apt-cacher/archive.ubuntu.com/ubuntu
```

Configure the Virtual Machine

At this point, you have a basic Ubuntu install, but it's totally unconfigured. If your host machine is running Ubuntu, you can save yourself some effort by copying in your */etc/apt/sources.list* file:

```
$ sudo cp -a /etc/apt/sources.list /vm/vm_base/etc/apt/
```

Then edit */vm/vm_base/etc/apt/sources.list* and change any references from `dapper` to `breezy`, since you are starting with a base Breezy install.

Now edit */vm/vm_base/etc/network/interfaces* and set up the loopback address and a virtual eth0 interface. You will need to adjust these values to suit your own network. The address allocated to eth0 needs to be one that's not in use yet:

```
auto lo
iface lo inet loopback
      address 127.0.0.1
      netmask 255.0.0.0

auto eth0
iface eth0 inet static
      address 192.168.0.101
      netmask 255.255.255.0
      gateway 192.168.0.1
```

Edit */vm/vm_base/etc/hostname* and set it to `vm01`.

Edit */vm/vm_base/etc/hosts*, and put in some basic entries:

```
127.0.0.1 localhost.localdomain   localhost  vm01

# The following lines are desirable for IPv6 capable hosts
::1    ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

The VM also needs to be given a filesystem configuration. Note that this is not necessarily the same as the configuration of the host, and the volume names have nothing to do with your actual local disk. The volumes that the VM will see when it starts up are virtual volumes that will be determined by the Xen configuration, so for now just edit */vm/vm_base/etc/fstab* and set it up like this:

```
/dev/hda1       /           ext3    defaults  1   2
/dev/hda5       none        swap    sw        0   0
/dev/pts        devpts      gid=5,mode=620    0   0
none            /dev/shm    tmpfs   defaults  0   0
```

Run Setup Within the VM Image

Use *chroot* to switch into the root of the VM filesystem:

```
$ sudo chroot /vm/vm_base
```

From now on, everything you do will be within the context of the VM, so you can manually invoke the tools that are normally run in the last stages of the regular Ubuntu installer. Make sure the universe repository [Hack #60] is enabled in *sources.list* (Breezy has a *universe* repository as well), and tell *apt* to update the package list and then set up locales:

```
# apt-get update
# apt-get install localeconf
# dpkg-reconfigure localeconf
```

You will then be asked a series of questions when the `localeconf` package is installed and then additional questions when you reconfigure it:

Manage locale configuration files with debconf?

Answer "Yes."

Replace existing locale configuration files?

Answer "Yes."

Default system locale

Select the applicable default, such as *en_US ISO-8859-1*.

Environment settings that should override the default locale

Don't select any options; just select "OK" to proceed.

At this point, many of the standard devices don't exist and locale generation won't be working, so mount the *proc* virtual filesystem and create the default devices:

```
# mount -t proc proc /proc
# cd /dev
# ./MAKEDEV generic
```

When *MAKEDEV* finishes run *base-config* to bootstrap the system:

```
# base-config
```

base-config will ask you all the usual questions that you normally see when running the Ubuntu Breezy installer, so answer all the questions as normal, with one exception: when asked to specify an access method for *apt*, select "edit sources list by hand," which will then put you into the *vim* editor with the *sources.list* file open. Since it's already set up correctly, just type `ZZ` to save and exit *vim*, and let *apt* run through the process of verifying the repositories and installing all the base packages.

Once the Breezy installer has finished, edit the */etc/apt/sources.list* again, and this time change all references from `breezy` to `dapper`. Then update the package list again and *dist-upgrade* to Dapper:

```
# apt-get update
# apt-get dist-upgrade
```

You now have a basic Dapper install inside the loopback filesystem, so finally you can exit the chroot:

```
# exit
```

Get Ready to Crank Up the VM

The *domU* virtual machine will be booted using the Xen kernel installed on the *dom0* host, so copy the kernel modules into its filesystem, disable TLS, and unmount the image:

```
$ sudo cp -dpR /lib/modules/2.6.12.6-xenU /vm/vm_base/lib/modules/
$ sudo mv /vm/vm_base/lib/tls /vm/vm_base/lib/tls.disabled
$ sudo umount /vm/vm_base
```

It's quite likely that *umount* will complain that the device is busy even though nothing is using it anymore, so ignore the warning.

You now have one complete, functional *domU* virtual machine, but it's quite likely that you will ultimately want to create a few of them, so rather than boot the image you just created, it's best to treat it as a template and make copies of it to play with.

Copy the disk image to create a new VM:

```
$ sudo cp -a /vm/images/vm_base.img /vm/images/vm01.img
$ sudo cp -a /vm/images/vm_base-swap.img /vm/images/vm01-swap.img
```

Configure the VM in Xen

You need to create a configuration file for each *domU* virtual machine so that Xen will know how to boot it. Start by copying the sample configuration file into place:

```
$ sudo cp /etc/xen/xmexample1 /etc/xen/vm01.conf
```

The sample config file is very well commented, and most of the defaults are fine as a starting point, so open */etc/xen/vm01.conf* in your favorite editor and start working through it. Highlights to pay attention to include:

Kernel

The `kernel` parameter specifies the path to the *domU* kernel in the host system. Note that the *dom0* and *domU* kernels are almost identical, so you can use the *dom0* kernel image to boot the *domU* guest if you prefer. The *domU* kernel is basically just the *dom0* kernel, but with most of the modules stripped out because networking and many other services are supplied by the *dom0* host and the virtual machine doesn't need to manage such things itself. This makes the *domU* kernel about 30 percent smaller than the *dom0* kernel.

Memory

The *dom0* host preallocates RAM to each virtual machine using the `memory` parameter in the config file, so set it to some sensible value to start with, such as 128. You need to make sure you leave enough memory for the *dom0* host to after all the *domU* guests have claimed their allocated address space, so if, for example, you have 512 MB RAM in the host, you could run three guests with 128 MB allocated each and still have 128 MB available for the host.

Name

Each *domU* host needs a unique name that will be used to identify it. Set it to `vm01` for now.

Virtual network interfaces

You can specify virtual network interface settings directly in the config file or leave a blank value to have the guest specify its own settings. For now, just leave this blank because you configured a virtual eth0 interface earlier: `vif = [ '' ]`

Disk

The *dom0* host exposes real filesystems to the *domU* guest as if they were real filesystems, so the disk parameter allows you to specify what will be seen by the guest. For now, you should use the loopback disk images that were created earlier, so if you set up the guest's */etc/fstab* in the way I suggested earlier, you should set the disk parameter to:

```
disk = [ 'file:/vm/images/vm01.img,hda1,w','file:/vm/images/vm01-swap.img,hda5,w' ]
```

The disk images don't have to be mounted: the `file` argument tells Xen that it has to treat the target as a loopback disk image, and takes care of everything for you.

Boot Your Virtual Machine

Virtual machines are managed using *xm*, the Xen Manager, which allows you to boot, shut down, list, and migrate them to other physical hosts. Start up your newly created VM:

```
$ sudo xm create -c /etc/xen/vm01.conf
```

If all goes well, you will see Linux run through its boot sequence all the way up to a login prompt into your virtual machine. Log in as the user you created during the setup process and poke around. Try a few things like:

```
$ cat /proc/cpuinfo
$ cat /proc/meminfo
$ uname -r
```

It should look exactly as if you're in a totally separate machine, and the network interface will be functional, with packets bridged to the physical Ethernet through the kernel in the *dom0* host so you can even `apt-get install` some packages or do whatever else you need to do to set up the virtual machine just the way you want it.

While your virtual machine is still running, open another terminal session onto the *dom0* host and try running:

```
$ sudo xm list
```

to see a list of currently running guests. You can also view a dynamically updated list of hosts along with resource utilization:

```
$ sudo xm top
```

If you want to nicely bring a VM down without logging in to it, you can do so through *xm*:

```
$ sudo xm shutdown vm01
```

When you've finished playing around in your new VM, you can either shut it down from the *dom0* host using *xm* or shut it down from inside just like any other machine using `sudo shutdown`.

Create Additional Guests

To create additional guests, you need to repeat a number of the previous steps. Copy the templates again:

```
$ sudo cp -a /vm/images/vm_base.img /vm/images/vm02.img
$ sudo cp -a /vm/images/vm_base-swap.img /vm/images/vm02-swap.img
```

Mount the base image:

```
$ sudo mount -o loop /vm/images/vm02.img /vm/vm_base
```

Edit */vm/vm_base/etc/network/interfaces*, */vm/vm_base/etc/network/hostname*, and */vm/vm_base/etc/network/hosts* to set appropriate values for the hostname and IP address.

Unmount the base image:

```
$ sudo umount /vm/vm_base
```

Copy the Xen configuration:

```
$ sudo cp /etc/xen/vm01.conf /etc/xen/vm02.conf
```

Then edit it to change the paths to the disk images and the machine name, and you're ready to start up a second VM.

Hack 92. Split Your Machine's Personality

Give your machine split personalities by installing VMWare Server, and get some virtual machines running.

Virtualization is one of the current buzzwords in the computer industry. *Virtualization* is where a thin software layer is installed on a computer that allows its processor and other resources to be split among several *virtual machines*. This allows the computer operator to run several different operating systems underneath this virtualization layer. Each virtual machine has its own set of resources and operates independently from any other virtual machines that may be running at the same time.

VMWare (http://www.vmware.com) is one of the oldest companies in the virtualization business. It has had a Linux-based product lineup available for some time, both in the personal computer space (their Workstation product) and the server space (GSX and ESX server). Recently, however, they VMWare has released two new free (as in beer) products: VMWare Player (http://www.vmware.com/products/player/) and VMWare Server (http://www.vmware.com/products/server/). The VMWare Player product allows you to "play," or use, a premade virtual machine. The VMWare Server product is much more interesting, as it allows the creation of new virtual machines, and its modular, network-enabled console lets you control the virtual machines regardless if they're installed on your computer or another computer on the network.

VMWare Server is like a Swiss army knife—its uses are limited only by your imagination. Need a particularly sticky Windows application working, but still want to run Linux? If it doesn't work in Wine [Hack #87], you can install VMWare Server and run that application in a virtual machine. Want to test the latest Linux distribution, but have only one computer? Install the new distro in a virtual machine and don't worry about your main system.

Installing VMWare Server

VMWare Server is really designed for use on a Red Hat–based Linux distribution, but it does work fine on Ubuntu. However, it requires that some packages be in place before installation, since it has to compile the VMWare-specific kernel modules. VMWare also has a dependency on *inetd*, if you want to remotely connect to your virtual machines from another computer on the network.

To install VMWare, start by installing the *build-essential* and *netkit-inetd* packages. If you use the *aptitude* package manager, the additional recommended packages will get installed without any other intervention (be sure to replace `linux-headers-386` with the appropriate package for your architecture, such as `linux-headers-686` or `linux-headers-k7`):

```
bill@lexington:~$ sudo aptitude install build-essential \\
              linux-headers-386 netkit-inetd
```

Next, get VMWare Server from the download page (http://www.vmware.com/download/server/). You'll need to register on the site, and VMWare will send you a serial number to activate your VMWare Server installation. Ensure that you get the *.tar.gz* package, and not the *rpm* variant. Optionally, you can download the web interface and Windows console program separately. Once you've got the package downloaded, untar the package by running this command:

```
bill@lexington:~$ tar zxvf VMware-server-e.x.p-22088.tar.gz
```

Then start VMware installer and answer the questions it asks. In most cases, you can accept the defaults. At the end of the installer, you will be prompted for your serial number. This was emailed to you by VMWare after you registered, so check your mail for it:

```
bill@lexington:~/vmware-server-distrib$ sudo ./vmware-install.pl
Password:
Creating a new installer database using the tar3 format.

Installing the content of the package.

In which directory do you want to install the binary files?
[/usr/bin]

What is the directory that contains the init directories (rc0.d/ to rc6.d/)?
[/etc]

What is the directory that contains the init scripts?
[/etc/init.d]

In which directory do you want to install the daemon files?
[/usr/sbin]

In which directory do you want to install the library files?
[/usr/lib/vmware]

The path "/usr/lib/vmware" does not exist currently. This program is going to
create it, including needed parent directories. Is this what you want? [yes]

In which directory do you want to install the manual files?
[/usr/share/man]

In which directory do you want to install the documentation files?
[/usr/share/doc/vmware]

The path "/usr/share/doc/vmware" does not exist currently. This program is goingto create it, including needed
```

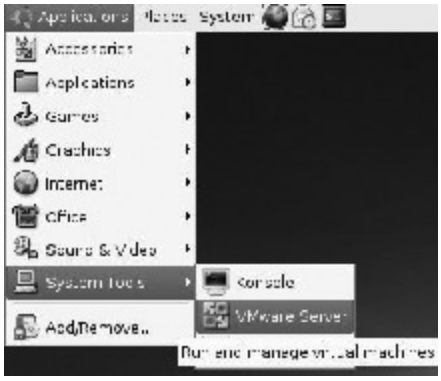
```
[yes]

The installation of VMware Server e.x.p build-22088 for Linux completed
successfully. You can decide to remove this software from your system at any
time by invoking the following command: "/usr/bin/vmware-uninstall.pl".

Before running VMware Server for the first time, you need to configure it by
invoking the following command: "/usr/bin/vmware-config.pl". Do you want this
program to invoke the command for you now? [yes]

... lots of output truncated ...

The configuration of VMware Server e.x.p build-22088 for Linux for this running
kernel completed successfully.
```

See the VMWare documentation for a detailed explanation of each installation and configuration option.

Running VMWare Server

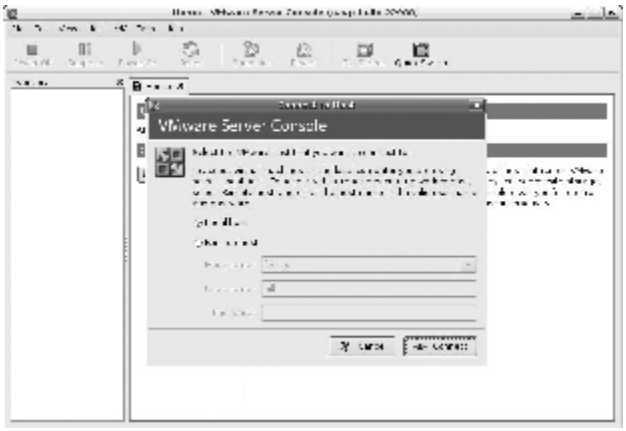
Once this process is over, the VMWare installer will have added a VMWare Server entry to your Applications→System Tools menu (see Figure 9-19). Click on this entry to start the VMWare console.

Figure 9-19. Starting the VMWare console



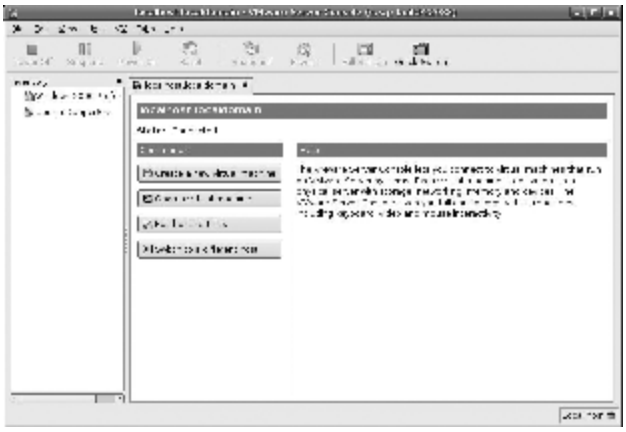
The VMWare console is where you will create, manage, and interact with your virtual machines. Once the console starts, it will ask you if you want to connect to your locally installed VMWare Server instance or connect over the network to another VMWare server. Select “Local host,” as shown in Figure 9-20, and then click Connect.

Figure 9-20. Connecting to a VMWare Server



Once you've successfully connected to the VMWare Server, you'll be presented with a screen in the console with various options (see Figure 9-21). You're going to create a new virtual machine, so click "Create a new virtual machine."

Figure 9-21. The main console window



The New Virtual Machine wizard will start up. Select Next to move past the splash screen. The wizard will ask you if you'd like to use a typical or custom virtual machine configuration—select Typical and click on Next. Now you get to the meat of the wizard; what type of operating system you'd like to install (see Figure 9-22). Select the guest OS family that you want to install in your new virtual machine, and choose the version. Click Next to move on.

Figure 9-22. Choosing your VM's operating system

At this point, you'll be prompted to name the new virtual machine and select a location to store its files. The defaults are usually fine here, unless you'd like to name the system something more descriptive. When you've named your machine, click on Next, and you'll be asked to select the type of networking you want for your virtual machine (see Figure 9-23). There are several different types of network options: *bridged networking*, where your virtual machine will appear as another host on your LAN; *network address translation*, where your VM will be "firewalled" behind your host machine; and *host-only*, where your VM can communicate only with the host computer. In this example, you'll select "Bridged networking." Click on Next to continue.

Figure 9-23. Selecting VM networking types

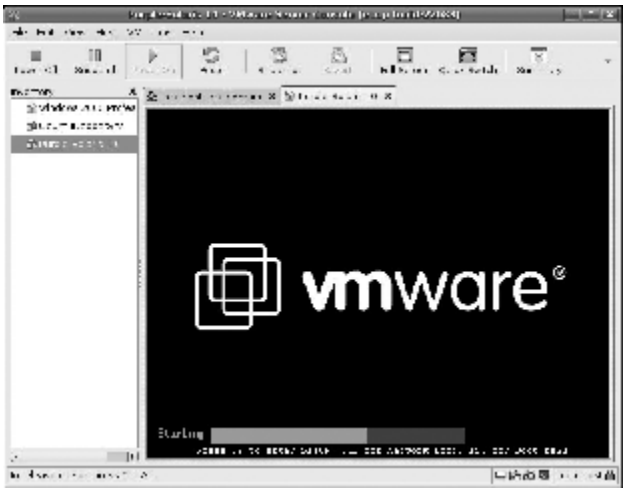
The final screen, shown in Figure 9-24, lets you select how large you want the virtual machine's hard disk to be. I usually uncheck the "Allocate all disk space now" and "Split disk into 2GB files" boxes. If the "Allocate all disk space now" checkbox is selected, VMWare Server will consume the amount of space specified in "Disk size" upon virtual-machine creation. If that box is unchecked, the virtual machine's disk will grow dynamically so as you install the OS and add applications, the disk will gradually get larger. The only real reason to check that box is for performance reasons, as it is faster than dynamic allocation; otherwise, leave it unchecked. Click on Finish to complete the New Virtual Machine Wizard.

Figure 9-24. Specifying VM disk size

At this point, your virtual machine is ready to be powered up for the first time and begin its OS installation. You can perform any tweaking to the virtual machine in the console here, prior to OS installation. All the items in the right pane are clickable and adjustable, such as the amount of RAM you want to dedicate to the VM and what installation media or mount point you wish to use. One cool thing about the VMWare console is that you can install your virtual machine from a CD in your server, a CD in your client, or straight from an *.iso* image (select VM→Removable Devices→CD-ROM 1→Edit). All those installation options really make life easy and give you options when you're installing a virtual machine.

All that remains at this point is to click the Power On button in the toolbar and watch your virtual machine boot for the first time. It will go through a virtual BIOS and POST check (see Figure 9-25), and then it'll begin installing from whatever boot media you selected. You'll need to click in the console window to "capture" your mouse and keyboard for use in VMWare (to get back to your main OS, press Ctrl and Alt on your keyboard at the same time).

Figure 9-25. First boot



Once you start using VMWare, you'll find new uses and ways to leverage virtual machines that you never thought possible. I've even used a virtual machine to kick-start other physical Linux boxes!

Chapter 10. Small Office/Home Office Server

ED: Need Introduction for this chapter

Hack 93. Install and Configure a Ubuntu Server

The Ubuntu installer makes it easy to do a clean and minimal server setup.

The Debian distribution has a well-deserved reputation as being extremely well suited for use in the datacenter, and Ubuntu builds on that by providing simplified installation and official commercial support, making it ideal for mission-critical server deployments.

Minimal Installation

A good principle when building servers is to install as few packages as possible, minimizing the number of things that can go wrong as well as the potential for security flaws. The Ubuntu installer offers a special "server" mode that makes it simple to create a basic server platform onto which you can install the software you require.

Before you perform the actual installation, boot up the server and enter the BIOS setup screen. Because servers typically run without a monitor attached, you will need to find the BIOS setting that tells the computer which errors it should consider fatal and make sure it won't fail on a "no keyboard" or "no monitor" error. The actual setting varies depending on the specific BIOS, so consult the manual for your computer or motherboard if necessary.

Save the BIOS changes and then boot the computer from the Dapper install CD, but don't proceed with the usual installation procedure. If you get a graphical menu, select Install a Server; otherwise type `server` at the first prompt. Then, go through the installation procedure [Hack #5]. This will give you a minimal selection of packages installed on the system. The server-mode installation doesn't include X or any services at all, giving you a clean platform to configure as you see fit.

One of the first services you will want to install is probably SSH, allowing you to use a secure shell to "Administer Your Server Remotely" [Hack #95].

Static Network Configuration

You may have a DHCP server on your network already, in which case your server will have been assigned an IP address, but most servers need to have static addresses assigned so they can be found on the network.

Open */etc/network/interfaces* (as *root*) in your favorite editor and find a section that looks like this:

```
# The primary network interface
auto eth0
iface eth0 inet dhcp
```

The `dhcp` argument tells Ubuntu to use a DHCP server to assign the configuration to this interface, so change it to a static configuration and specify the address, netmask and gateway (router) addresses. Here's an example:

```
# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.0.3
        netmask 255.255.255.0
        gateway 192.168.0.1
```

You can now manually force networking to restart with the following command, but be warned that if the static address you have assigned the server is different from the current address, any SSH sessions will be dropped. You will then be able to log back in at the new address:

```
$ sudo /etc/init.d/networking restart
```

UPS-Triggered Shutdown

A *uninterruptible power suppy* (UPS) will keep your server running during a short power failure, but batteries don't last forever, and you risk corrupted filesystems if the batteries go flat and the server stops abruptly. Connect your server to your UPS with a null-modem serial cable and install a program to monitor UPS status and begin a clean shutdown in the event of a blackout. Different brands of UPS have different communication methods, and there are a variety of UPS-management packages—including *genpower*, *apcd*, *apcupsd*, *powstatd*, and *nut*—each of which supports different types of UPS. If you run multiple servers on a single UPS, then *nut* (Network UPS Tools) is a good choice because it has the ability to initiate a shutdown of all your servers at once via the network:

```
$ sudo apt-get install nut
```

The exact setup process will depend on your UPS type, so start by looking through */usr/share/doc/nut/README.Debian.gz* for general background information, and then look at the example configurations in */usr/share/doc/nut/examples/*.

Network UPS Tools also has a number of supporting packages available:

nut-cgi
: Web interface sub system

nut-dev
: Development files

nut-snmp
: Meta SNMP Driver subsystem

nut-usb
: USB Drivers subsystem

Remember that if your server is shut down by the UPS-management software, it won't restart automatically when power returns. Now that you've got your server up and running, you may want to see "Build a Web Server" [Hack #96], "Build an Email Server" [Hack #97], "Build a Domain Name Server" [Hack #100], "Build A DHCP Server" [Hack #99], and "Build a File Server" [Hack #94].

Hack 94. Build a File Server

Share files with Linux, Windows and Macintosh machines.

There are many different file-sharing protocols, each with strengths and weaknesses and each coming from different development backgrounds. The traditional file-sharing protocol for Unix is NFS (Network File System); for Mac OS, it's AppleShare; and for Windows, it's SMB (Server Message Block). Running a mixed-environment file server used to require supporting a multitude of protocols simultaneously, but in recent years, there has been a convergence on the use of CIFS (Common Internet File System) across all platforms. CIFS is derived from SMB and is the standard file-sharing method in recent versions of Windows. It is also extremely well supported under both Linux and Mac OS as a client and as a server, thanks to the Samba project.

The server component of Samba can even run as a domain controller for a Windows network and supports several authentication backends, including LDAP and TDB. Large installations may benefit from using LDAP, but it is far more complex to set up, so this hack will cover the use of TDB, which is quite suitable for networks up to several hundred users.

Enable Quota Support

To work with quotas, first install the quota package:

```
$ sudo apt-get install quota
```

Open */etc/fstab* (the File System TABle) in your favorite editor and find the line that refers to the partition that will hold your shares. Add the `usrquota` and `grpquota` options. If you have */home* on a separate partition you will need to add the same options to that as well. The end result should look something like:

```
/dev/hda2     /        ext3 defaults,usrquota,grpquota    0   1
/dev/hda3     /home    ext3 defaults,usrquota,grpquota    0   2
```

Then set up the user and group quota files and remount the filesystem:

```
$ sudo touch /quota.user /quota.group
$ sudo chmod 600 /quota.*
$ sudo mount -o remount /
```

If you have a separate */home* partition, do the same for that file:

```
$ sudo touch /home/quota.user /home/quota.group
$ sudo chmod 600 /home/quota.*
$ sudo mount -o remount /home
```

Since there is already data on the partitions, you will need to run the Quota Check tool to scan the filesystems and record current usage per user and group, then activate quota enforcement:

```
$ sudo quotacheck -avugm
$ sudo quotaon -avug
```

The mechanism is now in place to enforce quotas, but no users or groups have limits set, so there is no limit yet on how much of the disk they can use.

Install Samba

On your server, install Samba itself plus some additional packages for documentation, share browsing, and printer sharing:

```
$ sudo apt-get install samba samba-doc libcupsys2-gnutls10 \\
              libkrb53 winbind smbclient
```

There are quite a few things to change in the default Samba config file, so open */etc/samba/smb.conf* in an editor and go through it to adjust the settings to match the following example. Most of the example can be copied verbatim, but set *WORKGROUP* to the name of the Windows domain (you can even leave it at WORKGROUP) and set *FILESERVER* to the hostname of your Linux server:

```
[global]
   workgroup = WORKGROUP
   netbios name = FILESERVER
   server string = %h server (Samba, Ubuntu)

   passdb backend = tdbsam
   security = user
   username map = /etc/samba/smbusers
   name resolve order = wins bcast hosts
   domain logons = yes
   preferred master = yes
   wins support = yes

   ## Use CUPS for printing
   printcap name = CUPS
   printing = CUPS

   ## Set default logon
   logon drive = H:
   #logon script = scripts/logon.bat
   logon path = \\\\fileserver\\profile\\%U

   ## User management scripts
   add user script = /usr/sbin/useradd -m %u
   delete user script = /usr/sbin/userdel -r %u
   add group script = /usr/sbin/groupadd %g
   delete group script = /usr/sbin/groupdel %g
   add user to group script = /usr/sbin/usermod -G %g %u
   add machine script = /usr/sbin/useradd -s /bin/false/ -d /var/lib/nobody %u
   idmap uid = 15000-20000
   idmap gid = 15000-20000
```

```
   ## Settings to sync Samba passwords with system passwords
   passwd program = /usr/bin/passwd %u
   passwd chat = *Enter\\snew\\sUNIX\\spassword:* %n\\n *Retype\\snew\\sUNIX\\ □
spassword:* %n\\n .
   passwd chat debug = yes
   unix password sync = yes

   ## Set the log verbosity level
   log level = 3

[homes]
   comment = Home
   valid users = %S
   read only = no
   browsable = no

[printers]
   comment = All Printers
   path = /var/spool/samba
   printable = yes
   guest ok = yes
   browsable = no

[netlogon]
   comment = Network Logon Service
   path = /home/samba/netlogon
   admin users = Administrator
  valid users = %U
   read only = no

[profile]
   comment = User profiles
   path = /home/samba/profiles
   valid users = %U
   create mode = 0600
   directory mode = 0700
   writable = yes
   browsable = no
```

The commented-out line that says:

```
   #logon script = scripts/logon.bat
```

defines a Windows batch script that will be executed by Windows workstations as soon as a user logs in. This can be really handy if you want to apply standard settings to all computers on your network; you may want to define server drive mappings, set up printers, or configure a proxy server. If you have a *logon.bat* script, uncomment that line.

Create directories to store domain logons and profiles:

```
$ sudo mkdir -p /home/samba/netlogon
$ sudo mkdir /home/samba/profiles
$ sudo mkdir /var/spool/samba
$ sudo chmod 777 /var/spool/samba/
$ sudo chown -R root:users /home/samba/
$ sudo chmod -R 771 /home/samba/
```

Make sure Samba has seen your new configuration:

```
$ sudo /etc/init.d/samba restart
```

To enable WINS (Windows Internet Name Service) host resolution edit the */etc/nsswitch.conf* and look for a line similar to:

```
hosts: files dns mdns
```

Change it to:

```
hosts: files wins dns mdns
```

Since your file server is going to be the domain controller (DC) for your Windows domain, you need to specify the computers that will be part of the domain. Open */etc/hosts* and add all the servers and workstations:

```
192.168.0.10 server1
192.168.0.101 workstation1
192.168.0.102 workstation2
...
192.168.0.107 workstation7
```

Windows typically has a special user named *Administrator* which is akin to the *root* user on Linux, so add the *root* user to the Samba password database and set up an alias for it. This will allow you to use the *Administrator* username to add new computers to the Windows domain:

```
$ sudo smbpasswd -a root
$ sudo sh -c "echo 'root = Administrator' > /etc/samba/smbusers"
```

To make sure everything has worked up to this point use *smbclient* to query Samba:

```
$ smbclient -L localhost -U%
```

The output will include details of several standard services, such as *netlogon* and *ADMIN*, as well as the machines that have registered in the domain:

```
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.21b]

        Sharename       Type      Comment
        ---------   ----      -------
        netlogon        Disk      Network Logon Service
        IPC$            IPC       IPC Service (fileserver server (Samba, Ubuntu))
        ADMIN$          IPC       IPC Service (fileserver server (Samba, Ubuntu))
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.21b]

        Server               Comment
        ---------            -------
        FILESERVER           fileserver server (Samba, Ubuntu)

        Workgroup  Master
        ---------            -------
        WORKGROUP            FILESERVER
```

Map some standard groups used in Windows domains to equivalent Linux groups:

```
$ sudo net groupmap modify ntgroup="Domain Admins" unixgroup=root
$ sudo net groupmap modify ntgroup="Domain Users" unixgroup=users
$ sudo net groupmap modify ntgroup="Domain Guests" unixgroup=nogroup
```

To allow users to authenticate in the domain, they need to be defined in the domain controller. This process needs to be repeated for each user, but for now just create one user to start with by first adding the Linux user and then setting a password for that user in Samba. The user also needs to be placed in the `users` group that was aliased to the Windows group `Domain Users` a moment ago:

```
$ sudo useradd jon -m -G users
$ sudo smbpasswd -a jon
```

That user should now be able to authenticate on workstations within your domain, but there haven't yet been any shares added.

Add Shares

Start by adding a simple share that all users can access. First create the directory that will become the share and set appropriate permissions on it:

```
$ sudo mkdir -p /home/shares/public
$ sudo chown -R root:users /home/shares/public
$ sudo chmod -R ug+rwx,o+rx-w /home/shares/public
```

Each share needs to be configured in Samba. Open */etc/samba/smb.conf* and add a new stanza at the end:

```
[public]
  comment = Public Share
  path = /home/shares/public
  valid users = @users
  force group = users
  create mask = 0660
  directory mask = 0771
  writable = yes
```

Each time you modify the configuration, you need to restart Samba:

```
$ sudo /etc/init.d/samba restart
```

The preceding `public` stanza allows all users in the *@users* group to access the share with full write privileges. This is probably not what you want in many cases. For a share that can be accessed only by specific users, you can substitute a line such as:

```
  valid users = jon,kyle,bill
```

To manage a large number of users, you can alternatively create another separate group, set permissions on the share appropriately, and specify it in the share definition. That way, you can add and remove users from that group without having to reconfigure or restart Samba:

```
  valid users = @authors
```

You can also create read-only shares by setting the `writable` option to `no`:

```
 writable = no
```

Another typical scenario is a share that is read/write by some users but read-only for others:

```
  valid users = @authors,@editors
  read list = @authors
  write list = @editors
```

Samba has a huge range of options for specifying various access restrictions, so refer to the extensive documentation at http://samba.org/samba/docs/ for more details.

Share Printers

If you have printers that you want to make accessible to your Windows workstations through Samba, start by following the steps in "Set Up Your Printer" [Hack #9] to get your printers working locally, and then use *cupsaddsmb* to tell Samba to make them available. To share all printers, run:

```
$ sudo cupsaddsmb -a
```

If you want to share only a specific printer, you can instead refer to its CUPS identity:

```
$ sudo cupsaddsmb laserjet
```

Hack 95. Administer Your Server Remotely

Install and configure SSH to securely connect and administer your server from any machine with a network connection.

Apart from when you are doing the base installation or some sort of local maintenance, generally a Linux server is meant to be run without a monitor connected. Most tasks you would need to perform on a server can be done via the command line, and these days Telnet is out and SSH is in. SSH provides you with the ability to remotely log in to your server and run commands—all over an encrypted channel. Plus SSH offers a number of advanced functions that can make remote administration simpler.

First things first, Ubuntu (at least the desktop version) does not install the SSH server by default, so you will need to install it. Either use your preferred package manager to install the *openssh-server* package or run:

```
$ sudo apt-get install openssh-server
```

The installation scripts included with the package will take care of creating the initial RSA and DSA keys you need, as well as providing you with a good default SSH config. Once the install finishes, you should be able to log in to the machine from other machines on the network by typing:

```
$ ssh
                ip_address
```

(Replace *ip_address* with the IP address or hostname for your remote Ubuntu server.)

Configure SSH

One issue with the default SSH config (*/etc/ssh/sshd_config*) that ships with Ubuntu is that it enables remote *root* logins and X11 forwarding, which create potential security concerns. Since the *root* account is disabled on Ubuntu by default anyway, it doesn't hurt to disable the *root* login option. Just find the line that says:

```
PermitRootLogin yes
```

and change it to say:

```
PermitRootLogin no
```

If you aren't planning on using X11 forwarding, you can disable that as well. Find the line that says:

```
X11Forwarding yes
```

and change it to:

```
X11Forwarding no
```

Once you have made your changes, type:

```
$ sudo /etc/init.d/ssh restart
```

to load the new configuration.

X11 Forwarding

Now while X11 forwarding should be disabled if you aren't planning to use it, if you are planning to use it, it can allow you to do some pretty interesting things. Essentially, X11 forwarding allows you to set up a secure communication channel between you and the remote server over

which you can run graphical applications. The performance of these applications will vary depending on the speed of your network connection. To take advantage of X11 forwarding, add the `-X` argument to *ssh* when connecting to the server:

```
$ ssh -X
                ip_address
```

Then start a graphical program such as *xterm* (which, when you think about it, wouldn't make much sense to run since you're already in a shell) or perhaps Synaptic. Give the application some time if you are over a slower network link, but eventually the graphical program should appear on your local desktop. This feature can be particularly useful if your server needs third-party graphical programs to manage hardware RAID volumes or backup programs and you need to manage these tools remotely.

Configure Passwordless Authentication

If you find yourself connecting to the same machine frequently, or you want to be able to set up a script to run commands on the machine when you aren't around, you will want to set up passwordless authentication. Essentially, this requires that you set up a public and private key on your local machine and then add the public key to a particular configuration file on the remote machine. First generate your keys on the local machine with the *ssh-keygen* program:

```
greenfly@ubuntu:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/greenfly/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/greenfly/.ssh/id_rsa.
Your public key has been saved in /home/greenfly/.ssh/id_rsa.pub.
The key fingerprint is:
b7:db:cc:2c:81:c5:8c:db:df:28:f3:1e:17:14:cd:63 greenfly@ubuntu
```

(If you want to generate DSA keys instead, replace `rsa` with `dsa` in the above command.) Notice that it prompted me for a passphrase. If you want to use this key for passwordless authentication, just hit Enter for no passphrase. When the program finishes, you will have two new files in your *~/.ssh/* directory called *id_rsa* and *id_rsa.pub*, which are your private and public keys, respectively. Now, for security reasons, the *id_rsa* file is set to be readable only by your user, and you should never share that file with anyone else, since she would then be able to log in to any machine that you could with that key. The public key is the one you will share with remote servers to allow you to log in passwordlessly. To do so, copy the *id_rsa.pub* file to the remote machine and append its contents to the *~/.ssh/authorized_keys* file. You can do this step by step, or you can use this handy one-liner to do it all in one fell swoop:

```
$ ssh
                user@remotehost
             "cat >> ~/.ssh/authorized_keys" < ~/.ssh/id_rsa.pub
```

Replace `user@remotehost` with your username and the hostname of the remote server. You will be prompted for your password one final time, and then the key will be appended to the remote host's *authorized_keys* file. You should now be able to *ssh* to the remote machine and log in without being prompted for a password.

If you do include a passphrase when you generate the key, you will be prompted for that key's passphrase every time you log in. You can make this step almost as convenient as passwordless login by running the command `ssh-agent bash` on your local machine; this starts up a bash shell session under the control of an SSH agent process. You can then add your key to the agent with *ssh-add*. You'll be prompted once for your password, and then you can *ssh* to *remotehost* without being prompted for your password, unless you exit that shell that you started with *ssh-agent*.

Copy Files Securely

Another common need is to be able to copy files between servers you are administering. While you could set up FTP on all of the servers, this is a less-than-ideal and potentially insecure solution. SSH includes within it the capability to copy files using the *scp* command. This has the added benefit of copying the files over a secure channel along with taking advantage of any key-based authentication you might have already set up.

To copy a file to a remote machine, type:

```
$ scp /path/to/file user@remotehost:
                /path/to/destination
```

Or, if you need to copy from the remote host to the local host, reverse the two arguments:

```
$ scp user@remotehost:/path/to/file
                /path/to/destination
```

scp supports recursion, so if you need to copy an entire directory full of files to a remote location use the `-r` argument:

```
$ scp -r
                /path/to/directory/ user@remotehost:/path/to/destination/
```

If you are transferring logfiles or other highly compressible files, you might benefit from the `-C` argument. This turns on compression, which, while it will increase the CPU usage during the copy, should also increase the speed in which the file transfers, particularly over a slow link.

Alternatively, if you want to copy a file but can't afford to saturate your upload with the transfer, use the `-l` argument to limit how much bandwidth is used. Follow `-l` with the bandwidth you want to use in kilobits per second. So, to transfer a file and limit it to 256 Kbps, type:

```
$ scp -l 256
                /path/to/file user@remotehost:/path/to/destination
```

Hack 96. Build a Web Server

Serve web content using the massively popular and capable Apache web server.

Ubuntu makes an ideal web-server platform, with Apache and a huge range of supporting software available quickly and easily from the official Ubuntu archives. But just installing the software gets you only halfway there: with a few small tweaks you can have a very flexible and capable web hosting environment.

Install Apache

First, install Apache:

```
$ sudo apt-get install apache2
```

Then make sure Apache is running:

```
$ sudo /etc/init.d/apache2 restart
```

The Apache installation will create a directory at */var/www*, which is the *document root* of the default server. Any documents you place in this directory will be accessible via a web browser at http://localhost/ or the IP address assigned to your computer.

Install PHP

PHP is a server-side scripting language that is commonly used by content-management systems, blogs, and discussion forums, particularly in conjunction with either a MySQL or Postgres database:

```
$ sudo apt-get install libapache2-mod-php5
```

Restart Apache to make sure the module has loaded:

```
$ sudo /etc/init.d/apache2 restart
```

To check that the module is loaded properly, create a PHP file and try accessing it through the web server. PHP has a built-in function called `phpinfo` that reports detailed information on its environment, so a quick way to check if everything is working is to run:

```
sudo sh -c "echo '<?php phpinfo( ); ?>' > /var/www/info.php"
```

and then point your browser at http://localhost/info.php to see a page showing the version of PHP that you have installed.

One possible problem at this point is that your browser may prompt you to download the file instead of displaying the page, which means that Apache has not properly loaded the PHP module. Make sure there is a line in either */etc/apache2/apache2.conf* or */etc/apache2/mods-enabled/php5.conf* similar to:

```
AddType application/x-httpd-php .php .phtml .php3
```

If you make that change, you'll need to stop and start Apache manually to make sure it re-reads the configuration file:

```
$ sudo /etc/init.d/apache2 stop
$ sudo /etc/init.d/apache2 start
```

Configure Dynamic Virtual Hosting

Web servers typically host multiple web sites, each with its own *virtual server*, and Apache provides support for the two standard types of virtual server: IP-based and name-based.

- *IP-based* virtual servers use a separate IP address for each web site. This approach does have some advantages, but due to the shortage of IPv4 addresses, it's usually used only as a last resort, such as when SSL (Secure Sockets Layer) encryption is required.
- *Name-based* virtual servers share a single IP address between multiple web sites, with the server using the `Host:` header from the HTTP request to determine which site the request is intended for. The usual way to achieve this is to create a configuration for each virtual server individually, specifying the name of the host and the directory to use as the "root" of the web site. However, that means you have to modify the Apache configuration and restart it every time you add a new virtual server.

Dynamic virtual hosting allows you to add new virtual hosts at any time without reconfiguring or restarting Apache by using a module called *vhost_alias*. Enable *vhost_alias* by creating a symlink in Apache2's *mods-enabled* directory:

```
$ sudo ln -s /etc/apache2/mods-available/vhost_alias.load \\
             /etc/apache2/mods-enabled/vhost_alias.load
```

To allow the module to work, there are some changes that need to be made to */etc/apache2/apache2.conf* to turn off canonical names, alter the logfile configuration, and specify where your virtual hosts will be located. Add or alter any existing settings to match the following:

```
# get the server name from the Host: header
UseCanonicalName Off

# this log format can be split per virtual host based on the first field
LogFormat "%V %h %l %u %t "%r" %s %b" vcommon
CustomLog /var/log/apache2/access_log vcommon

# include the server name in the filenames used to satisfy requests
```

```
VirtualDocumentRoot /var/www/vhosts/%0/web
VirtualScriptAlias /var/www/vhosts/%0/cgi-bin
```

Create the directory that will hold the virtual hosts:

```
$ sudo mkdir /var/www/vhosts
```

Create a skeleton virtual server:

```
$ sudo mkdir -p /var/www/vhosts/skeleton/cgi-bin
$ sudo cp -a /var/www/apache2-default /var/www/vhosts/skeleton/web
```

Restart *apache2* so the configuration changes take effect:

```
$ sudo /etc/init.d/apache2 restart
```

You are now ready to create name-based virtual hosts by copying the skeleton to the hostname you want it to respond to. For example, to create a new virtual server for *www.example.com*, you would simply run:

```
$ sudo cp -a /var/www/vhosts/skeleton /var/www/vhosts/
                www.example.com
```

Any HTTP connections made to your server with the `Host:` header set to *www.example.com* will now be answered out of that virtual server.

To make the virtual hosts accessible to other users, you will need to put appropriate entries in a publicly accessible DNS server and have the domains delegated to it, but for a quick local test you can edit your */etc/hosts* file and add an entry similar to:

```
127.0.0.1 www.example.com
```

Hack 97. Build an Email Server

Setting up an email server is remarkably straightforward, but there are a couple of things to be very careful of so it doesn't end up being a haven for spammers.

An email server consists of several components: an SMTP (Simple Mail Transport Protocol) server to handle mail transfer between hosts, POP and IMAP servers to give users access to mailboxes from their desktop mail clients, and often some kind of mail filtering system for reducing spam and viruses passing through the system.

Postfix SMTP Server

There are many SMTP servers available in Ubuntu, and many administrators have their own personal preference, but the Postfix SMTP server is a good general-purpose choice that is fast, secure, and extensible:

```
$ sudo apt-get install postfix
```

The installation process will ask some questions about how the system will operate. Select Internet Site as the operation mode and set Mail Name to your domain.

Once the package has been installed, open */etc/postfix/main.cf* in an editor and find a line like:

```
mynetworks = 127.0.0.0/8
```

To allow computers on your network to send outgoing email through the server, you need to add your network range to the `mynetworks` value. For example, if your network is the 192.168.0.0 class-C range, you would edit the line to read:

```
mynetworks = 127.0.0.0/8 192.168.0.0/24
```

This setting is critical to preventing your mail server being used as a relay by spammers, so only add network ranges that you trust.

When mail is delivered to a local user, it can be stored in several different ways. The older and most common approach is the *mbox* format, which stores all mail in a single file for each user, but the performance of the mbox format falls off dramatically with large mail volumes. Most newer mail systems use the *maildir* format which stores messages in individual files nested inside directories. Postfix can handle either format equally well. Add this line to *main.cf* to use the maildir format:

```
home_mailbox = Maildir/
```

The `Maildir/` value is appended to the home directory path of the recipient, and the trailing slash indicates to use the maildir format for storage.

Finally, look for a line that starts with `mydestination =` . Mail for all domains listed in this line will be accepted by your mail server and local delivery will be attempted, so if you will host mail for multiple domains, add them here.

Restart Postfix to make your changes take effect:

```
$ sudo /etc/init.d/postfix restart
```

If you will be using your mail server only as an outbound mail gateway, that's all you need to do. Configure your email client to use your mail server for outbound mail and try sending a message to an external email account.

If the message doesn't come through, try "putting a tail" on the Postfix logfile to see what went wrong, and adjust your configuration as necessary:

```
$ sudo tail -f /var/log/mail.log
```

Reduce Spam With Greylisting

There are a variety of methods to protect your users from spam, but unfortunately there is no magic solution that causes absolutely no false positives or negatives. Greylisting is one approach that requires very little ongoing maintenance but has a very high success rate with very few false positives where valid email is mistakenly rejected.

Greylisting works on the premise that valid mail servers will attempt redelivery of mail if they receive a "temporarily unavailable" error from the destination server, while spam hosts and viruses will typically only attempt delivery once and then move on to the next target. This means legitimate mail from a remote system will be delayed, but afterwards your mail server will remember that the sender is valid and let the mail straight through. The delay on the first message can be inconvenient, but on the whole, greylisting is one of the most successful spam-mitigation techniques currently available. To take advantage of greylisting, install Postgrey

```
$ sudo apt-get install postgrey
```

Postgrey runs as a daemon on your mail server on port 60000, so configure Postfix to use it as a delivery policy service. Open */etc/postfix/main.cf* and add an entry for the service:

```
smtpd_recipient_restrictions =
        reject_unauth_destination,
        check_policy_service inet:127.0.0.1:60000
```

Then restart Postfix and put a tail on the Postfix logfile before sending a test message to the system from an external mail server. On the first delivery attempt, you will see the message rejected with a nonfatal error, and then after five minutes your mail server will allow the message to be delivered. Subsequent messages from the same remote system will be delivered immediately.

Activity Reporting

To see how much traffic your mail server is handling, install the *mailgraph* package and start it up:

```
$ sudo apt-get install mailgraph
$ sudo /etc/init.d/mailgraph start
```

Mailgraph watches mail-server activity and logs it in an extremely efficient database, and then builds graphs that you can access through a web browser at http:// *yourhost* /cgi-bin/mailgraph.cgi. By default, the graphs are accessible from anywhere, so if you prefer to keep them secret, you may wish to restrict access to them using an Apache *.htaccess* file or with explicit access control in the Apache configuration.

POP And IMAP Services

To allow users to collect mail from the server, you need to run IMAP and/or POP services. Once again, there is a variety of alternatives, each of which have advantages and disadvantages, but the Courier suite provides very simple setup and natively supports maildir format:

```
$ sudo apt-get install courier-imap courier-imap-ssl \\
             courier-pop courier-pop-ssl
```

If you configured Postfix to use maildirs as described above, you don't need to make any changes to the Courier configuration: it will automatically detect the maildirs, and everything should just work.

Hack 98. Build A Caching Proxy Server

If you have multiple computers on your network you can save bandwidth and improve browser performance with a local proxy server.

A proxy server sits on your network; intercepts requests for HTML files, CSS files, and images; and keeps a local copy handy in case another user wants to access the same file. If multiple users visit the same site, a proxy server will save bandwidth by not downloading everything to your local network for each user individually, and performance will be improved because objects will come from the local network instead of the Internet.

The Squid Web Proxy Cache (www.squid-cache.org) is a full-featured proxy cache for Linux and Unix.

Basic Squid Setup

Install the Squid caching proxy:

```
$ sudo apt-get install squid
```

The installation process will automatically create a directory structure in */var/spool/squid* where downloaded objects will be stored. Old objects will be cleaned out automatically, but if you run a busy proxy server, it can still use up a lot of disk space so make sure you have plenty of room available.

Squid's default configuration file */etc/squid/squid.conf* is one of the longest and most verbosely commented in the entire history of software: over 3000 lines, with an extensive explanation for every possible config option. It's easy to get lost in it, so to get started, here are some basic options you need to look for.

Around line 1890 are some options that trip up most first-time Squid administrators. Squid implements ACLs (Access Control Lists) to determine who is allowed to connect through the proxy. By default, the only system allowed to connect is *localhost*:

```
#acl our_networks src 192.168.1.0/24 192.168.2.0/24
#http_access allow our_networks
http_access allow localhost
```

To allow machines on your network to connect, you need to uncomment and edit the `our_networks` definition to include the IP address range of your local network, and uncomment the line that permits the `our_networks` ACL to use the proxy. The end result will probably be something like this:

```
acl our_networks src 192.168.0.0/24
http_access allow our_networks
http_access allow localhost
```

Then go to approximately line 53 to find the `http-port` option:

```
# http_port 3128
```

This option specifies the port that Squid will listen on. 3128 is a good default, but some proxies run on port 8080 or even port 80, so you may prefer to change the value and uncomment it.

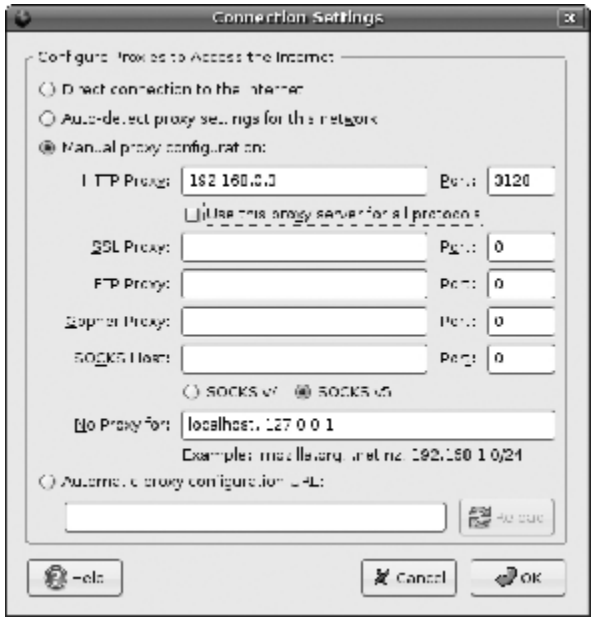
Once you are satisfied with your changes, restart Squid:

```
$ sudo /etc/init.d/squid restart
```

Restarting Squid can take a while on an active proxy because it waits for existing connections from clients to close cleanly before restarting.

You can test your proxy by manually updating your Firefox configuration to connect through it. In Firefox go to Edit→Preferences→General→Connection Settings, select "Manual proxy configuration," and put in the details for your proxy server, as shown in Figure 10-1.

Figure 10-1. Browser proxy settings



To see the activity passing through the proxy, put a tail on the Squid logfile and then try accessing a web site. Squid stores its access logs in */var/log/squid*, so run

```
$ sudo tail -f /var/log/squid/access.log
```

to have *tail* "follow" the end of the logfile. If the web page loads normally and you also see entries appear in the logfile then congratulations, Squid is working!

Proxy Traffic Reports

The popular web-server-analysis program Webalizer can read Squid logfiles natively. Install Webalizer:

```
$ sudo apt-get install webalizer
```

Then use your favorite text editor to open */etc/webalizer.conf*, and look around line 36 for an entry like this:

```
LogFile    /var/log/apache/access.log.0
```

Change it to reference Squid's rotated logfile:

```
LogFile    /var/log/squid/access.log.0
```

Around line 42, you will see the option to set the directory where the report will be created. If you have a default Apache installation on your proxy server you shouldn't need to change the default setting, but if your web document root is in an alternative location or you already have a report being generated for your web server, you may need to change it:

```
OutputDir    /var/www/webalizer
```

If you've only just installed and tested Squid, you probably won't have a rotated logfile yet, so manually rotate the file:

```
$ sudo /etc/init.d/squid stop
```

The output directory is not created automatically so you'll need to do it manually:

```
$ sudo mkdir /var/www/webalizer
```

Now run Webalizer:

```
$ sudo webalizer
```

When it's finished, you'll find a bunch of files in */var/www/webalizer*, and you should be able to view the report by pointing your browser at http://yourcache.example.com/webalizer/.

Peering Proxies

If your ISP provides a proxy, you can chain it together with your Squid proxy. Your local clients will connect to your proxy, which in turn will use your ISP's proxy. In the Squid configuration file, go to about line 190 and add a line similar to:

```
cache_peer cache.example.com parent 3128 0 no-query
```

where *cache.example.com* is the address of your ISP's cache. The `parent` setting tells your proxy to treat this as an upstream source rather than a local peer. You may need to change the `3128` setting if your ISP uses a different proxy port. The `0` and `no-query` values tell your proxy not to use ICP (Internet Cache Protocol) to communicate with the cache. ICP is a protocol typically used when multiple proxies run in parallel as a load-sharing group, and allows them to communicate cache state to each other very rapidly.

Restart Squid, put a tail on the logfile again, and try accessing a popular site. If the upstream proxy already had some of the items in its cache, you should see this reported as `PARENT_HIT` in your proxy log.

Hack 99. Build A DHCP Server

Use a DHCP server to automatically configure the network settings for all computers on your network.

DHCP (Dynamic Host Configuration Protocol) dramatically simplifies the connection of new computers to your network. With a properly configured DHCP server, any new computers you connect will automatically be assigned an IP address, the address of your router, and nameserver addresses. And to really make things easy on yourself, you can link your DHCP server to the BIND9 DNS server and have new computers automatically assigned a hostname that maps correctly to its dynamically assigned IP address.

Install the DHCP Daemon

First make sure you don't already have a DHCP server running on your network; two servers providing conflicting information is a recipe for obscure network problems! Install the Internet Software Consortium (ISC) DHCP server:

```
$ sudo apt-get install dhcp3-server
```

Basic Configuration

Open the configuration file */etc/dhcp3/dhcpd.conf*, where you will see various configuration options that apply both globally and to specific subnets. The majority of the sample options included in the file are quite self-explanatory, so put appropriate entries in the global settings, and then add a basic stanza for your network:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
        range 192.168.0.20 192.168.0.50;
        option routers 192.168.0.1;
}
```

The `range` setting specifies the pool of IP addresses to use when new computers connect to your network, and the `routers` option is passed on so they can add a default route to use to connect to the Internet.

Assign Addresses to Specific Hosts

Sometimes it can be helpful to force specific IP addresses to be associated with certain hosts, such as printers. When a host connects to the DHCP server, it provides the MAC (Media Access Control) address of the network interface, and the DHCP server can then use that to associate the host with a specific configuration.

If you don't know the MAC address of your computer, you can find it printed on a label on most Ethernet cards; network printers often have it labeled somewhere near the Ethernet connector. On Linux, you can obtain it using *ifconfig*:

```
$ /sbin/ifconfig eth0 | grep HWaddr
```

Back on the DHCP server, open */etc/dhcp3/dhcpd.conf* and add a stanza near the end for each host:

```
host workstation51 {
  hardware ethernet 08:00:07:26:c0:a5;
  fixed-address 192.168.0.51;
}
```

Make sure the `fixed-address`es you set don't fall within a range that has been nominated for general assignment.

Finally, restart the DHCP server so your configuration will take effect:

```
$ sudo /etc/init.d/dhcp3-server restart
```

Hacking the Hack

DNS provides a hostname-to-IP-address resolution service so you don't need to care what actual IP address has been assigned to a computer, but DHCP allows IP addresses to be dished out semi-randomly to machines on your network, which makes it very hard to maintain sensible DNS entries. However, if you use BIND9 to build a domain name server [Hack #100], you can link it to your DHCP server and have DNS records updated automatically each time a computer joins or leaves your network.

First get your DNS and DHCP servers functioning correctly independently. Once you are happy that they are doing what they are meant to, open the BIND9 configuration file (*/etc/bind/named.conf.options*) and add a new stanza at the end:

```
controls {
        inet 127.0.0.1 allow {localhost; } keys { "rndc-key"; };
};
```

The `localhost` setting specifies that only local processes are allowed to connect, and `rndc-key` is the name of a secret key that will be used to authenticate connections. The actual key is stored in */etc/bind/rndc.key*, which is pre-populated with a randomized key value when the *bind9* package is installed. If your DNS and DHCP servers are on the same physical machine, these settings will work nicely, but if they are on different machines, you will need to tell BIND to allow connections from your DHCP host and copy the key file across. Open */etc/bind/*

named.conf.local, add forward and reverse zones for your local network, and specify that these zones can be updated by clients that know the secret key:

```
zone "example.com" {
        type master;
        file "/etc/bind/zones/example.com.hosts";
        allow-update { key "rndc-key"; };
        notify yes;
};

zone "0.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/zones/192.168.0.hosts";
        allow-update { key "rndc-key"; };
        notify yes;
};
```

Set up the zone files for *example.com.hosts* and *192.168.0.hosts* as usual, including any statically assigned hostname values

You also need to tell BIND to load the key file, so after the `zone` stanzas, add an `include` line:

```
include "/etc/bind/rndc.key";
```

Once you restart BIND, it will be ready to accept dynamic zone updates:

```
$ sudo /etc/init.d/bind9 restart
```

Your DHCP server now needs to be told to send update notifications to your DNS server. Open */etc/dhcp3/dhcpd.conf* and add these entries to the top of the file:

```
server-identifier           server;
ddns-updates                on;
ddns-update-style           interim;
ddns-domainname             "example.com.";
ddns-rev-domainname "in-addr.arpa.";
ignore                      client-updates;
include     "/etc/bind/rndc.key";

zone example.com. {
    primary 127.0.0.1;
 key rndc-key;
}
```

You may need to comment out existing settings that conflict, such as the `ddns-update-style none;` option included in Ubuntu's default DHCP configuration.

Restart DHCP to apply your changes:

```
$ sudo /etc/init.d/dhcp3-server restart
```

From now on, any hosts that register themselves with DHCP will also be automatically added in-memory to your DNS zone.

Hack 100. Build a Domain Name Server

Run your own DNS server to map hostnames to IP addresses.

The Domain Name System (DNS) is a distributed directory service that maps machine hostnames to IP addresses and vice versa. DNS allows hostnames to be just "pointers" to the actual network location of the server, providing a consistent human-readable hostname even if the actual IP address changes.

Understand DNS in 60 Seconds

The reason DNS is called a "distributed" service is that there is no single machine that contains a comprehensive lookup table for the entire Internet. Instead DNS functions as a tree, with root servers distributed around the world that look after the top-level domains (TLDs) such as *.com*. You can find more information about the current root servers at http://www.root-servers.org. The area that each nameserver is responsible for is called a *zone*, and the details of each zone are typically stored in a configuration file called a *zonefile*.

At each level, a DNS server can delegate authority for part of its zone below itself, so the root servers delegate authority for *.au* to certain Australian nameservers, which in turn delegate authority for *.com.au* to other nameservers, which then delegate authority for *.oxer.com.au* to my nameservers, which then manage specific host records such as *jon.oxer.com.au* and provide mappings to IP addresses. The system is very hierarchical and allows the authority to manage specific hostname data by delegating it right out to the edges of the Internet.

Note that there is nothing stopping you from setting up a domain name server and putting any data in it you like: you could put in an entry for www.microsoft.com that points to the www.oreilly.com server if you wanted to, and if you used that DNS server, that's exactly what you would see. However, unless your DNS server is part of the global namespace that comes under the authority of the root nameservers, nobody else will ever see the records you put in it. It's not enough to just set up a nameserver: you need to have domains "delegated" to your nameserver by the appropriate upstream authority so that other computers will know your server is authoritative for that domain. Otherwise, you are just running an *orphan zone*.

There are, however, complete alternative DNS namespaces that have been set up outside the usual root servers, but they are accessible only to people using nameservers that have been specially reconfigured. Using these alternative namespaces, you can register domains in *.indy*, *.parody*, *.job*, and even *.www*, but you'll be cut off from the vast majority of users on the Net.

One final point to be very careful of is that, strictly speaking, domains actually end in a "." (period) character, although the final dot is implied and most Internet users don't even realize that it should be there. Try it yourself: point your browser at www.oreilly.com. including the final dot and see what happens. If everyone were being technically correct, that final dot would be included on all URLs, but things just work anyway because DNS servers assume we're just being lazy and treat our URLs as if the dot were silently appended. To most people, that's just a piece of useless Net trivia, but once you start configuring your own DNS server, it becomes critical —so keep it in mind.

DNS is actually a very complex subject that can't really be understood in a mere 60 seconds, but the critical things to remember are that it's structured as a hierarchical tree starting from ".", that zones are delegated down the nameserver hierarchy and become more specific at each level, and that zones can map hostnames to IP addresses and vice versa.

Authoritative and Recursive Lookups

When a computer needs to look up a hostname and convert it to an IP address, there are two types of lookups that may be performed.

An *authoritative lookup* is a query to a nameserver that can answer the request directly from its own knowledge of what does or does not exist in that zone. For example, if you queried the root nameservers for the host www.oreilly.com, they would not be able to answer authoritatively because they do not contain specific information about the hosts in that zone. However, a query to O'Reilly's own nameservers would return an authoritative answer. Nameservers at hosting companies typically spend most of their time providing authoritative answers about zones they manage.

A *recursive lookup* involves a query to a nameserver that does not specifically know about the requested hostname, but which can then work through the DNS tree to obtain the answer before returning it to the requesting computer. Nameservers at ISPs typically spend most of their time performing recursive lookups on behalf of users rather than serving authoritative answers.

Authoritative and recursive lookups are actually totally different operations, and so there is specialized DNS server software available for each type of query. It's not uncommon for a DNS server to run two different packages to handle authoritative and recursive queries.

Install BIND9

For a general-purpose DNS server, a good software choice is BIND (the Berkeley Internet Name Daemon), which is a very popular DNS server that can handle both authoritative and recursive lookups natively:

```
$ sudo apt-get install bind9
```

If all you want is a recursive DNS service, that's actually all you need to do. If you look in */etc/bind/db.root*, you'll find that BIND has been seeded with the latest IP addresses of the root nameservers, allowing it to look up delegation information and issue recursive lookup requests on behalf of other computers right out of the box.

You can test it from a Linux machine without changing your nameserver configuration by specifying the address of your DNS server and performing a manual lookup using the *nslookup* tool, which is in the *dnsutils* package:

```
jon@jbook:~$ nslookup jon.oxer.com.au 192.168.0.2
Server:         192.168.0.2
Address:        192.168.0.2#53

Non-authoritative answer:
Name:   jon.oxer.com.au
Address: 202.91.207.154
```

As you can see, the result was returned nonauthoritatively because the server had to refer to an external source to obtain the answer. If that worked, you can edit */etc/resolv.conf* on your workstations and have them use your DNS server for lookups.

Create an Authoritative Forward Zone

Authoritative nameservers come in two types: *master* and *slave*. A master nameserver is explicitly configured with all the details of the zones it manages, while a slave is simply told the names of the zones and pointed at a master to periodically refresh its locally cached copies of the zone by performing a *zone transfer*. In this hack, you'll learn how to configure a master nameserver.

In fact there's nothing to stop you running all your nameservers as masters; as long as you keep all their configurations synchronized, everything will work fine. External machines doing lookups don't know the difference between a master and a slave: a master/slave setup is purely a convenience issue.

The most master BIND configuration file is */etc/bind/named.conf*. Rather than modify it directly, though, it's best to keep your customizations in separate files and have them "included" into the main configuration. The default installation on Ubuntu includes */etc/bind/named.conf.local*, which you can use to define your own zones.

To keep everything neat, create a subdirectory to store your actual zone files in:

```
$ sudo mkdir /etc/bind/zones
```

Now create a zone file for your zone named after the zone itself, such as */etc/bind/zones/example.com.hosts*, and put in the file something like the following:

```
example.com. IN      SOA     ns1.example.com. hostmaster.example.com. (
          2001061407  ; serial
          10800       ; refresh
          3600        ; retry
          432000      ; expire
          38400 )     ; ttl
example.com. IN      NS      ns1.example.com.
example.com. IN      NS      ns2.example.com.
example.com. IN      MX      30 mail.example.com.
www.example.com.     IN      A       202.91.207.152
mail.example.com.    IN      A       202.91.207.152
```

The first line specifies the zone, the Start Of Authority as the nameserver *ns1.example.com*, and the administrative contact as *hostmasterexample.com*. Notice that the @ symbol is replaced by a dot in the zone file: BIND treats the first item in the string as the username and the rest as the domain. The subsequent values specify how the zone should be treated by other nameservers, such as how long results can be cached.

The `NS` records specify the nameservers that are authoritative for this zone, the `MX` record specifies the mail exchange host for this domain along with a priority from 1 to 100 (lower numbers being more important), and the `A` records map specific hostnames to IP addresses.

Note that the full hostnames in the zone file all end in a period, and this is where properly specifying hostnames becomes critical. You might leave the dot off the end of URLs when you type them into your browser, but you can't be ambiguous in the zone file! If you leave the final dot off, BIND assumes the hostname has not been explicitly terminated and appends the domain to it, leaving you with addresses like *www.example.com.example.com*. You can take advantage of this behavior by deliberately leaving off the domain entirely and specifying just the first part of the hostname without a trailing dot:

```
www             IN      A 202.91.207.152
```

For BIND to know about your new zone file, you need to edit */etc/bind/named.conf.local* and add an entry at the end similar to:

```
zone "example.com" {
        type master;
 file "/etc/bind/zones/example.com.hosts";
        };
```

Then restart BIND:

```
$ sudo /etc/init.d/bind9 reload
```

Now if you try a query against the nameserver for a host in your zone, you will see the result shows your IP address and isn't flagged as "nonauthoritative":

```
jon@jbook:~$ nslookup www.example.com 192.168.0.2
Server:         192.168.0.2
Address:        192.168.0.2#53

Name:   www.example.com
Address: 202.91.207.152
```

Firewall Rules

If you set up a firewall [Hack #69], you will need to add specific rules to allow queries from external machines to reach it. DNS queries are sent on port 53 using UDP by default, falling back to TCP if the request packet exceeds 512 bytes in size. You therefore need to allow both UDP and TCP on port 53 through your firewall to your DNS server.