Matthew Helmke

Ubuntu 16.04 LTS on DVD

Ubuntu

UNLEASHED

2017 Edition

Covering 16.10, 17.04, and 17.10

SAMS

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Matthew Helmke
with Andrew Hudson
and Paul Hudson

Ubuntu

UNLEASHED

2017 Edition

SAMS | 800 East 96th Street, Indianapolis, Indiana 46240 USA

Ubuntu Unleashed 2017 Edition

Visit us on the Web: informit.com/aw

ISBN-13: 978-0-13-451118-4

ISBN-10: 0-13-451118-2

Library of Congress Cataloging-in-Publication Data: 2016951091

Printed in the United States of America

Text printed in the United States on recycled paper at RR Donnelley, Crawfordsville, Indiana.

First Printing October 2016

Editor-in-Chief
Mark Taub

Acquisitions Editor
Debra Williams Cauley

Managing Editor
Sandra Schroeder

Project Editor
Lori Lyons

Production Manager
Dhayanidhi

Proofreader
Sasirekha

Technical Editor
José Antonio Rey

Editorial Assistant
Kim Boedigheimer

Media Producer
Dan Scherf

Cover Designer
Chuti Prasertsith

Compositor
codeMantra

Contents at a Glance

NOTE

Chapters 45–47 can be accessed online at informit.com/title/9780134268118.

Table of Contents

NOTE

Chapters 45–47 can be accessed online at informit.com/title/9780134268118.

Bonus Chapters

About the Author

Matthew Helmke is an active member of the Ubuntu community. He served from 2006 to 2011 on the Ubuntu Forum Council, providing leadership and oversight of the Ubuntu Forums (www.ubuntuforums.org), and spent two years on the Ubuntu regional membership approval board for Europe, the Middle East, and Africa. He has written about Ubuntu for several magazines and websites, is a lead author of *The Official Ubuntu Book*, and coauthored *The VMware Cookbook*. He works as a technical writer for Canonical, Inc., documenting cloud- and DevOps-related software. Matthew first used Unix in 1987 while studying LISP on a VAX at the university. He has run a business using only free- and open-source software, has consulted, and has a master's degree in Information Resources and Library Science from the University of Arizona. You can find out more about Matthew at matthewhelmke.com or drop him a line with errata or suggestions at matthew@matthewhelmke.com.

Andrew Hudson is a freelance journalist who specializes in writing about Linux. He has significant experience in Red Hat- and Debian-based Linux distributions and deployments and can often be found sitting at his keyboard tweaking various settings and config files just for the hell of it. He lives in Wiltshire, which is a county of England, along with his wife, Bernice, and their son, John. Andrew does not like Emacs. He can be reached at andy.hudson@gmail.com.

Paul Hudson is a recognized expert in open source technologies. He is a professional developer and full-time journalist for Future Publishing. His articles have appeared in *MacFormat*, *PC Answers*, *PC Format*, *PC Plus*, and *Linux Format*. Paul is passionate about free software in all its forms and uses a mix of Linux and BSD to power his desktops and servers. Paul likes Emacs. Paul can be contacted through http://hudzilla.org.

Dedication

To Saralyn, Sedona, and Philip—the most amazing kids a guy could hope for; to Sandra and Evan, who are wonderful and welcome additions to our lives; to my grandfather for always believing in me and for teaching me to believe in myself; and to my friends in the Ubuntu, developer, sysadmin, cloud computing, and DevOps communities.

Acknowledgments

Matthew wishes to thank the many people who helped with past editions, with helpful comments and ideas, with technical edits, and with both formal and informal advice. I owe a huge debt of gratitude to the Ubuntu community, Canonical, and Mark Shuttleworth for inviting me to participate in in the community, including my role in the forums, a turn on the EMEA membership board, and two Ubuntu Developer Summits, back when we had to travel to be a part of them. Thanks to the Ubuntu All Stars for the chance to jam with you on guitar. Thank you to the entire Ubuntu community for your labor of love to create this wonderful operating system. Finally, thanks to my colleagues at Pearson, especially Debra Williams Cauley, for the trust placed in me and the opportunity to collaborate on projects like this one.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: consumer@samspublishing.com

Mail: Sams Publishing
ATTN: Reader Feedback
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

This page intentionally left blank

Introduction

IN THIS INTRODUCTION

- Licensing
- Who This Book Is For
- What This Book Contains
- Conventions Used in This Book

We are pleased to present the 2017 edition of *Ubuntu Unleashed*. Ubuntu is a Linux-based computer operating system that has taken the world by storm. From its humble beginning in 2004, Ubuntu has risen to be the vanguard of desktop Linux, as well as a popular choice for servers.

Ubuntu descends from one of the oldest and most revered Linux distributions, Debian. Debian is assembled by a team of talented volunteers, is one of the most stable and customizable distributions of Linux, and is well respected for its quality and technological prowess. It is, however, an operating system for geeks; the bar for entry into the Debian realm is set high, and its userbase tends to be highly proficient and expects new users to learn the ropes before joining in. That is both appropriate and okay.

What Ubuntu has done is leverage the quality of Debian to create an operating system that ordinary people can use. That doesn't mean that Ubuntu users are not technologically proficient, just that they do not have to be. In fact, many talented and respected software developers love Ubuntu because it enables them to concentrate on their specific interests instead of the details of the operating system. This book is for these people and for those who aspire to join their ranks.

If you are new to Linux, you have made a great decision by choosing this book. Sams Publishing's *Unleashed* books offer an in-depth look at their subjects, taking in both beginner and advanced users and moving them to a new level of knowledge and expertise. Ubuntu is a fast-changing distribution that has an updated release twice a year. We have tracked the development of Ubuntu from early on to make sure that the information in this book mirrors closely the

development of the distribution. A full copy of Ubuntu is included on the enclosed disc, and it is possible for you to install Ubuntu from that disc in less than an hour!

A QUICK WORD ABOUT MARKETING

Almost all of the content in this book applies regardless of what Ubuntu release version you are using, so long as it is reasonably current. The book has been written to try to focus on information that is useful for the longest amount of time possible. Some chapters, like those covering installation or the basics of the default Ubuntu graphical user interface, will have their information change frequently. Those chapters are the exception. The blurb on the cover of the book about which editions this book covers was added to account for these chapters and to denote clearly when the book was most recently revised.

Do not let the highly technical reputation of Linux discourage you, however. Many people who have heard of Linux think that it is found only on servers, looking after websites and email. Nothing could be further from the truth. Distributions like Ubuntu are making huge inroads in to the desktop market. Corporations are realizing the benefits of running a stable and powerful operating system that is easy to maintain and easy to secure. The best part is that as Linux distributions make improvements, the majority of those improvements are shared freely, allowing you to benefit from the additions and refinements made by one distribution, such as Red Hat, while continuing to use a different distribution, such as Ubuntu, which in turn shares its improvements. You can put Ubuntu to work today and be assured of a great user experience. Feel free to make as many copies of the software as you want; Ubuntu is freely and legally distributable all over the world—no copyright lawyers are going to pound on your door.

Licensing

Software licensing is an important issue for all computer users and can entail moral, legal, and financial considerations. Many consumers think that purchasing a copy of a commercial or proprietary operating system, productivity application, utility, or game conveys ownership, but this is not true. In the majority of cases, the *end user license agreement (EULA)* included with a commercial software package states that you have paid only for the right to use the software according to specific terms. This generally means you may not examine, make copies, share, resell, or transfer ownership of the software package. More onerous software licenses enforce terms that preclude you from distributing or publishing comparative performance reviews of the software. Even more insidious licensing schemes (and supporting legislation, especially in the United States) contain provisions allowing onsite auditing of the software's use!

This is not the case with the software included with this book. You are entirely free to make copies, share copies, and install the software on as many computers as you want—we encourage you to purchase additional copies of this book to give as gifts, however. Be sure to read the README file on the disc included with this book for important information regarding the included software and disk contents. After you

install Ubuntu, go to www.gnu.org/licenses/gpl.html to find a copy of the GNU GPL. You will see that the GPL provides unrestricted freedom to use, duplicate, share, study, modify, improve, and even sell the software.

You can put your copy of Ubuntu to work right away in your home or at your place of business without worrying about software licensing, per-seat workstation or client licenses, software auditing, royalty payments, or any other type of payments to third parties. However, be aware that although much of the software included with Ubuntu is licensed under the GPL, some packages on this book's disc are licensed under other terms. There is a variety of related software licenses, and many software packages fall under a broad definition known as *open source*. Some of these include the Artistic License, the BSD License, the Mozilla Public License, and the Q Public License.

For additional information about the various GNU software licenses, browse to www.gnu.org/. For a definition of open source and licensing guidelines, along with links to the terms of nearly three dozen open-source licenses, browse to www.opensource.org/.

Who This Book Is For

This book varies its coverage from deep to shallow over its wide range of topics. This is intentional. There are some topics that are Ubuntu-specific and are not covered by any other book, and so deserve deep coverage here. There are some topics that every power user really must master. There are other topics that power users should know about, so that they understand some history, know some other options, or simply have what they need to be able to listen and participate in further discussions with other technical people without being completely confused.

Some topics, like using the Linux command line, receive deep and extensive coverage because I believe that information to be vital to anyone who wants to be a power user or become a skilled DevOps guru. That topic gets two full chapters.

Other topics, like the chapter that mentions ADA and Fortran, along with more than 15 other programming languages, only get brief coverage so that people who are interested get a few guideposts to help them continue if they are interested. In this case, around 20 programming languages are covered in about a dozen pages. These are useful topics to some, but not topics I would consider vital.

Additionally, some topics are just too broad to be covered in great depth in this book, but are topics that deserve a mention because, again, an intermediate to advanced user should have at least a foundational knowledge of them. These are covered and then information is provided to help you find more resources and expand your understanding, as needed.

Those Wanting to Become Intermediate or Advanced Users

Ubuntu Unleashed is intended for intermediate and advanced users or those who want to become one. Our goal is to give you a nudge in the right direction, to help you enter the higher stages by exposing you to as many different tools and ideas as possible; we want to give you some thoughts and methods to consider and spur you on to seek out more.

Although the contents are aimed at intermediate to advanced users, new users who pay attention will benefit from the advice, tips, tricks, traps, and techniques presented in each chapter. Pointers to more detailed or related information are also provided at the end of each chapter.

If you are new to Linux, you might need to learn some new computer skills, such as how to research your computer's hardware, how to partition a hard drive, and (occasionally) how to use a command line. This book helps you learn these skills and shows you how to learn more about your computer, Linux, and the software included with Ubuntu. Most important, it helps you overcome your fear of the system by telling you more about what it is and how it works.

We would like to take a moment to introduce a concept called "The Three Levels of Listening" from Alistair Cockburn's *Agile Software Development*, published by Addison Wesley. These describe how a person learns and masters a technique. We all start at the first stage and progress from there. Few reach the last stage, but those who do are incredibly effective and efficient. People aiming for this stage are the very ones for whom we intend this book.

- **Following—**The stage where the learner looks for one very detailed process that works and sticks to it to accomplish a task.
- **Detaching—**The stage where the learner feels comfortable with one method and begins to learn other ways to accomplish the same task.
- **Fluent—**The stage where the learner has experience with or understanding of many methods and doesn't think of any of them in particular while doing a task.

Myriad books focus on the first set of users. This is not one of them. It is our goal in *Ubuntu Unleashed* to write just enough to be sufficient to get you from where you are to where you want or need to be. This is not a book for newcomers who want or need every step outlined in detail, although we do that occasionally. This is a book for people who want help learning about what can be done and a way to get started doing it. The Internet is an amazing reference tool, so this is not a comprehensive reference book. This book is a tool to help you see the landscape; to learn enough about what you seek to get you started in the right direction with a quality foundational understanding.

Sysadmins, Programmers, and DevOps

Systems administrators, or Sysadmins, are the people who keep servers and networks up and running. Their role is sometimes called *operations*. They deal with software installation and configuration, security, and do all the amazing things behind the scenes that let others use these systems for their work. They are often given less respect than they deserve, but the pay is good and it is a ton of fun to wield the ultimate power over a computer system. It is also a great responsibility, and these amazing guys and gals work hard to make sure they do their jobs well, striving for incredible system uptime and availability. Ubuntu is an excellent operating system for servers and networks, and in this book you can find much of the knowledge needed to get started in this role.

Programmers are the people who write software. They are sometimes called *developers*. Programmers work with others to create the applications that run on top of those systems. Ubuntu is a great platform for writing and testing software. This is true whether you are doing web application development or writing software for desktop or server systems. It also makes a great platform for learning new programming languages and trying out new ideas. This book can help you get started.

DevOps is a portmanteau of *developer* and *operations*. It signifies a blending of the two roles already described. The information technology (IT) world is changing, and roles are becoming less clear cut and isolated from one another. In the past, it was common to witness battles between programmers excited about new technology and sysadmins in love with stability. DevOps realizes that neither goal is healthy in isolation, but that seeking a balance between the two can yield great results by removing the barriers to communication and understanding that sometimes cause conflict within a team. Because of the rise of cloud computing and virtualization, which are also covered in this book, and more agile forms of development, DevOps is a useful perspective that enables people working in IT to do an even better job of serving their ultimate clients: end users. This book is a great foundation for those wanting to learn knowledge that will help with both roles, hopefully presented in a way that balances them nicely.

What This Book Contains

Ubuntu Unleashed is organized into six parts, described here. A disc containing the entire distribution is included so that you have everything you need to get started.

Part I, "Installation and Configuration" takes you through installing Ubuntu on your computer in the place of any other operating system you might be running, such as Windows.

Part II, "Desktop Ubuntu," is aimed at users who want to use Ubuntu on desktop systems.

Part III, "System Administration," covers both elementary and sophisticated details of setting up a system for specific tasks and maintaining that system.

Part IV, "Ubuntu as a Server," gives you the information you need to start building your own file, web, and other servers for use in your home or office.

Part V, "Programming Linux," provides a great introduction to how you can extend Ubuntu capabilities even further using the development tools supplied with it.

In addition to what has already been mentioned, after the spring release of Ubuntu, bonus chapters will be available online at www.informit.com/title/9780134511184 .

In addition, this book is part of InformIT's exciting Content Update Program, which provides content updates for major technology improvements! As significant updates are made to Ubuntu, sections of this book will be updated or new sections will be added to match the updates to the technologies. As updates become available, they will be delivered to you via a free Web Edition of this book, which can be accessed with any Internet connection. To learn more, visit informit.com/cup.

How to access the Web Edition: Follow the instructions inside to learn how to register your book to access the FREE Web Edition.

Conventions Used in This Book

It is impossible to cover every option of every command included in Ubuntu. Besides, with the rise of the Internet and high-speed connections, reference materials are far less valuable than they used to be because most of these details are only a quick Google search away. Instead, we focus on teaching you how to find information you need while giving a quality overview worthy of the intermediate or advanced user. Sometimes this book offers tables of various options, commands, and keystrokes to help condense, organize, and present information about a variety of subjects.

To help you better understand code listing examples and sample command lines, several formatting techniques are used to show input and ownership. For example, if the command or code listing example shows typed input, the input is formatted in boldface after the sample command prompt, as follows:

```
matthew@seymour:~$ ls
```

If typed input is required, as in response to a prompt, the sample typed input also is in boldface, like so:

```
Delete files? [Y/n] y
```

All statements, variables, and text that should appear on your display use the same boldface formatting. In addition, command lines that require root or super-user access are prefaced with the sudo command, as follows:

```
matthew@seymour:~$ sudo printtool &
```

Other formatting techniques include the use of italic for placeholders in computer command syntax. Computer terms or concepts are also italicized upon first introduction in text.

Finally, you should know that all text, sample code, and screenshots in *Ubuntu Unleashed* were developed using Ubuntu and open-source tools.

Read on to start learning about and using the latest version of Ubuntu.

CHAPTER 9

Managing Software

IN THIS CHAPTER

In this chapter, we look at the options you have to manage your software in Ubuntu. If you are used to an environment where you are reliant on visiting different vendor websites to download updates, you are in for a pleasant surprise. Updating a full Ubuntu installation, including all the application software, is as simple as running the Update Manager program. You will discover just how easy it is to install and even remove various software packages.

Ubuntu provides a variety of tools for system resource management. The following sections introduce the graphical software management tools that you will use for most of your software management. This chapter also covers monitoring and managing memory and disk storage on your system.

Ubuntu Software

Ubuntu Software is a graphical utility for package management in Ubuntu. You can find it in the Applications menu as Ubuntu Software; the package and executable program is named `ubuntu-software`. Ubuntu Software enables you to easily select and install a large array of applications by using the intuitive built-in search and easy one-click installation. When you open the program, you see the main screen, as shown in Figure 9.1.

Along the top side of the screen, you have three menu options: All, Installed, and Updates. Just below that is a search bar you can use to search for packages. Scroll down to find software listed by categories.

FIGURE 9.1 The initial Ubuntu Software screen enables you to browse through packages sorted by groups.

Installing new software via Ubuntu Software is as simple as finding it in the package list, double-clicking, and clicking the Install button. When you do so, you may be asked for your password; then the application is downloaded and installed. You can remove an application by finding it in Ubuntu Software and clicking the Remove button.

Using Synaptic for Software Management

Ubuntu Software works just fine for adding and removing applications, but if you need to install something specific—such as a library—you need to use Synaptic (Figure 9.2). You can install Synaptic using Ubuntu Software described earlier; it is not installed by default.

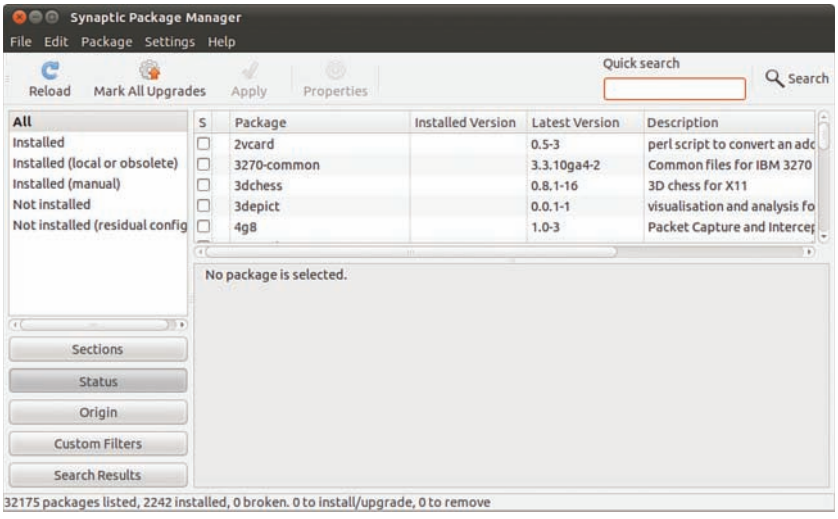

FIGURE 9.2 For more advanced software management in a GUI, Synaptic is the preferred tool.

Along the left are software categories (although this time there are more of them), along the top right are the package selections for that category, and on the bottom right is the Package Information window that shows information about the currently selected package. To install or remove software, click the check box to the left of its name, and you'll see a menu that offers the following options:

- **Unmark**—If you have marked this package for installation, upgrade, or one of the other options, this option removes that mark.
- **Mark for Installation**—Add this package to the list that will be installed.
- **Mark for Re-installation**—If you have some software already installed, but for some reason it's not working, this option reinstalls it from scratch. Existing configuration files are retained, so any edits you have made are safe.
- **Mark for Upgrade**—If the software has updates available, this option downloads and installs them.
- **Mark for Removal**—This option deletes the selected package from your system but leaves its configuration files intact so that if you ever reinstall it you do not have to reconfigure it.
- **Mark for Complete Removal**—This option deletes the selected package from your system but also removes any configuration files, purging everything from the system.

After you have made your changes, click the Apply button to have Synaptic download, install, upgrade, and uninstall as necessary. If you close the program without clicking Apply, your changes are lost.

Beneath the categories on the left side of the screen, you see six buttons: Sections, Status, Origin, Custom Filters, Search Results, and Architecture. These customize the left list: Sections is the Categories view; Status enables you to view packages that are installed or upgradable; Origin lists the different repositories available to download packages; Custom Filters has some esoteric groupings that are useful only to advanced users, Search Results stores results of your searches; and Architecture shows the packages specific to each architecture of Ubuntu.

You can press Ctrl+F at any time to search for a particular package. By default, it is set to search by package description and name. You may change the Look In box setting to only search for Name. As mentioned already, your search terms are saved under the Search view (the button on the bottom left), and you can click from that list to re-search on that term.

As well as providing the method of installing and removing software, Synaptic provides the means to configure the servers you want to use for finding packages. In fact, this is where you can make one of the most important changes to your Ubuntu system: You can open it up to the Ubuntu Universe and Multiverse.

Ubuntu is based on the Debian distribution, which has thousands software packages available for installation. Ubuntu uses only a subset of that number but makes it easy for you to install the others, along with many packages that are not available in Debian. When you use Synaptic, you see small orange Ubuntu logos next to many packages; this identifies them as being officially supported by the Canonical-supported Ubuntu developers. The packages that do not have this logo are supported by the wider Ubuntu community of developers.

To enable the Universe and Multiverse repositories, go to Settings, Repositories. This list shows all the servers you have configured for software installation and updates and includes the Universe and Multiverse repositories. When you find them, check them, and then click Close.

Synaptic shows a message box warning you that the repository listings have changed and that you need to click the Reload button (near the top left of the Synaptic window) to have it refresh the package lists. Go ahead and do that, and you should see a lot more software appear for your selection. However, notice that only a small number have the official Ubuntu “seal” attached, which means you may want to be a bit more careful when installing software.

NOTE

Much of the software discussed in this book is available only through the Universe repository. Therefore, we highly recommend enabling it to get full use out of this book and your Ubuntu installation.

Staying Up-to-Date

Although you can manage your software updates through Synaptic, Ubuntu provides a dedicated tool called Software Updater (shown in Figure 9.3). This tool is designed to be simple to use: When you run it, Software Updater automatically downloads the list of updates available and checks them all in the list it shows. If the update list was downloaded automatically not too long ago, you can force Ubuntu to refresh the list of available updates by clicking the Check button. Otherwise, all you need to do is click Install Updates to bring your system up to date. If you want a little more information about the updates, click Show Details at the bottom to see what has changed in the update.

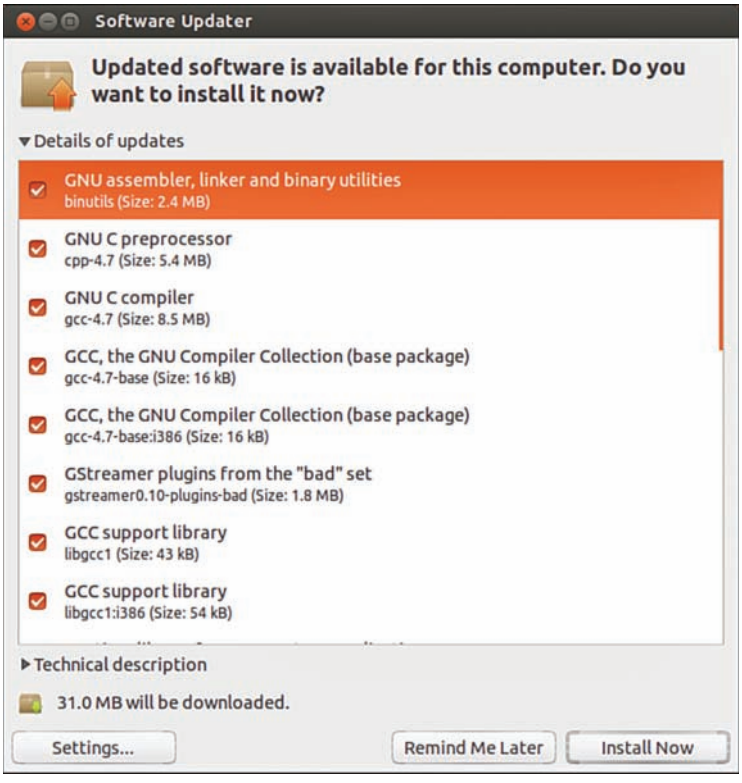

FIGURE 9.3 If you need to update your software to apply bug fixes and security upgrades, use Software Updater.

Ubuntu automatically checks for updates periodically and notifies you when critical updates are available. However, there's no harm running Software Updater yourself every so often, just to make sure; it's better to be safe than sorry.

Working on the Command Line

With so much software available for installation, it is no surprise that Debian-based distros have many ways to manage software installation. At their root, however, they all use Debian's world-renowned *Advanced Package Tool (APT)*. A person posting on Slashdot.com once said, "Welcome to Slashdot. If you can't think of anything original, just say how much APT rocks and you'll fit right in." You see, even though many other distros have tried to equal the power of APT, nothing else even comes close.

Why is APT so cool? Well, it was the first system to properly handle dependencies in software. Other distros, such as Red Hat, used RPM files that had dependencies. For example, an RPM for Gimp would have a dependency on Gtk, the graphical toolkit on which Gimp is based. As a result, if you tried to install your Gimp RPM without having the Gtk RPM, your install would fail. So, you grab the Gtk RPM and try again. Aha: Gtk has a dependency on three other things that you need to download. And those three other things have dependencies on 20 other things. And so on, and so on, usually until you can't find a working RPM for one of the dependencies, and you give up.

APT, on the other hand, was designed to automatically find and download dependencies for your packages. So, if you want to install Gimp, it downloads Gimp's package and any other software it needs to work. No more hunting around by hand, no more worrying about finding the right version, and certainly no more need to compile things by hand. APT also handles installation resuming, which means that if you lose your Internet connection part-way through an upgrade (or your battery runs out, or you have to quit, or whatever), APT picks up where it left off the next time you rerun it.

Day-to-Day Usage

To enable you to search for packages both quickly and thoroughly, APT uses a local cache of the available packages. Try running this command:

```
matthew@seymour:~$ sudo apt-get update
```

The `apt-get` update command instructs APT to contact all the servers it is configured to use and download the latest list of file updates. If your lists are outdated, it takes a minute or two for APT to download the updates. Otherwise, this command executes it in a couple of seconds.

After the latest package information has been downloaded, you are returned to the command line. You can now ask APT to automatically download any software that has been updated, using this command:

```
matthew@seymour:~$ sudo apt-get upgrade
```

If you have a lot of software installed on your machine, there is a greater chance of things being updated. APT scans your software and compares it to the latest package information from the servers and produces a report something like this:

```
mmatthew@seymour:~$ sudo apt-get upgrade
Reading package lists... Done
```

```
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  cabextract google-chrome-beta icedtea6-plugin language-pack-en
  language-pack-en-base language-pack-gnome-en language-pack-gnome-en-base
  libfreetype6 libfreetype6-dev libsmbclient libwbclient0 openjdk-6-jre
  openjdk-6-jre-headless openjdk-6-jre-lib samba-common samba-common-bin
  smbclient upstart winbind xserver-common xserver-xorg-core
21 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 84.8MB of archives.
After this operation, 623kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

Each part of that report tells you something important. Starting at the top, the line "the following packages will be upgraded" gives you the exact list of packages for which updates are available. If you're installing new software or removing software, you see lists titled "The following packages will be installed" and "The following packages will be removed." A summary at the end shows a total of 21 packages that APT will upgrade, with 0 new packages, 0 to remove, and 0 not upgraded. Because this is an upgrade rather than an installation of new software, all those new packages take up only 623KB of additional space. Although you have an 84.8MB download, the packages are overwriting existing files.

It's important to understand that a basic `apt-get` upgrade never removes software or adds new software. As a result, it is safe to use to keep your system fully patched because it should never break things. However, occasionally you will see the "0 not upgraded" status change, which means some things cannot be upgraded. This happens when some software must be installed or removed to satisfy the dependencies of the updated package, which, as previously mentioned, `apt-get` upgrade will never do.

In this situation, you need to use `apt-get dist-upgrade`, so named because it's designed to allow users to upgrade from one version of Debian/Ubuntu to a newer version—an upgrade that inevitably involves changing just about everything on the system, removing obsolete software, and installing the latest features. This is one of the most-loved features of Debian because it enables you to move from version to version without having to download and install new CDs. Keeping regular upgrades and distro upgrades separate is very useful for making sure that security updates and simple bug fixes don't change software configurations that you may be counting on, especially on a machine that needs to be consistently available and working, such as a server.

Whereas `apt-get upgrade` and `apt-get dist-upgrade` are there for upgrading packages, `apt-get install` is responsible for adding new software. For example, if you want to install the MySQL database server, you run this:

```
matthew@seymour:~$ sudo apt-get install mysql-server
```

Internally, APT queries "mysql-server" against its list of software and find that it matches the mysql-server-5.5 package. It then finds which dependencies it needs that you don't already have installed and gives you a report like this one:

```
matthew@seymour:~$ sudo apt-get install mysql-server
[sudo] password for matt:

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18
  libnet-daemon-perl libplrpc-perl libterm-readkey-perl mysql-client-5.5
  mysql-client-core-5.5 mysql-server-5.5 mysql-server-core-5.5
Suggested packages:
  libipc-sharedcache-perl tinyca mailx
The following NEW packages will be installed:
  libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18
  libnet-daemon-perl libplrpc-perl libterm-readkey-perl mysql-client-5.5
  mysql-client-core-5.5 mysql-server mysql-server-5.5 mysql-server-core-5.5
0 upgraded, 13 newly installed, 0 to remove and 0 not upgraded.
Need to get 26.8 MB of archives.
After this operation, 96.2 MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

This time, you can see that APT has picked up and selected all the dependencies required to install MySQL Server 5.5, but it has also listed one recommended package and two suggested packages that it has not selected for installation. The "recommended" package is just that: The person who made the MySQL package (or its dependencies) thinks it would be a smart idea for you to also have the mailx package. If you want to add it, press `N` to terminate `apt-get` and rerun it like this:

```
matthew@seymour:~$ sudo apt-get install mysql-server mailx
```

The "suggested" packages are merely a lower form of recommendation. They don't add any crucial features to the software you selected for install, but it's possible that you might need them for certain non-crucial (to the main piece of software being installed) features or tasks.

NOTE

APT maintains a package cache where it stores DEB files it has downloaded and installed. This usually lives in `/var/cache/apt/archives` and can sometimes take up many hundreds of megabytes on your computer. You can have APT clean out the package cache by running `apt-get clean`, which deletes all the cached DEB files. Alternatively, you can run `apt-get autoclean`, which deletes cached DEB files that are beyond a certain age, thereby keeping newer packages.

If you try running `apt-get` install with packages you already have installed, APT considers your command to be `apt-get update` and looks to see whether new versions are available for download.

The last day-to-day package operation is removing things you no longer want, which you do through the `apt-get remove` command, as follows:

```
matthew@seymour:~$ sudo apt-get remove firefox
```

Removing packages can be dangerous because APT also removes any software that relies on the packages you selected. For example, if you were to run `apt-get remove libgtk2.0-0` (the main graphical toolkit for Ubuntu), you would probably find that APT insists on removing more than a hundred other things. The moral of the story is this: When you remove software, read the APT report carefully before pressing `Y` to continue with the uninstall.

A straight `apt-get remove` leaves behind the configuration files of your program so that if you ever reinstall it you do not also need to reconfigure it. If you want to remove the configuration files as well as the program files, run this command instead:

```
matthew@seymour:~$ sudo apt-get remove –purge firefox
```

Or:

```
matthew@seymour:~$ sudo apt-get purge firefox
```

Either will perform a full uninstall.

NOTE

You can see a more extensive list of `apt-get` parameters by running `apt-get` without any parameters. The cryptic line at the bottom, "This APT has Super Cow Powers," is made even more cryptic if you run the command `apt-get moo`.

Finding Software

With so many packages available, it can be hard to find the exact thing you need using command-line APT. The general search tool is called `apt-cache` and is used like this:

```
matthew@seymour:~$ apt-cache search kde
```

Depending on which repositories you have enabled, that tool returns about a thousand packages. Many of those results will not even have KDE in the package name but will be matched because the description contains the word *KDE*.

You can filter through this information in several ways. First, you can instruct `apt-cache` to search only in the package names, not in their descriptions. You do this with the `-n` parameter, like this:

```
matthew@seymour:~$ apt-cache -n search kde
```

Now the search has gone down from more than 1,000 packages to a few hundred.

Another way to limit search results is to use some basic regular expressions, such as `^`, meaning "start," and `$`, meaning "end." For example, you might want to search for programs that are part of the main KDE suite and not libraries (usually named something like `libkde`), additional bits (such as `xmms-kde`), and things that are actually nothing to do with KDE yet still match our search (like `tkdesk`). Do this by searching for packages that have a name starting with `kde`, as follows:

```
matthew@seymour:~$ apt-cache -n search ^kde
```

Perhaps the easiest way to find packages is to combine `apt-cache` with `grep`, to search within search results. For example, if you want to find all games-related packages for KDE, you could run this search:

```
matthew@seymour:~$ apt-cache search games | grep kde
```

When you've found the package you want to install, run it through `apt-get install` as usual. If you first want a little more information about that package, you can use `apt-cache showpkg`, like this:

```
matthew@seymour:~$ apt-cache showpkg mysql-server-5.0
```

This shows information on "reverse depends" (which packages require, recommend, or suggest mysql-server-5.0), "dependencies" (which packages are required, recommended, or suggested to install mysql-server-5.0), and "provides" (which functions this package gives you). The "provides" list is quite powerful because it allows different packages to provide a given resource. For example, a MySQL database-based program requires MySQL to be installed, but isn't fussy whether you install MySQL 4.1 or MySQL 5.5. In this situation, the Debian packages for MySQL 4.1 and MySQL 5.0 both have "mysql-server-4.1" in the provides list, meaning that they offer the functionality provided by MySQL 4.1. Therefore, you can install either version to satisfy the MySQL-based application.

Using `apt` Instead of `apt-get`

There is a new, simplified interface to APT that removes the hyphen and the second part of the command. It also includes lovely updates like a progress bar. Although this is new, in testing for this edition of the book, it was stable and pleasant to use. Table 9.1 lists some of the new commands and what they replace. Both versions work, so no relearning is necessary if you do not want to switch.

TABLE 9.1 apt-get versus apt

apt-get Command	apt Command
apt-get install	apt install
apt-get remove	apt remove
apt-get update	apt update
apt-get upgrade	apt upgrade

apt-get dist-upgrade	apt full-upgrade
apt-get remove –purge	apt purge
apt-get autoremove	apt autoremove
apt-get purge	apt purge

Compiling Software from Source

Compiling applications from source is not that difficult. There are two ways to do this: You can use the source code available in the Ubuntu repositories, or you can use source code provided by upstream developers (most useful for those projects that are not available in the Ubuntu repositories). For either method, you need to install the `build-essential` package to ensure that you have the tools you need for compilation. You may also need to install `automake` and `checkinstall`, which are build tools.

Compiling from a Tarball

Most source code that is not in the Ubuntu repositories is available from the original writer or from a company's website as compressed source *tarballs*—that is, `tar` files that have been compressed using `gzip` or `bzip`. The compressed files typically uncompress into a directory containing several files. It is always a good idea to compile source code as a regular user to limit any damage that broken or malicious code might inflict, so create a directory named `source` in your home directory.

From wherever you downloaded the source tarball, uncompress it into the `~/source` directory using the `-C` option to `tar`:

```
matthew@seymour:~$ tar zxvf packagename.tgz -C ~/source
matthew@seymour:~$ tar zxvf packagename.tar.gz -C ~/source
matthew@seymour:~$ tar jxvf packagename.bz -C ~/source
matthew@seymour:~$ tar jxvf packagename.tar.bz2 -C ~/source
```

If you are not certain what file compression method was used, use the `file` command to figure it out:

```
matthew@seymour:~$ file packagename
```

Now, change directories to `~/source/packagename` and look for a file named `README`, `INSTALL`, or a similar name. Print out the file if necessary because it contains specific instructions on how to compile and install the software. Typically, the procedure to compile source code is as follows:

```
matthew@seymour:~/source/packagename$ ./configure
```

This runs a script to check whether all dependencies are met and the build environment is correct. If you are missing dependencies, the configure script normally tells you exactly which ones it needs. If you have the Universe and Multiverse repositories enabled in Synaptic, chances are you will find the missing software (usually libraries) in there.

When your configure script succeeds, run the following to compile the software:

```
matthew@seymour:~/source/packagename$ make
```

And finally, run the following:

```
matthew@seymour:~/source/packagename$ sudo make install
```

If the compile fails, check the error messages for the reason and run the following before you start again:

```
matthew@seymour:~/source/packagename$ make clean
```

You can also run the following to remove the software if you do not like it:

```
matthew@seymour:~/source/packagename$ sudo make uninstall
```

Compiling from Source from the Ubuntu Repositories

You might sometimes want to recompile a package, even though a binary package is available in the Ubuntu repositories. For example, a program might have been compiled into a binary with a specific feature disabled that you would like to use. Here is how you can do this. We will call the software package we want to compile *foo*.

First, get the source from the Ubuntu repositories:

```
matthew@seymour:~$ apt-get source foo
```

Install the build dependencies for the package:

```
matthew@seymour:~$ sudo apt-get build-dep foo
```

Change to the directory for the source code (may include the version number):

```
matthew@seymour:~$ cd foo-4.5.2
```

Make whatever changes you want to make to the package or to the compilation flags. You can do this using `./configure` and `make`, or sometimes by making manual changes to a configuration file. Each package has the potential to do this differently, so you need to see that program's documentation. Try looking for a README file in the source code to get started.

Next, create a new `debian/changelog` entry. After you enter this command, you need to enter a message that tells why a new version was made, perhaps something like *Matthew's flight of fancy with extra sauce*.

NOTE

Ubuntu package numbering follows a specific pattern. To help yourself later, you should stick to this pattern. Using the foo numbers shown here, a typical Ubuntu package that was inherited from Debian with no changes would then be 4.5.2-1. A package inherited

from Debian, but changed for Ubuntu would be 4.5.2-1ubuntu1 (and then ubuntu2 for a second version, and so on). A package that did not have a version in Debian but which was created for Ubuntu would be 4.5.2-0ubuntu1 (and ubuntu2 and so on).

```
matthew@seymour:~$ dch -i
```

Build the source package. This creates all the files necessary for uploading a package:

```
matthew@seymour:~$ debuild -S
```

Finally, you are left with a `foo-4.5.2-1ubuntu1custom.deb` package (using whatever version number or suffix you created earlier) that you can install, and later uninstall as well, using your package manager. In some instances, multiple DEB files might be created, in which case you would replace the individual package name in the example here with `*.deb`.

```
matthew@seymour:~$ sudo dpkg -Oi foo-4.5.2-1ubuntu1custom.deb
```

Configuration Management

This section provides a quick introduction to a couple tools that might be useful for those who want more control over system configuration management. For larger needs, see Chapter 36, "Managing Sets of Servers."

dotdee

If you run Linux-based systems, you will find a series of directories that end with a `.d` and that store configuration files. These are sometimes called .d or "dot dee" directories. If you look in `/etc/`, you find many (such as `apparmor.d` and `pam.d`). Opening these directories reveals a large number of configuration files and perhaps other directories containing even more. In Ubuntu or other Debian-based systems, it is a violation of etiquette (and Debian policy) for any software package to be allowed to directly change the configuration files of another package. This can be problematic if you want to use system configuration management software.

dotdee solves this problem by allowing you to take any flat file in your filesystem and replace it with a symlink pointing to a file that is generated from a .d-style directory. It saves the original file and then updates the generated file automatically and dynamically any time any file in the original .d directory is added, deleted, or modified. This way, the Debian policy and general etiquette standards are met, but configurations can be modified as needed by an external program.

dotdee works its magic using `inotify` to dynamically and instantly update the master file. The master file can be built three different ways: using flat files, which are concatenated; using diff/patch files, which are applied in a quiltlike manner; and using executables, which process `stdin` and dump to `stdout`. This flexibility should make any system administrator or developer guru happy.

Snappy Ubuntu Core

Snappy Ubuntu Core takes the absolute minimum of files and code necessary for a usable Ubuntu server image and adds to it a new means of managing software packages. The idea is similar to how smart phones like Android-based phones provide software. In this method, the software packages include everything they need to run on the operating system, effectively making it so that a package is isolated from the operating system more completely. This is designed to protect from the possibility of a package breaking other packages or an entire operating system installation. It is also intended to make updates easier and cleaner. With the idea of convergence, where Ubuntu is aiming to use the same set of software for traditional desktops, laptops, tablets, and phones, all these devices will share the core operating system and Unity interface, and packages that work on any one should also work on the others. This feature first appeared in Ubuntu 16.04 LTS.

Using Snaps

Software bundles packaged this way are called *snaps*. Snaps can be installed using Ubuntu Software or from the command line. On the command line, snaps have a new command. Use the following to interact with snaps.

To show a list of snap packages that are available to be installed:

```
matthew@seymour:~$ snap find
```

Because snaps are new, few packages are available today. However, this list is growing and is likely to become unwieldy at some point. Sure, you can use grep to search through the output to try to find a specific package in the list, but you can also use:

```
matthew@seymour:~$ snap find searchterm
```

To install a snap package:

```
matthew@seymour:~$ sudo snap install packagename
```

To show a list of snap packages that are currently installed:

```
matthew@seymour:~$ snap list
```

To update a snap package:

```
matthew@seymour:~$ sudo snap refresh packagename
```

To remove a snap package:

```
matthew@seymour:~$ sudo snap remove packagename
```

To display a list of changes, such as when snaps were installed, updated, or removed:

```
matthew@seymour:~$ snap changes
```

It is possible for you to create a snap package. See Chapter 39, "Opportunistic Development," to learn how. Learn more and keep up to date as Ubuntu Snappy Core continues develop by checking out https://developer.ubuntu.com/en/snappy/.

References

- **www.debian.org/doc/manuals/project-history/ch-detailed.en.html—**History of the Debian Linux package system.
- **www.nongnu.org/synaptic/—**Home of the Synaptic package manager.
- **www.ubuntu.com/usn—**The official list of Ubuntu security notices.

This page intentionally left blank

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z