Learn LINUX
IN A MONTH OF LUNCHES

STEVEN OVADIA

FOREWORD BY JIM WHITEHURST

MANNING

Learn Linux in a Month of Lunches

by Steven Ovadia

Sample Chapter 17

brief contents

Part 3

Home system admin on Linux

This third and final part of *Learn Linux in a Month of Lunches* shows you how to be a system administrator for your home Linux setup. This section is all about taking care of your system, just like SysAdmins do, but on a much smaller and more manageable level. You'll learn how to keep your system up-to-date, how to set up printing, and basic networking concepts. This section is your advanced Linux class. In this final section, you'll learn about:

- Package management
- Updating your operating system
- Linux security
- Connecting to other computers
- Printing (a surprisingly interesting chapter!)
- Collaboration using version control

17 An in-depth look at package management and maintenance

Welcome to part 3 of our journey! You just finished part 2, which was about learning to work with Linux as a day-to-day user. That's why we learned about commands and text editors and software. Those are things you need to do your personal work.

In part 3, we're going to explore the administrative aspects of running Linux. This section will help you with your own work, but will also help your system to run smoothly. And just to be clear, when I say administrative, I mean systems administration and not the people at work who boss you around. Although, in a sense, you will be learning to boss around your system. So maybe the term works on a few levels.

We talked about installing and removing software in chapter 7 with a focus on using the repositories of software that come with your Linux system. In this chapter, we're going to further explore installing and removing software, with a focus on three areas:

- *Installing software from outside of the repository.* The ability to install software from outside of the repository is useful for situations where the software you want isn't included with your distribution. You'll recall from the previous chapter that I mentioned a launcher called Synapse that isn't in the Ubuntu repositories. In this chapter, you're going to learn how to install it.
- *Package dependencies.* This section will help you to understand your Linux system and how it runs programs. You understand the mechanics of installing software, but now it's time to learn what goes on underneath that. Many software issues are caused by dependency problems, meaning the programs needed by other programs. This section will help you to understand those challenges.

- *Advanced commands to install and remove software.* Advanced commands will help you keep your system free of unnecessary dependencies that, at best, take up memory on your computer and, at worst, can interfere with other programs.

This chapter will also help prepare you for chapter 18, where you're going to update and upgrade your system, which also involves the package manager.

Let's get started installing Synapse, which is not in the Ubuntu 14.04 repositories.

17.1 Installing software from outside of the repositories

In chapter 7, you learned about the convenience of the software repository. All of the software you need is in one place, and is either a click or a command away. But what about when the software you want *isn't* in a repository?

This happens for many reasons. Sometimes a piece of software doesn't make it into a particular distribution release. Sometimes a piece of software isn't in the repositories for licensing reasons. For instance, Dropbox, the popular file syncing/sharing program, is commercial software that's not available in any repositories. What if you wanted Dropbox installed on your system?

There are usually three options for installing software that's outside of the main repository:

- Installing software via a package file.
- Adding another repository to your system and then installing the software via your package manager.
- Installing software from source code. There are lots of different ways to do this. The software often has instructions on how to install it, usually in the form of a README file that contains installation directions. Because this process is so idiosyncratic, I'm not going to explore it in this chapter.

The method you choose depends upon how the software is available. Most of the time, you don't have a choice, so you use whichever option is available. Let's start with package files.

17.1.1 Installing software with package files

Software package files are like the .exe files you use to install Windows software. Debian-based systems, like Ubuntu, use .deb files, whereas other systems use other formats. Fedora and OpenSUSE both use .rpm files. Installing software via this method is as simple as downloading a file and then opening it. As an example, the Brackets and Atom text editors, which we talked about in chapter 9, are both available via software package files, as is Dropbox. Let's install Dropbox (don't worry if you don't have a Dropbox account; you won't need one for this exercise):

1. Go to www.dropbox.com/install?os=lnx.
2. Click download for Ubuntu (.deb) 32-bit.
3. Save the file to your Downloads folder.

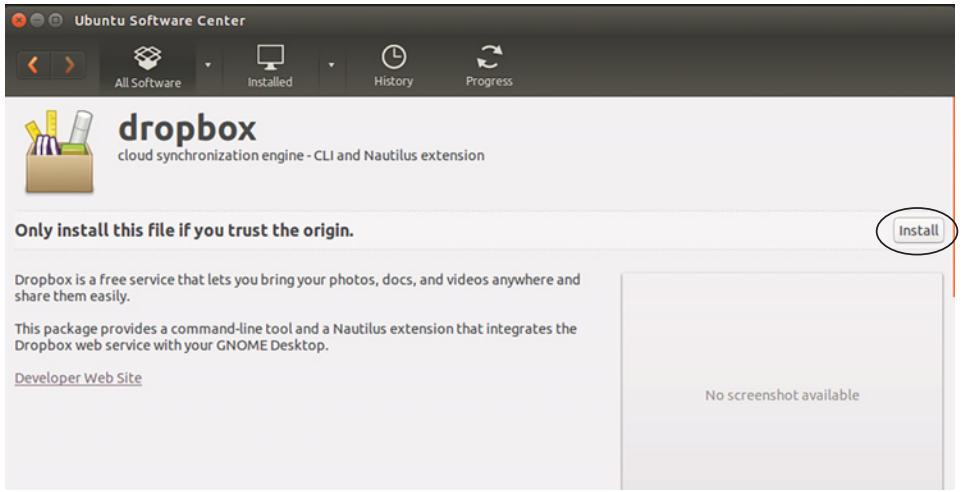

Figure 17.1 Ubuntu will install .deb files with Ubuntu Software Center.

4. Double-click the .deb file and Ubuntu will install Dropbox using the Ubuntu Software Center (see figure 17.1) after you enter in your password.
5. You'll see a green check once Dropbox is installed. You can now delete the .deb file from your Downloads folder.

INSTALLING .DEB FILES If you don't like using the Ubuntu Software Center to install .debs, you can install GDebi, which is a tool for installing .deb files. I prefer GDebi to Ubuntu Software Center, which always feels kind of slow to me. Once you have GDebi, right-click on the .deb, click Open With… and choose GDebi to use it as the default for opening .deb files.

This is a fairly painless way to install software on your system. However, always take care when installing software packages. You could potentially be installing something harmful on your system. Research the origins of the software package file. If it's coming from a business you've heard of, like Google or Dropbox, then it should be fine. But if it's a standalone file on a site or message board you're not familiar with, then you should probably avoid using it—at least until you have confirmation, based on researching what others have said about the package, that the file won't harm your system.

Now that you know how to use software package files, let's explore how to install a repository.

17.1.2 Viewing and adding repositories

When you download software out of a repository, it feels like you're pulling software from one big collection. But the reality is that you're working with multiple repositories and interacting with them through a single interface, like Synaptic. When you add

a repository to your system, it's like you're connecting a hose to your package manager and the hose is bringing in new packages from that repository you just added (see figure 17.2).

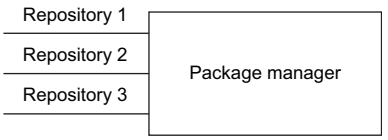

Figure 17.2 The package manager connects to multiple repositories.

Before we add a repository to our system, let's make sure the software we want to add isn't in our current repositories. Don't worry! Nothing bad will happen if you try to install software that's not in the repositories. You'll just get an error message.

First, let's look for Synapse. Go into Synaptic and look for a package called `synapse`. It won't come up, though. It's another launcher but unlike Kupfer and GNOME Do, it's not in the Ubuntu repositories. If Kupfer and GNOME Do didn't quite resonate with you, Synapse could turn out to be your preferred launcher. Software is a personal choice, and before I choose a tool, I like to take a few for a test spin.

TO SEE REPOSITORIES ON YOUR SYSTEM

Before we add the Synapse repository, let's look at our current repositories and see which ones are included.

1. Go into Dash and launch Software & Updates. This is a tool to manage your repositories. We used it in chapter 4 to look at our drivers. If you change a repository here, it's changed across your system—even if you use a different package manager. The opening screen shows you the four repositories that make up our system's software list: main, universe, restricted, and multiverse (see figure 17.3). We talked about this in chapter 7, but to review:
   - *Main* is free and open source software maintained by the Ubuntu developers.

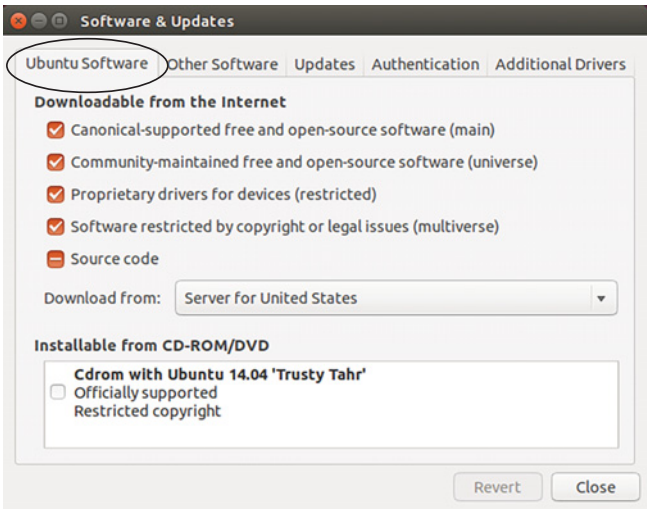

Figure 17.3 Ubuntu's software list is made up of four repositories.

- *Universe* refers to software contributed by the community but not officially supported by Ubuntu.
- *Multiverse* refers to non-free, proprietary software (there is no way to purchase software through Synaptic so it doesn't refer to cost).
- *Restricted* refers to proprietary drivers.

Now we're ready to add the Synapse repository.

To add a repository using graphically

Adding a repository is a two-step process:

1. Click the Other Software tab. Anything with a check in the box is software in your repository, but that isn't an official part of the Ubuntu project. For instance, you'll see the Dropbox repositories in there now that we installed it (see figure 17.4). Anything without a check in the box is not in your repository, but can be added with a check.

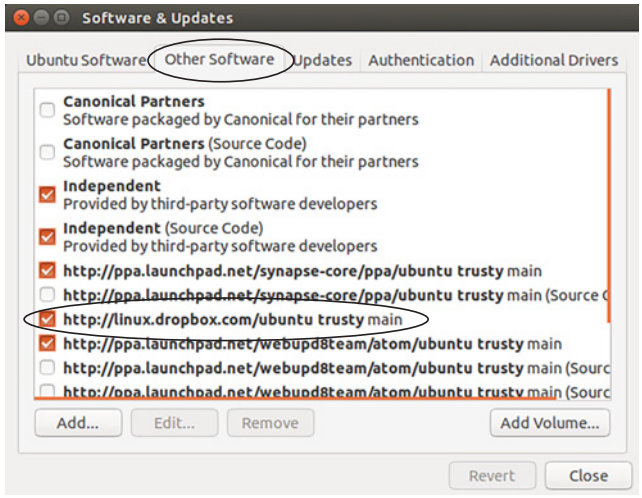

Figure 17.4 Other software shows you repositories from beyond the main Ubuntu project.

2. Click the Add button. A pop-up will open asking you to Enter the Complete APT Line of the Repository That You Want to Add as Source. You're going to type in `ppa:synapse-core/ppa`. The PPA information comes from the project page on Launchpad.net, which is where Canonical hosts different software projects.

Ubuntu will use that information to pull in the web address of the repository for you:

1. Click Close. Ubuntu will now prompt you to reload its software because it's out-of-date.
2. Click the Reload button. Ubuntu will now re-index the repositories. Software & Updates will close when it's done.

3. Return to Synaptic.
4. Click Reload to refresh Synaptic's holdings.
   - Search for synapse again.
   - The package is now available for you.
   - Install it just like we installed Vim in chapter 7.
   - Once Synaptic finishes, close out of it.
   - Go into Dash and launch Synapse. It's another great launcher to play with if you didn't get enough of launchers in the last chapter!

When you typed in the repository information, you used the initials PPA. It's an abbreviation for *Personal Package Archive*, which is a repository maintained by a project or individual, rather than the distribution. PPAs are specific to Debian-based systems but other distributions have similar concepts. Returning to the hose analogy at the start of this section, a PPA is the hose that brings packages into your system. Most distributions have methods of adding repositories that are conceptually similar to PPAs. For instance, Arch Linux has *AUR*, which stands for *Arch User Repository*. This is where Arch users can place software not in the main Arch repository and share it with the Arch community. OpenSUSE has a comparable concept it calls the *Open Build Service*.

TO ADD A REPOSITORY USING COMMANDS

As you might expect, there are also commands to add a new repository (see figure 17.5):

- `sudo add-apt-repository ppa:synapse-core/ppa`
- `sudo apt-get update`
- `sudo apt-get install synapse`

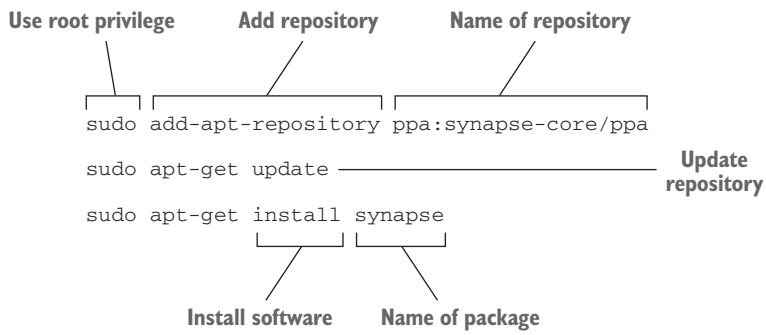

Figure 17.5 As you might expect, there are also commands to add a repository and install new software.

Now that you know the different ways to install software, we're ready to move into dependencies, which are the programs needed by other programs to run. In the next section, we're going to look at what dependencies are, how they work, and how you can remove unneeded dependencies to save space on your system.

17.2 Dependencies

I've talked about package dependencies at different times in this book. Dependencies are files and libraries (modular software shared and used by multiple programs) needed by a program to run it. Part of the job of the package manager is to take care of the dependencies for you. It looks at what files a package needs to run, making sure you have them on your system and installing them if you don't. The package manager even makes sure you have the correct version of a program or library, since sometimes a piece of software needs a specific version of a file or library.

When you just installed Synapse, you didn't only install Synapse. You also installed the programs and libraries it needs to run. What are they? Let's take a look:

1. Open Synaptic and search for Synapse.
2. Right-click on it and click Properties. This area will tell you all about a package.
3. Click the Dependencies tab (see figure 17.6) to see other programs Synapse needs to run. The number in parenthesis means the version of the package or library needed by Synapse. For instance, the program libgee2 needs to be version greater than or equal to 0.5.0. This is useful if you're having trouble with a program. Sometimes the package manager is using the incorrect version of a dependency. Looking at the dependencies yourself gives you the ability to see what the issue might be—whether it's a missing dependency or the incorrect version of one.
4. Close out of properties and close Synapse.

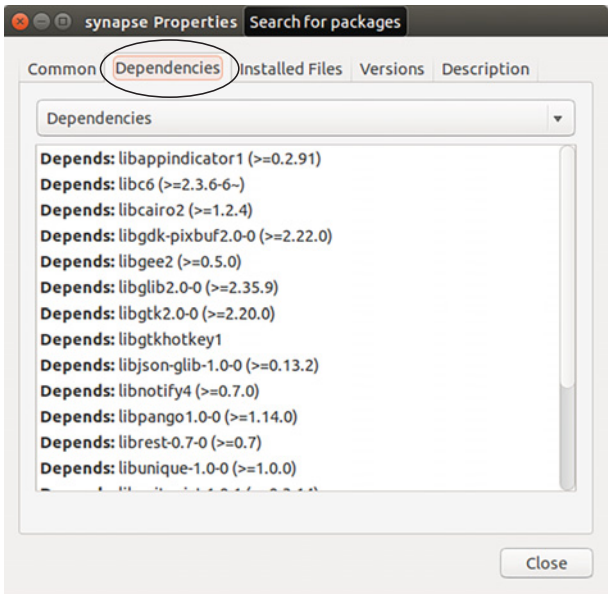

Figure 17.6 The Synaptic package manager allows you to see dependencies.

Of course, if you were to remove Synapse from your system, you would no longer need some of these dependencies but they would remain on your system, taking up space. In the next section, you'll see how to remove them.

17.2.1 Using advanced commands to remove dependencies

Before you can remove dependencies, you have to remove the program. If you try to remove a dependency, the package manager will also remove software using the dependency. This won't break your system, but it's annoying if a program you use gets removed because then you have to reinstall it.

First, let's remove Synapse using the command line. In chapter 12, we used the command `sudo apt-get remove` to uninstall software on our system. That command removes software but leaves behind configuration files. A more complete way to remove almost everything about a program is to use `sudo apt-get --purge remove` and the name of the package. This is the command I usually use to remove software in order to keep my system as clean as possible. Go into the terminal and remove Synapse with `sudo apt-get --purge remove synapse`.

If you look at your terminal text, you'll see a message at the top that you might not have noticed previously when removing software:

```
The following packages were automatically installed and are no longer
     required:
  consolekit libck-connector0 libgtkhotkey1 libpam-ck-connector pastebinit
  python-gnome2 python-pyorbit
Use 'apt-get autoremove' to remove them.
```

Those packages were used by Synapse and aren't used by anything else on your system, so you don't need them. All they're doing is taking up space. If you install another piece of software that needs one of those programs, the package manager will reinstall what it needs. Go back to the terminal and type `sudo apt-get autoremove` to get rid of those unneeded dependencies.

AUTOREMOVE `autoremove` can be a controversial command in that it occasionally removes packages needed by other packages, thus breaking your system. I have never run into this issue personally, but I have read about other users running into it. The only time I would skip autoremove is when working with:

- *A metapackage*—A metapackage is a preconfigured collection of programs that allows you to easily install something complex, like a desktop environment, without having to install lots of packages one-by-one.
  The packages bundled in a metapackage aren't necessarily dependencies. Instead, they make software more usable to the user by pre-selecting everything needed to make it work well. If I removed a metapackage, like `xfce` or `kde`, which are both desktop environments, I wouldn't follow-up with `autoremove`. Instead, I would leave whatever packages were left behind on my system.

- *GNOME applications*—The GNOME desktop environment has a lot of software tightly integrated with it, like the Evolution email client and GNOME Videos, the video player. These programs are tightly integrated into GNOME and it seems that when you remove those programs and then their dependencies, you often wind up breaking GNOME. When dealing with GNOME applications and the GNOME desktop, I leave the dependencies alone. You can see a list of GNOME applications on the GNOME wiki: https://wiki.gnome.org/

Other than those two situations, I use `autoremove` constantly and have never run into an issue with it personally.

You'll remember we looked at Synapse's dependencies and saw a program called `libgee2`. It wasn't on the `autoremove` list, so that means at least one other program on our system uses it. Let's see what happens when we try to remove `libgee2`. Go into the terminal and type `sudo apt-get --purge remove libgee2`.

You'll see a *long* message beginning with:

```
The following packages were automatically installed and are no longer
    required:
```

These are all the programs that also use `libgee2`. If we remove `libgee2`, the package manager will remove *all* of the programs that use it. Handling dependencies this way is useful behavior since it prevents us from removing programs needed by other programs. Rather than leave us with broken software, the package manager offers to remove everything that would be broken were the dependency removed.

It's also why you need to be careful removing programs, because if you blindly said yes to removing both programs, you might wonder what happened to that huge list of software shown in the terminal message. That's why it's important to always carefully read system messages. Type `N` and press Enter to cancel removing `libgee2`.

17.3 Wrapping up

You now know different ways to install software when the software you want isn't in a repository. This will open up your Linux world even more. You also should have a sense of how dependencies work and how to remove them from your system when they're no longer needed, which helps to save space. All of this work will come in handy in the next chapter when we talk about updating and upgrading your system, which also relies on your package manager.

Glossary of terms

In this chapter I explained:

Arch User Repository (AUR)—A repository containing packages submitted by projects or individuals, rather than the Arch distribution.

Dependency—Files and libraries (modular software shared and used by multiple programs) needed by a program to run it.

Personal Package Archive (PPA)—A repository maintained by a project or individual, rather than the distribution. PPAs are specific to Debian-based systems.

Software package files—The Linux version of the .exe files you use to install Windows software. Debian-based systems, like Ubuntu, use .deb files while other systems use other formats. For instance, Fedora and OpenSUSE use .rpm files.

17.4 Lab

This chapter was a lesson about installing and deleting software from your computer:

1. Use only commands to install the Atom text editor. Its PPA is ppa:webupd8team/atom and details can be found on the project page at https://launchpad.net/~webupd8team/+archive/ubuntu/atom.
2. What dependencies does Atom have?
3. Remove Kupfer and its dependencies. If you're using a live session, you'll have to install it again, which we did in the previous chapter.

LINUX

Learn LINUX IN A MONTH OF LUNCHES

STEVEN OVADIA
FOREWORD BY JIM WHITEHURST

If you've only used Windows or Mac OS X, you may be daunted by the Linux operating system. And yet learning Linux doesn't have to be hard, and the payoff is great. Linux is secure, flexible, and free. It's less susceptible to malicious attacks, and when it is attacked, patches are available quickly. If you don't like the way it looks or behaves, you can change it. And best of all, Linux allows users access to different desktop interfaces and loads of software, almost all of it completely free.

Learn Linux in a Month of Lunches shows you how to install and use Linux for all the things you do with your OS, like connecting to a network, installing software, and securing your system. Whether you're just curious about Linux or need it for your job, you'll appreciate how this book focuses on just the tasks you need to learn. In easy-to-follow lessons designed to take an hour or less, you'll learn how to use the command line, along with practical topics like installing software, customizing your desktop, printing, and even basic networking. You'll find a road map to the commands and processes you need to be instantly productive.

WHAT'S INSIDE

- Master the command line
- Learn about file systems
- Understand desktop environments
- Go from Linux novice to expert in just one month

This book is for anyone looking to learn how to use Linux. No previous Linux experience required.

Steven Ovadia is a professor and librarian at LaGuardia Community College, CUNY. He curates The Linux Setup, a large collection of interviews with desktop Linux users, and writes for assorted library science journals.

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit www.manning.com/books/learn-linux-in-a-month-of-lunches

"Guides readers through Linux basics in a clear and systematic way."
—From the Foreword by Jim Whitehurst, Red Hat

"An essential reference for beginners."
—Shawn Bolan
New Horizons

"Relevant to the tasks you will find yourself doing on a daily basis."
—Robert Walsh
Excalibur Solutions

"Great beginner's guide to the world of Linux, with plenty of good examples."
—Selcuk Beydilli, OTPP

MANNING $39.99 / Can $45.99 [INCLUDING eBOOK]

ISBN-13: 978-1-617-29328-3
ISBN-10: 1-617-29328-8
53999
9 781617 293283