LEARNING MADE EASY

10th Edition

# Linux®

for dummies®

A Wiley Brand

Install Linux and start using open source software

Browse the web, use email, edit photos, and listen to music

Learn to use Ubuntu and openSUSE distributions

**Richard Blum**

Online Linux instructor

# Linux®

for dummies®

A Wiley Brand

# Linux®

10th Edition

**Richard Blum**

for dummies®
A Wiley Brand

## Linux® For Dummies®, 10th Edition

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com



Published simultaneously in Canada



Library of Congress Control Number: 2020943580

ISBN 978-1-119-70425-6 (pbk); ISBN 978-1-119-70426-3 (ebk); ISBN 978-1-119-70427-0 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

# Contents at a Glance

# Table of Contents

# Introduction

Welcome to the fascinating world of open source software that is Linux. In this book, I introduce you to the wonders of the Linux operating system, originally created as a labor of love by Linus Torvalds in the early 1990s. My goal is to initiate you into the rapidly growing ranks of Linux users and enthusiasts busily rewriting the rules for the operating system marketplace.

If you've contemplated switching to Linux but find the prospect too forbidding, you can relax. If you can boil water or set your alarm clock, you too can become a Linux user. (No kidding!)

When this book appeared in its first edition, Linux was an emerging phenomenon that was neither terribly well known nor understood. In this edition — for a new generation of Linux users — so much material is available that I've steered this particular title toward what Linux is and how you can make the best use of it on your desktop. To that end, these pages contain various online resources, tips, and tricks, as well as more general instruction. If you're looking for material on servers, *Linux All-in-One Desk Reference For Dummies* by Emmett Dulaney (John Wiley & Sons, Inc.) can serve your needs.

In this book, I keep the amount of technobabble to a minimum and stick with plain English as much as possible. Besides plain talk about Linux installation, boot up, configuration, and software, I include many examples, plus lots of detailed instructions to help you set up and use your very own Linux machine with a minimum of stress or confusion.

## About This Book

Think of this book as a friendly, approachable guide to tackling terminology and the Linux collection of tools, utilities, and widgets. Although Linux isn't terribly hard to figure out, it does pack a boatload of details, parameters, and *administrivia* (administrative trivia, in UNIX-speak). You need to wrestle those details into

shape while you install, configure, manage, and troubleshoot a Linux-based computer. Some sample topics you find in this book include the following:

- Understanding where Linux comes from and what it can do for you
- Installing the Linux operating system
- Working with a Linux system to manage files and add software
- Setting up Internet access and surfing the web
- Customizing your Linux desktop
- Managing Linux system security and resources

Although it may seem, at first glance, that working with Linux requires years of hands-on experience, tons of trial and error, advanced computer science training, and intense dedication, take heart! It's not true! If you can tell somebody how to find your office, you can certainly build a Linux system that does what you want. The purpose of this book isn't to turn you into a full-blown Linux geek (that's the ultimate state of Linux enlightenment, of course); it's to show you the ins and outs that you need to master to build a smoothly functioning Linux system and to give you the know-how and confidence to use it.

# How to Use This Book

This book tells you how to install, configure, and customize a Linux desktop system. Although you can do most things in Linux these days by pointing and clicking, you still may want to try using Linux at the command prompt — where you type detailed instructions to load or configure software, access files, and do other tasks. In this book, input appears in monospace type like this:

```
rmdir /etc/bin/devone
```

When you type Linux commands or other related information, be sure to copy the information exactly as you see it in the book, including uppercase and lowercase letters, because that's part of the magic that makes Linux behave properly.

A failure to follow instructions exactly can have all kinds of unfortunate, unseemly, or unexpected side effects.

The margins of a book don't give you the same amount of room as your computer screen; therefore, in this book, some URLs and lengthy commands at the command prompt may appear wrapped to the next line. Remember that your computer sees these wrapped lines as a *single set of instructions*, or as a single URL — so

if you're typing a hunk of text, keep it on a single line. Don't insert a hard return if you see one of these wrapped lines. I clue you in that it's supposed to be all one line by breaking the line at a slash mark or a natural word break (to imply "Wait — there's more!") and slightly indenting the overage, as in the following silly example:

```
www.infocadabra.transylvania.com/nexus/plexus/lexus/ praxis/
  okay/this/is/a/make-believe/URL/but/some/real/ones/ are/
  SERIOUSLY/long.html
```

Note that as you dig your way into and through this book — and other sources of Linux wit, wisdom, and inspiration that you're likely to encounter — you may find some terms used interchangeably. For example, you may see the same piece of software called a *program*, a *command*, a *utility*, a *script*, an *application*, or a *tool*, depending on the source, the context, and the source of the information you're consulting. To a large extent, you can treat these terms as interchangeable, and when an important distinction needs to be made among them, I point it out. Similarly, when you're working with various commands or configuration controls, you may also encounter terms such as *flag*, *switch*, *option*, or *parameter* used more or less interchangeably. In this case, all these terms refer to ways in which you can control, refine, or modify basic commands or programs to make them do what you want. Again, wherever distinctions and clarifications may be needed, I provide them.

# Three Presumptuous Assumptions

They say that making assumptions makes a fool of the person who makes them and of the person about whom those assumptions are made. (And just who are *they*, anyway? I *assume* that I know, but — never mind.) Even so, practicality demands that I make a few assumptions about you, gentle reader:

- You can turn your computer on and off.
- You know how to use a mouse and a keyboard.
- You want to install, configure, and/or use a desktop Linux system because you're curious or interested or it's your job to do so.

You don't need to be a master logician or a wizard in the arcane art of programming to use this book, nor do you need a PhD in computer science. You don't even need a complete or perfect understanding of what's going on in your computer's innards.

If you have an active imagination and the ability to solve rudimentary problems, that's even better — you have already mastered the key ingredients necessary to making Linux work for you. The rest are mere details and a bit of patience. I can help you with the details, but the patience is up to you!

# Icons Used in This Book

Within each chapter, icons highlight particularly important or useful information. You find the following icons in this book:


TIP

The Tip icon flags useful information that makes living with your Linux system even less complicated than you feared that it might be.


REMEMBER

I sometimes use this icon to point out information you just shouldn't pass by — don't overlook these gentle reminders. (The life, sanity, or page you save may be your own.)


WARNING

Be cautious when you see this icon — it warns you of things you shouldn't do. This icon is meant to emphasize that the consequences of ignoring these bits of wisdom can be severe.


TECHNICAL STUFF

This icon signals technical details that are informative and interesting but not critical to understanding and using Linux. Skip these paragraphs if you want (but please come back and read them later).

# Where to Go from Here

This is where you pick a direction and hit the road! *Linux For Dummies*, 10th Edition, is much like *1001 Nights* because it almost doesn't matter where you start out. You look at lots of different scenes and stories as you prepare yourself to build your own Linux system. Although each story has its own distinctive characters and plot, the whole is surely something to marvel at. Don't worry — you can handle it. Who cares whether anybody else thinks that you're just goofing around? I know that you're getting ready to have the time of your life.

And don't forget to check out the cheat sheet. Just go to `dummies.com` and search for "*Linux For Dummies cheat sheet.*"

Enjoy!

# 1

# Getting Your Feet Wet

**IN THIS PART. . .**

Find out what Linux is, where it came from, and how it works.

Tackle the various tasks and activities involved in preparing for and installing Linux on a PC.

Boot up your new Linux environment.

Find your way around the desktop and tweak it to your liking.

IN THIS CHAPTER

- Diving into Linux origins and features
- Telling Linux apart from the rest of the operating system pack
- Depending on GNU and the GPL
- Marveling at the Linux Company (or lack thereof)
- Checking out popular Linux distributions

Chapter 1

# Getting Acquainted with Linux

Welcome to the world of Linux, the operating system developed by thousands of people around the world! In this chapter, you find out about Linux itself — what it is, where it comes from, and why it gets so much attention. Prepare to have your assumptions challenged about how software *must* be developed and sold, and your mind opened to new possibilities.

## Is Free Really Free?

Understanding Linux requires a radical shift of thought regarding the way that you acquire and use computer software. (***Note:*** By *radical,* I mean getting to the root of the matter, rather than putting on beads and camping out in the administration building.) Your first step toward shifting your mindset is to alter your general connotation of the word *free* to represent *freedom,* rather than *free lunch.* That's right; you can sell "free" software for a fee . . . and you're encouraged to do so, as long as you relay the same freedom to each recipient of the software.

Don't scratch your head too hard; these concepts are tough to grasp initially, especially when you consider the conditioning you've received from the commercial software industry's marketing departments. Perhaps you don't know that when you purchase most proprietary software packages, you don't actually *own* the software. Rather, you're granted permission to use the software within the bounds dictated by the licensor.

Linux also has a license. However, the motives and purpose of the license are much different from those of most commercial software. Instead of using a license to restrict use of the software, the GNU General Public License (GPL) that Linux uses ensures that the software will always be open to anyone. No company can ever own Linux or dictate the way in which you use or modify Linux — although they can have their own individual copyrights and trademarks on their various brands of it, such as Red Hat and SUSE. In essence, you already own Linux, and you can use it for anything you like, as long as you propagate the GPL freedoms to any further recipients of the software.

# Linux: Revolution or Just Another Operating System?

Before going any farther into Linux, I need to get some terminology out of the way.

*Tux* is the formal name of the mascot penguin that represents Linux. Rumor has it that Linux's creator, Linus Torvalds, is rather fond of these well-dressed inhabitants of the Antarctic.

An *operating system* is the software that runs your computer, handling all interactions between you and the hardware. Whether you're writing a letter, calculating a budget, or managing your recipes on your computer, the operating system provides the essential air that your computer breathes. Furthermore, an operating system isn't just one program; it consists of hundreds of smaller programs and utilities that allow us humans to use a computer to do something useful. You then run other programs (such as your word processor) on top of the operating system to get everything done.

In recent technological history, Linux has evolved from a techie playground to a rock-solid solution for the business enterprise. The same software that was once dismissed as rogue is now being adopted and promoted by industry leaders such as IBM, Hewlett-Packard, Motorola, Microsoft, and Intel. Each of these computer manufacturers has determined that Linux provides value for their customers in some way (as well as for their own operations).

Linux has been accused of being "just another operating system." On the surface, it may appear so, but if you look deeper, you can see that this isn't so. The Linux project is a flagship leading the current trend toward open source and free (as in freedom, not free lunch) software within the computing industry. A rock-solid operating system because of the model under which it was (and continues to be) developed, Linux represents much that is good in software development.

Two fundamental distinctions separate Linux from the rest of the operating-system pack:

- Linux is licensed under the unique and ingenious *GNU General Public License,* which you can read about in the next section.
- Linux is developed and maintained by a worldwide team of volunteer and paid programmers, working together over the Internet.

Linux is great for many reasons, including the fact that the folks who built it from the ground up wanted it to be

- **Multiuser:** More than one user can be logged in to a single computer at one time.
- **Multiprocess:** True *preemptive multitasking* enables the operating system core to efficiently juggle several programs running at once. This is important for providing multiple services on one computer.
- **Multiplatform:** While Mac OS only runs on Intel CPUs and Windows only runs on Intel and ARM CPUs, Linux currently runs on more than 24 different CPU *platforms* (hardware types), including 32- and 64-bit Intel-based PCs, Digital/Compaq Alpha, all variants of the Apple Macintosh, Sun SPARC, the Apple iPod, ARM CPUs, and even the Microsoft XBox.
- **Interoperable:** Linux plays nice with most network protocols (languages) and operating systems, allowing you to interact with users and computers running Microsoft Windows, UNIX, Apple Macintosh computers, and other, more niche groups.
- **Scalable:** As your computing needs grow, you can rely on Linux to grow with you. The same Linux operating system can run on a tiny electronic photo frame, a desktop computer, or a very large, industrial-strength server system.
- **Portable:** Linux is mostly written in the C programming language. *C* is a language created specifically for writing operating system–level software and can be readily *ported* (translated) to run on new computer hardware.
- **Flexible:** You can configure the Linux operating system as a network host, router, graphical workstation, office productivity PC, home entertainment computer, file server, web server, cluster, or just about any other computing appliance that you can think of.

- **Stable:** The Linux *kernel* (the core of the operating system) has achieved a level of maturity that makes most software developers envious. It's not uncommon to hear reports of Linux servers running for years without crashing.
- **Efficient:** The modular design of Linux enables you to include only the components needed to run your desired services. Even older computers can utilize Linux and become useful again.
- ***Free:*** To most people, the most intriguing aspect of Linux is the fact that it's often available free of charge. How (the capitalists murmur) can anyone build a better mousetrap with no incentive of direct monetary return?

In this chapter, I intend to answer that last question for you. I also hope to paint a picture of the open source software development model that created Linux.

## SO WHERE DID LINUX COME FROM?

The quickest way to understand Linux is to take a peek at its rich heritage. Although programming of the Linux core started in 1991, the design concepts were based on the time-tested *UNIX* operating system.

UNIX was developed at Bell Telephone Laboratories in the late 1960s. The original architects of UNIX, working back when there were few operating systems, wanted to create one that shared data, programs, and resources both efficiently and securely — something that wasn't available then (and is still sought after now). From there, UNIX evolved into many different versions; its current family tree is so complicated that it looks like a kudzu infestation!

In 1991, Linus Torvalds was a computer science student at the University of Helsinki in Finland. He wanted an operating system that was like the UNIX system that he'd grown fond of at the university, but both UNIX and the hardware it ran on were prohibitively expensive. A UNIX version called Minix was available for free, but it didn't quite meet his needs. So, as a computer science student, Torvalds studied Minix and then set out to write a new version himself. In his own words (recorded for posterity on the Internet because this was in an early version of an online chat room), his work was "just a hobby, won't be big and professional like GNU."

Writing an operating system is no small task. Even after six months of hard work, Torvalds had made very little progress toward the general utility of the system. He posted what he had to the Internet — and found that many people shared his interest and curiosity. Before long, some of the brightest minds around the world were contributing to Linus's project by adding enhancements or fixing bugs (errors in the code).

# Anatomy of an Open Source Software Project

To the casual observer (and some corporate IT decision makers), Linux appears to be a freak mutation — a rogue creature randomly generated by anarchy. How, after all, can something so complex and discipline dependent as a computer operating system be developed by a loosely knit band of volunteer computer geeks from around the world?

Just as science is constantly attempting to classify and explain everything in existence, technology commentators are still trying to understand how the open source approach can create superior software, especially in cases where there is no charge. Often the reasons have much to do with the usual human desire to fill a need with a solution. When a programmer in the Linux world wants a tool, the programmer simply writes one — or bands together with other people who want a similar package, and they write it together.

## GNU who?

Imagine — software created out of need rather than projected profit. Even though UNIX ultimately became expensive proprietary software, the ideas and motives for its creation were originally based on practical needs. What people usually refer to (in the singular) as the *Linux operating system* is actually a collection of software tools that were created with the express purpose of solving specific computing problems.

The speed of Linux's popularity also wouldn't be possible without the vision of a man whom Steven Levy (author of the book *Hackers*) refers to as "The Last of the Great MIT AI-LAB Hackers" — in the original sense of the word *hacker* is someone who is an expert at coding, not the current popular meaning that implies criminal intent. This pioneer and advocate of *freedom* software is Richard Stallman.

The Massachusetts Institute of Technology (MIT) has long held a reputation for nurturing the greatest minds in the technological disciplines. In 1984, Stallman, a gifted student and brilliant programmer at MIT, was faced with a dilemma — sell his talent to a company for a tidy sum of money or donate his gifts to the world. He did what we'd all do . . . right?

Stallman set out on a journey to create a completely free operating system that he would donate to the world. He understands — and continues to live — the original hacker ethic, which declares that information wants to be free. This concept wasn't

new in his time. In the early days of the computing industry, many advancements were made by freely sharing ideas and programming code. Manufacturer-sponsored user groups brought the best minds together to solve complicated problems. This ethic, Stallman felt, was lost when companies began to hoard software as their own intellectual property with the single purpose of profit.

As you may or may not have gathered by this point, widespread and accessible source code is paramount to successful software development. *Source code* is the term for the human-readable text (as opposed to the unreadable cyber-hieroglyphics in an "executable" file) that a programmer types to communicate instructions to the computer.

Writing computer programs using code that computers can run directly is an extremely arduous task. Modern computer software is usually written in a human-friendly language and then *compiled,* or translated, into the computer's native instruction set. To make changes to this software, a programmer needs access to a program's source code. Most proprietary software comes only as a precompiled product; the software developer keeps the source code for those programs under lock and key.

After determining that his operating system would be built around the conceptual framework of UNIX, Stallman wanted the project name to distinguish his system from UNIX. So, he chose the recursive acronym *GNU* (pronounced ga-*new*), which means GNUs not Unix.

To finance the GNU project, Stallman organized the Free Software Foundation (FSF), which sold open source software to help feed the programmers who worked on its continuing development. (Remember, we're talking *free* as in *freedom,* not *free lunch*.) Although this organization and goal of creating a complete operating system was necessary and important, a much more important piece of the puzzle had to be put into place to protect this new software from big-business pirates — a concern still all too relevant today as a former Linux company tries to hijack ownership of decades of volunteer work from thousands of people around the world.

The *GNU General Public License* (GPL) is a unique and creative software license that uses copyright law to protect the freedom of the software user, which is usually the opposite of how a copyright works. Generally, a copyright is an enforceable designation of ownership and restriction from duplication by anyone but the copyright holder. When software is licensed under the GPL, recipients are bound by copyright law to respect the freedom of anyone else to use the software in any way they choose. Software licensed with the GPL is also known as copy*left* software (the reverse of right, get it?). Another way to remember the GPL is through its ultimate result: Guaranteed Public for Life.

While Stallman's work set the stage for Linux's rapid climb to popularity, the operating system he and his crew were working on took longer than expected. If you're interested in the completed version, go to `www.gnu.org/software/hurd/hurd.html`.

## Who's in charge of Linux, anyway?

As an open source project evolves, various people emerge as leaders. This leader is often known as the project's *benevolent dictator.* A person who becomes benevolent dictator has probably spent more time than anyone else on a particular problem and often has some unique insight. Normally, the words *democratic* and *dictator* are never paired in the same sentence, but the open source model is a very democratic process that endorses the reign of a benevolent dictator.

Linus Torvalds is still considered the benevolent dictator of the Linux *kernel* (the operating system's core). He ultimately determines what features are added to the kernel and what features aren't. The community trusts his vision and discretion. In the event that he loses interest in the project, or the community decides that he has gone senile, a new leader will emerge from amongst the very competent people working with him.

## Einstein was a volunteer

Someone who is a volunteer or donates time to a project isn't necessarily providing a second-rate effort (or only working on weekends and holidays). In fact, any human resources expert will tell you that people who choose to do a job of their own free will produce the highest quality products.

The volunteers who contribute to open source projects are often leaders in their fields who depend on community collaboration to get useful work done. The open source concept is no stranger to the scientific community. The impartial peer-review process that open source projects foster is critical in validating some new feature or capability as being technically correct.


REMEMBER

Those who paint the open source community as copyright violators and thieves often misunderstand — or outright ignore — these vital issues. Open source programmers are very proud of their work *and* are also very concerned about their own copyrights, not wanting their work to be stolen by others — hence licenses such as the GPL. This concern creates an atmosphere with the greatest respect for copyright. Bandits who claim that they're "just being open source" when they steal other people's hard work are grossly misusing the term to soothe their own consciences.

Many have also pointed out that if copyright is violated in open source, it's easy to tell. Watch the news and notice how often large software corporations are convicted of stealing other people's code and incorporating it into their own work. If the final product is open source, it's easy for anyone to look and make sure nothing stolen is in it. As you might imagine, tracking down such copyright violations is much more difficult in a closed source scheme.

# Packaging Linux: The Distribution

A complete Linux system package is called a *distribution*. A Linux distribution contains the Linux kernel, the GNU project's tools, and any number of open source software projects to provide various functionality for the system.

Lots of different Linux distributions are available to meet just about any computing requirement you could have. Most distributions are customized for a specific user group, such as business users, multimedia enthusiasts, software developers, or normal home users.

Each customized distribution includes the software packages required to support specialized functions, such as audio and video editing software for multimedia enthusiasts, or compilers and integrated development environments (IDEs) for software developers.

The different Linux distributions are often divided into three categories:

- Full core Linux distributions
- Specialized distributions
- Live test distributions

The following sections describe these different types of Linux distributions, and shows some examples of Linux distributions in each category.

## Core Linux distributions

A core Linux distribution contains the Linux and GNU operating system, one or more graphical desktop environments, and just about every Linux application that is available, ready to install and run. The core Linux distribution provides one-stop shopping for a complete Linux installation, no matter what your requirements are!

Table 1-1 shows some of the more popular core Linux distributions out there.