INTRODUCTION

I want to tell you a story. No, not the story of how, in 1991, Linus Torvalds wrote the first version of the Linux kernel. You can read that story in lots of Linux books. Nor am I going to tell you the story of how, some years earlier, Richard Stallman began the GNU Project to create a free Unix-like operating system. That's an important story too, but most other Linux books have that one, as well. No, I want to tell you the story of how you can take back control of your computer.

When I began working with computers as a college student in the late 1970s, there was a revolution going on. The invention of the microprocessor had made it possible for ordinary people like you and me to actually own a computer. It's hard for many people today to imagine what the world was like when only big business and big government ran all the computers. Let's just say you couldn't get much done.

Today, the world is very different. Computers are everywhere, from tiny wristwatches to giant data centers to everything in between. In addition to

ubiquitous computers, we also have a ubiquitous network connecting them together. This has created a wondrous new age of personal empowerment and creative freedom, but over the last couple of decades something else has been happening. A single giant corporation has been imposing its control over most of the world's computers and deciding what you can and cannot do with them. Fortunately, people from all over the world are doing something about it. They are fighting to maintain control of their computers by writing their own software. They are building Linux.

Many people speak of "freedom" with regard to Linux, but I don't think most people know what this freedom really means. Freedom is the power to decide what your computer does, and the only way to have this freedom is to know what your computer is doing. Freedom is a computer that is without secrets, one where everything can be known if you care enough to find out.

Why Use the Command Line?

Have you ever noticed in the movies when the "super hacker"—you know, the guy who can break into the ultra-secure military computer in under 30 seconds—sits down at the computer, he never touches a mouse? It's because movie makers realize that we, as human beings, instinctively know the only way to really get anything done on a computer is by typing on a keyboard.

Most computer users today are familiar with only the *graphical user interface (GUI)* and have been taught by vendors and pundits that the *command line interface (CLI)* is a terrifying thing of the past. This is unfortunate, because a good command line interface is a marvelously expressive way of communicating with a computer in much the same way the written word is for human beings. It's been said that "graphical user interfaces make easy tasks easy, while command line interfaces make difficult tasks possible," and this is still very true today.

Since Linux is modeled after the Unix family of operating systems, it shares the same rich heritage of command line tools as Unix. Unix came into prominence during the early 1980s (although it was first developed a decade earlier), before the widespread adoption of the graphical user interface and, as a result, developed an extensive command line interface instead. In fact, one of the strongest reasons early adopters of Linux chose it over, say, Windows NT was the powerful command line interface, which made the "difficult tasks possible."

What This Book Is About

This book is a broad overview of "living" on the Linux command line. Unlike some books that concentrate on just a single program, such as the shell program, `bash`, this book will try to convey how to get along with the command line interface in a larger sense. How does it all work? What can it do? What's the best way to use it?

This is not a book about Linux system administration. While any serious discussion of the command line will invariably lead to system administration topics, this book touches on only a few administration issues. It will, however, prepare the reader for additional study by providing a solid foundation in the use of the command line, an essential tool for any serious system administration task.

This book is very Linux-centric. Many other books try to broaden their appeal by including other platforms, such as generic Unix and Mac OS X. In doing so, they "water down" their content to feature only general topics. This book, on the other hand, covers only contemporary Linux distributions. Ninety-five percent of the content is useful for users of other Unix-like systems, but this book is highly targeted at the modern Linux command line user.

Who Should Read This Book

This book is for new Linux users who have migrated from other platforms. Most likely you are a "power user" of some version of Microsoft Windows. Perhaps your boss has told you to administer a Linux server, or maybe you're just a desktop user who is tired of all the security problems and want to give Linux a try. That's fine. All are welcome here.

That being said, there is no shortcut to Linux enlightenment. Learning the command line is challenging and takes real effort. It's not that it's so hard, but rather it's so *vast*. The average Linux system has literally *thousands* of programs you can employ on the command line. Consider yourself warned: Learning the command line is not a casual endeavor.

On the other hand, learning the Linux command line is extremely rewarding. If you think you're a "power user" now, just wait. You don't know what real power is—yet. And, unlike many other computer skills, knowledge of the command line is long lasting. The skills learned today will still be useful 10 years from now. The command line has survived the test of time.

It is also assumed that you have no programming experience—not to worry. We'll start you down that path as well.

What's in This Book

This material is presented in a carefully chosen sequence, much as though a tutor were sitting next to you, guiding you along. Many authors treat this material in a "systematic" fashion, which makes sense from a writer's perspective but can be very confusing to new users.

Another goal is to acquaint you with the Unix way of thinking, which is different from the Windows way of thinking. Along the way, we'll go on a few side trips to help you understand why certain things work the way they do and how they got that way. Linux is not just a piece of software; it's also a small part of the larger Unix culture, which has its own language and history. I might throw in a rant or two, as well.

This book is divided into four parts, each covering some aspect of the command line experience:

- **Part 1: Learning the Shell** starts our exploration of the basic language of the command line, including such things as the structure of commands, filesystem navigation, command line editing, and finding help and documentation for commands.
- **Part 2: Configuration and the Environment** covers editing configuration files that control the computer's operation from the command line.
- **Part 3: Common Tasks and Essential Tools** explores many of the ordinary tasks that are commonly performed from the command line. Unix-like operating systems, such as Linux, contain many "classic" command-line programs that are used to perform powerful operations on data.
- **Part 4: Writing Shell Scripts** introduces shell programming, an admittedly rudimentary, but easy to learn, technique for automating many common computing tasks. By learning shell programming, you will become familiar with concepts that can be applied to many other programming languages.

How to Read This Book

Start at the beginning of the book and follow it to the end. It isn't written as a reference work; it's really more like a story with a beginning, a middle, and an end.

Prerequisites

To use this book, all you will need is a working Linux installation. You can get this in one of two ways:

- **Install Linux on a (not so new) computer.** It doesn't matter which distribution you choose, though most people today start out with Ubuntu, Fedora, or OpenSUSE. If in doubt, try Ubuntu first. Installing a modern Linux distribution can be ridiculously easy or ridiculously difficult, depending on your hardware. I suggest a desktop computer that is a couple of years old and has at least 256MB of RAM and 6GB of free hard disk space. Avoid laptops and wireless networks if at all possible, as these are often more difficult to get working.
- **Use a live CD.** One of the cool things you can do with many Linux distributions is run them directly from a CD-ROM without installing them at all. Just go into your BIOS setup, set your computer to "Boot from CDROM," insert the live CD, and reboot. Using a live CD is a great way

to test a computer for Linux compatibility prior to installation. The disadvantage of using a live CD is that it may be very slow compared to having Linux installed on your hard drive. Both Ubuntu and Fedora (among others) have live CD versions.

Note: *Regardless of how you install Linux, you will need to have occasional superuser (i.e., administrative) privileges to carry out the lessons in this book.*

After you have a working installation, start reading and follow along with your own computer. Most of the material in this book is "hands on," so sit down and get typing!

WHY I DON'T CALL IT "GNU/LINUX"

In some quarters, it's politically correct to call the Linux operating system the "GNU/Linux operating system." The problem with "Linux" is that there is no completely correct way to name it because it was written by many different people in a vast, distributed development effort. Technically speaking, Linux is the name of the operating system's kernel, nothing more. The kernel is very important, of course, since it makes the operating system go, but it's not enough to form a complete operating system.

Enter Richard Stallman, the genius-philosopher who founded the Free Software movement, started the Free Software Foundation, formed the GNU Project, wrote the first version of the GNU C Compiler (GCC), created the GNU General Public License (the GPL), etc., etc. He *insists* that you call it "GNU/Linux" to properly reflect the contributions of the GNU Project. While the GNU Project predates the Linux kernel and the project's contributions are extremely deserving of recognition, placing them in the name is unfair to everyone else who made significant contributions. Besides, I think "Linux/GNU" would be more technically accurate since the kernel boots first and everything else runs on top of it.

In popular usage, "Linux" refers to the kernel and all the other free and open source software found in the typical Linux distribution—that is, the entire Linux ecosystem, not just the GNU components. The operating system marketplace seems to prefer one-word names such as DOS, Windows, Solaris, Irix, AIX. I have chosen to use the popular format. If, however, you prefer to use "GNU/Linux" instead, please perform a mental search and replace while reading this book. I won't mind.