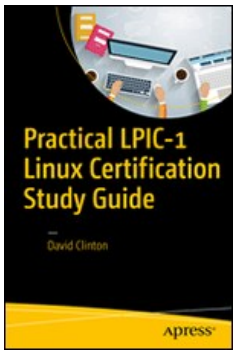

Practical LPIC-1 Linux Certification Study Guide

by David Clinton

Introduction

Overview

First of all, welcome.

Whether you're reading this book because you've decided to earn the Linux Professional Institute's Server Professional Certification or because you simply want to learn more about Linux administration, you've made a great choice. Right now, for a thousand reasons, Linux administration skills are opening doors to some of the hottest job markets on earth. And with the ongoing explosive growth of the cloud computing world—the vast majority of it being built with Linux—the opportunities will only get richer.

Now, about this book. I chose to have the chapters closely follow the LPIC exam topics. Not only will this make it much easier for you to study for each of the two exams required for the LPIC-1 certification, but I believe that the exam objectives are actually nicely aligned with the tools you'll need in the real world. Whether or not you end up taking the exam, if you manage to learn this material, you'll have done yourself a real favor.

By far the most important element of your success, however, will have very little to do with this or any other book. No matter how much time you spend studying a book, very little of the information you read will magically translate into knowledge and skills, unless you put it to work.

If you want to really "get" this stuff, you'll have to roll up your sleeves, open up a terminal, and *do it*. As soon as you finish a chapter or a section, try out what you've learned on a real living, breathing Linux system. Even better, take on your own projects. Be ambitious. Be adventurous. Take (managed) risks.

To this end, I include suggestions for practical exercises at the end of each chapter (right before the Test Yourself quizzes). Be prepared to spend longer than you expected on some of those tasks, sometimes longer than it took you to read the chapters they're based on. Also, accept that you will probably make some mistakes that will require even more time to fix. This is all as it should be. Remember: you learn more from experience than anything else.

You will notice that I used the words "complete" and "quick" to describe this book. Let me explain what I meant. The book is complete in the sense that every concept, principle, process, and resource that might make an appearance on the exam is fully represented (even a few that are now quite obsolete and/or useless: I'm looking at you, X Font server).

However, your journey through this book may also be relatively quick, since I've tried to be as selective as possible about what I included. As you will see soon enough, I didn't even try to include every single option for every single utility, which would have been highly impractical. But it would also have been largely useless, because I don't believe any normal human being could possibly absorb page after page after page of that kind of dry, abstract information.

If you want to see the full, formal documentation for a particular Linux utility, simply consult the man pages that came preinstalled with your Linux distribution. As an example, from the command line, you can type:

```
man cp
```

Besides including only the more common command options, I also tried to avoid discussing more general IT issues that don't relate directly to the LPIC exam. It's not that they're not important, but I figured that they may only interest a relatively small number of my readers and, importantly, they're all easily accessible on the Internet. I'd like to introduce you to one of my best friends: the Internet search engine.

So if you're curious about something that isn't discussed in these pages or if a project you're working on needs greater detail, then by all means, dive in deep. But because I know that the Internet has answers to just about any question you're likely to have, I'm able to focus this book more narrowly on the curriculum that interests everyone.

Having said that, please visit our web site, bootstrap-it.com. We'll try to make your visit worthwhile and, more importantly, provide you all with the opportunity to talk to us—and to each other. Let us know how you're doing and what you think.

About Linux

There's so much I could say about Linux:

- It's the operating system used by more than 95% of the world's supercomputers.
- Google, Netflix, and Facebook? Linux, Linux, and Linux.
- The vast majority of virtual machines fired up on the leading cloud computing platforms (like Amazon's AWS) are running Linux, and that includes Microsoft's Azure!
- There's a very good chance that the software powering your car, television, smartphone, air traffic control system, and even neighborhood traffic lights is one flavor or another of Linux.

If there's innovation in the worlds of science, finance, communications, entertainment, and connectivity, it's almost certainly being driven by Linux. And if there are dozens of attractive, virus-free, secure, and reliable desktop and mobile operating systems freely available to fill all kinds of roles, those too are driven by Linux.

Note By they way, you may be interested to know that this book was produced in its entirety on Linux, using only open source software. The whole thing: research, testing, and image processing.

The Linux Foundation recently (September 2015) estimated that, over just the past few years, collaborative projects under their umbrella have produced an estimated $5 billion in economic value. This was, again according to the Foundation, "work that would take 1,356 developers more than 30 years to replicate."

But where did all this innovation, productivity, and value come from? Who actually makes it all happen? It seems that the little operating system built a couple of decades ago by Linus Torvalds and then donated to the world, is maintained by an army of thousands of developers. According to the Linux Foundation, through 2015, 7.71 changes were accepted into the Linux kernel each HOUR and those contributions were the work of, besides Torvalds himself, more than 4,000 developers scattered around the world, many of whom, it must be noted, are sponsored by the companies they work for.

That's the *power* of open source. "Open source?" I hear you ask. "But who will support us when things go wrong?"

That's the *beauty* of open source. Because when I can't figure out how to do something or when I discover a bug in some open source software, I can usually quickly find the answer through an Internet search or, if not, there are knowledgeable and helpful folks online just waiting to help me. Try it out. You might, as I have from time to time, quickly find yourself in direct contact with the project developers themselves.

Some years ago, I wrote a white paper arguing the business case for transitioning small and medium-sized businesses from proprietary office productivity software suites (Microsoft Office) to open source alternatives (LibreOffice). When I compared the response/resolution times delivered by Microsoft with the average times seen on volunteer-staffed online OpenOffice and LibreOffice help forums, the latter would consistently produce a quicker turnaround.

Now it's your turn. All that innovation is going to need administrators to apply it to the real world. After all, we system administrators know just how little developers would get done without us. As the IT world grows and changes, you will be on the cutting edge.

Or will you? Let me tell you a story about an old friend of mine who, 25 years ago, had a great job as a Unix admin. As he tells it, the problem was that Unix (which, for the purpose of this discussion, is effectively synonymous with Linux) was getting so good at automating processes and system audits that all kinds of midlevel admins simply became unnecessary. My friend lost his job.

Could this happen to you? Absolutely. Unless, that is, you make an effort to keep up with technology as it evolves. There will be new areas to keep your eyes on (embedded tech, container virtualization, and others not yet imagined). It's the 21st century: you're never finished learning.

Nevertheless, I predict that 95% of the basic Linux skills you will learn here will probably still be in use ten and even 20 years from now. This is solid, foundational material.

About the LPIC-1 Exams

The two exams you'll need to pass to earn your Server Professional Certification (LPIC-1 101 and 102) are also known as CompTIA Linux+ LX0-103 and LX0-104. Until a few years ago, CompTIA offered a Linux certification that was so similar to the LPIC that the two eventually merged. All you have to know is that, whatever they're called, they work the same way and will get you to the same place.

That is not true of LPI's Linux Essentials (LPI-010) exam, which is a single, introductory exam that's meant for individuals with far less experience and knowledge than a candidate for the Server Professional would have. Besides those, the LPI offers two other sets of exams designed to demonstrate added skills and experience beyond those of the LPIC-1: the LPIC-2 (Linux Network Professional Certification) and LPIC-3 (Mixed Environments, Security, or Virtualization and High Availability).

This book is based on the April 2015 edition of the exams (Version 4.0). The people who maintain the certification and exams are, by design, very conservative in the way they adopt major changes, so you can be confident that the key exam topics won't be changing dramatically any time soon. Still, you should make sure that the training material on which you're relying does match the current version of the exam.

The Linux Professional Institute is vendor neutral, meaning that no one mainstream Linux distribution or software stack is favored over any other. You will therefore need to become familiar with a range of technologies. So, for example, expect to see both the Systemd and Upstart process managers, or both the apt and yum package managers. And that's a really good thing, because all of those systems are widely used (for now, at least) and all have unique valuable features. You can only gain from understanding how they all work. Success with the LPIC-1 will also automatically earn you the SUSE CLA certification.

Each exam is made up of 60 multiple choice and fill in the blank questions which must be completed within 90 minutes. To pass an exam, you will need to score 500 marks out of a total of 800. Since the questions are weighted by topic, there is no guarantee that one question will be worth the same number of marks as another. You can book an exam through the web site of either the Pearson VUE or Prometric test administration companies.

As with most technical certification exams, you will need to present the exam provider with two forms of identification, one of them a government-issued photo ID. You will also be expected to surrender any electronic devices or notebooks. (If you're very nice to the proctors, they might give them back to you once you're done.)

More than most certifications, the LPI has done a great job communicating exactly what you will need to know. You should spend some time carefully reading through the two exam objectives pages from their web site (lpi.org/study-resources/lpic-1-101-exam-objectives and lpi.org/study-resources/lpic-1-102-exam-objectives) before you begin this study and then go through them again at the end of the process to make sure you haven't missed anything. For your convenience, I've included the objectives in an appendix at the end of this book.

You will notice that each topic is given a weight between one and five. Those indicate the relative importance of a topic in terms of how large a role it will play in the exam. Table 1 is a simple chart that adds up the weights by topic to illustrate the importance of each.

Table 1: Topics and Their Weighting

Topic	Weight	
101	8	System Architecture
102	11	Linux Installation and Package Management
103	26	GNU and Unix Commands
104	15	Devices, Filesystems, Filesystem Hierarchy Standard
Total:	**60**	
105	10	Shells, Scripting and Data Management
106	4	User Interfaces and Desktops
107	12	Administrative Tasks
108	11	Essential System Services
109	14	Networking Fundamentals
110	9	Security
Total:	**60**	

Exam Tips

Try to arrive at the exam center as relaxed and well rested as possible. Carefully and slowly read each question and each possible response. Look for important details and for details that are only there to distract you. If you're not absolutely sure which answer is correct, try to narrow down the field a bit by eliminating answers that are obviously incorrect. You can always skip hard questions and return to them later when you've completed the rest.

Finally, remember that more people fail this exam on their first try than pass: it's designed to inspire your best effort. So don't give up.

Linux Survival Skills

Why only a single section—isn't this whole book about Linux survival skills? Well yes, but how are you going to survive between now and the time you finish reading it? Just to get you started, it might be useful to pick up a few super-critical, can't-live-without-me tools.

First, nearly everything in Linux administration will happen through the terminal. But I know that at least some of you are sitting in front of a shiny new Linux GUI interface right now and wondering where the #$%@! the terminal is (if you'll excuse my language). The answer is: that depends. Ubuntu, for instance, changes their menu design with just about every distribution, so exactly where terminal will appear on your desktop is hard to predict. In some ways, things just got more complicated with some more recent desktop manager versions, which got rid of menus altogether.

If you're not interested in poking around looking for it, you can try hitting the Alt+f2 combination and then typing terminal (or gnome-terminal) into the dialog box. Or, on some systems, Ctrl+Alt+t will get you there directly.

Once you're in the terminal, try running a command. Type:

```
pwd
```

which stands for present work directory. This is the folder (something that's almost always called a directory in Linuxland, by the way) you're currently in. You can list the files and subdirectories in your current directory with ls:

```
ls -l
```

Adding the -l argument gives you a longer, more detailed list displaying file attributes. If it's already installed (and it usually will be), you can use the nano text editor to, in this case, create and edit a new text file:

```
nano myfile.txt
```

Go ahead and type a few words and then hit Ctrl+x to save and exit. You can now quickly view your literary creation using cat:

```
cat myfile.txt
```

Try that again, but this time, type only cat my without the rest of the file name. Instead, hit the Tab key and Linux Command Completion should figure out what you're after and finish the command for you. Just hit Enter to accept the suggestion. Trust me: this one can save you a great

many keystrokes and a whole lot of time over the coming years.

Let's create a new directory:

```
mkdir newplace
```

and change directory into newplace and then run pwd once again:

```
cd newplace
pwd
```

Perhaps you'd like to copy the file you just created into this directory. To do this, you'll need to keep in mind where the personal "home" directory exists in the larger Linux filesystem. Let's assume that the account is called bootstrap-it, which is therefore the name of the home directory:

```
cp /home/bootstrap-it/myfile.txt .
```

The /home/bootstrap-it/myfile.txt section identifies the file you want to copy, and the dot (.) tells the cp command to copy it to the current directory. Run ls to confirm that a copy has arrived:

```
ls
```

You can change a file's name or move it using mv:

```
mv myfile.txt mynewfile.txt
ls
```

And you can permanently delete the file using rm:

```
rm mynewfile.txt
```

To help you experiment with Linux skills without having to worry about making a mess of your important stuff, you might try working with disposable systems. One way to do that is by loading a Linux image on to a USB stick, and then booting your computer to a live Linux session. Unless you mount and play around with your existing hard drive, nothing you do will have any permanent impact on your "real" data or system settings, and nothing you do to the live filesystem will survive a reboot. This has the added potential advantage of exposing you to a wide range of Linux distributions beyond the one that you've chosen for your main work.

Of course, installing the VirtualBox package on your system will let you load virtual operating systems of nearly any flavor within your desktop environment to get a good taste of how things work in other Linux distributions.

LXC Containers

You can also create virtual machines within a working installation using LXC. An LXC container (as its called) is a fully functioning, persistent virtual "machine" that likes to imagine that it lives all by itself on your hardware (see Figure 1). You can play around in this sandbox-like environment to your heart's content and, when you break something (as you probably will), you can just destroy it and start again with a new one. I highly recommend using LXCs for exploration and experimentation. I use them myself all the time and they've saved me untold hours of heartache.

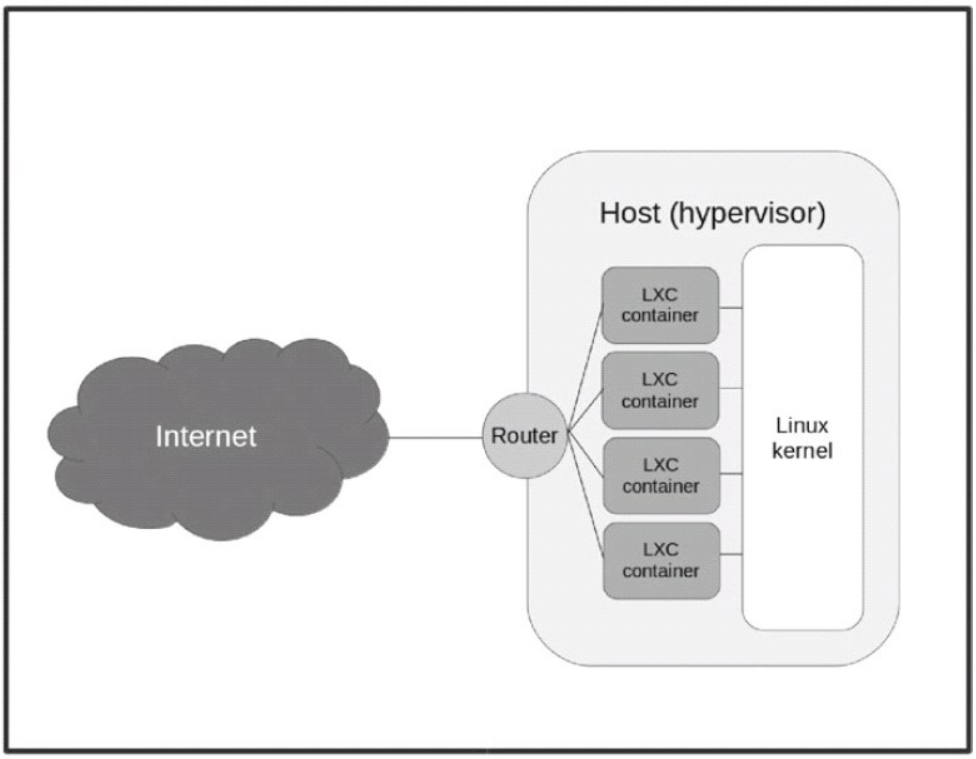

Figure 1: LXC container architectural design

Here are the simple steps you'll need to get started with LXC (none of this is included among the LPIC-1 exam expectations). This assumes that you're using an Ubuntu machine; some commands may be a bit different for other distributions. First, make sure that openssh is installed on your host machine (I'll talk a lot more about what that is later in the book):

```
sudo apt-get update
sudo apt-get install openssh-server
```

Now install lxc:

```
sudo apt-get install lxc
```

Then create a new container called newcon using the ubuntu template:

```
sudo lxc-create -t ubuntu -n newcon
```

Once that's done (and it should only take a minute or two), boot the new container:

```
sudo lxc-start -d -n newcon
```

The -d tells lxc to detach from the container, to allow it to survive your exit from the shell. Now let's list all the existing containers (it might take a short while before newcon is listed as fully up):

```
sudo lxc-ls --fancy
```

Assuming that the IP address for newcon (listed by our previous command) is 10.0.3.120, let's ssh into the container:

```
ssh ubuntu@10.0.3.120
```

And voila! A brand new computer playground, waiting for us to come and play! Now you've got no excuses: get to work.