CompTIA Linux+ XK0-005 Reference Guide

Get the knowledge and skills you need to become a Linux certified professional

Philip Inshanally

www.bpbonline.com

First published: 2024

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55515-735

www.bpbonline.com

Dedicated to

My Creator:

GOD Almighty

&

My loving Son ***Matthew Zach Inshanally***

About the Author

Philip Inshanally is a consultant with over 22+ years in the IT/ICT/OT industry and the first Guyanese IT Author to have written a US-based IT certification book (CompTIA Linux +, released in 2018). He is also the first Guyanese to have created international IT training video courses for various companies such as INE (www.ine.com) and Cybrary (cybrary.it), to name a few. Furthermore, Philip is the only Cisco Certified Internetwork Expert (CCIE#61204) residing in Guyana, South America.

He has extensive experience in IT/ICT/OT consulting, Cybersecurity, Wireless technologies, Designing, Planning, Implementing, and Documenting various IT/ICT/OT environments.

As one of the industry's leading IT/ICT/OT experts, he has a competitive edge due to his ongoing development and acquired excellent skills, which have helped him deal with customers locally and internationally.

He has always been methodical, detail-oriented, and production environment sensitive.

About the Reviewer

Vijay Gaikwad is an accomplished Linux enthusiast with a diverse skill set. He has a keen focus on Linux security, hardening, command line, diagnostic, and troubleshooting aspects, ensuring the accuracy and relevance of technical content related to Linux. His passion for Linux and dedication to the IT community make him a reliable resource for individuals and teams seeking guidance on Linux systems, security, and cloud integration. Moreover, as an AWS Cloud Practitioner, he possesses a strong understanding of cloud concepts and services. Vijay's well-rounded personality and commitment to continuous growth and learning are evident in his love for music, particularly the guitar. When not immersed in complex technical challenges, he can often be found strumming his guitar strings, showcasing his dedication to technology and the arts.

Acknowledgement

I want to express my gratitude to my Creator for bestowing his choicest blessings upon me. Also, my family and friends for their unwavering support and encouragement throughout this book's writing, especially my loving son Matthew Zach Inshanally.

I am also grateful to BPB Publications for their guidance and expertise in bringing this book to fruition. It was a long journey of revising this book, with valuable participation and collaboration of reviewers, technical experts, and editors.

I would also like to acknowledge the valuable contributions of my colleagues and co-worker during many years working in the tech industry, who have taught me so much and provided valuable feedback on my work.

Finally, I would like to thank all the readers who have taken an interest in my book and for their support in making it a reality. Your encouragement has been invaluable.

Preface

The CompTIA Linux+ is a certification program offered by CompTIA, a non-profit trade association that provides vendor-neutral IT certifications. The Linux+ certification is designed to validate the skills and knowledge of IT professionals who work with the Linux operating system.

The Linux+ certification is suitable for individuals who want to demonstrate their proficiency in Linux administration, system configuration, maintenance, and basic networking. It is often pursued by those seeking to build a career in Linux system administration or other IT roles that involve working with Linux-based systems.

With this book, you will gain the necessary knowledge and skillset required for achieving the CompTIA Linux+ certification. I hope you will find this book informative and helpful.

Chapter 1: Introduction to Linux Environment - This chapter explains everything needed for the reader to understand the Linux File Hierarchy System (FHS), navigating between the directories, viewing configuration files, etc. It introduces various methods of identifying and locating hardware information from within the terminal environments. Furthermore, this chapter also explains and illustrates the Grand Unified Bootloader 2.

Chapter 2: Files, Directories, and Storage - This chapter presents various methods of file compression and focuses on various ways to manipulate files and directories from within the terminal environment. Furthermore, this chapter explains various ways of managing storage.

Chapter 3: Processes, Services and Network Configuration - This chapter covers various ways for identifying and managing process(s). Next, this chapter focuses on handling services from within the terminal environment, such as: starting and stopping process(s) / daemon(s), and the service vs systemctl command. Furthermore, this chapter covers network configuration both from within the Desktop and terminal environment.

Chapter 4: Managing Modules and Software - This chapter allows the reader to learn various methods of managing modules from within the terminal environment. Furthermore, this chapter places heavy emphasis on software management.

Chapter 5: User and Password Management - The reader will learn various methods for managing users, such as adding, removing, and updating user properties, etc. Next, this chapter focuses on group management and password management.

Chapter 6: Firewall, Remote Access and SELinux - The reader will learn to configure various methods of TCP wrappers, firewalld, and iptables, among others. Next, this chapter focuses on remote access, and the reader will learn to configure remote access. Furthermore, the reader will learn to configure SELinux, which provides an extra layer of access control and enforcement.

Chapter 7: Shell Scripting and Containers - This chapter explains shell scripting in detail, starting with the basic structure of a script, then moving on to the for loop, while loop, and conditionals. Furthermore, this chapter covers containers in depth, such as Linux containers (LXC), Docker.

Chapter 8: Configuration Management with YAML, JSON and Ansible - This chapter is dedicated to configuration management. First, this chapter focuses on YAML or YAML Ain't Markup Language. Next, the reader will learn to manage configuration using JavaScript Object Notation (JSON). Furthermore, the reader will learn Ansible for configuration management.

Chapter 9: Troubleshooting Network and System Issues - This chapter explains various methods of troubleshooting. First, this chapter focuses on network troubleshooting, tools such as ping, ping6, traceroute, traceroute6, to name a few. Next, the reader will learn to troubleshoot common memory issues using tools such as free, top, among others. Following this, the reader will learn to troubleshoot hard disk problems using commands such as smartctl, iotop, to name a few. Furthermore, the reader will learn how to troubleshoot common boot issues by examining various log files and using commands such as fsck, mount, and many more.

Chapter 10: Mock Exams - The reader will be presented with two mock exams. The format of the questions is multiple choice in both mock exams. Furthermore, the questions cover materials learned throughout this book.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/6dbjtst

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/CompTIA-LinuxPlus-XK0-005-Reference-Guide**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

CHAPTER 1
Introduction to Linux Environment

Introduction

In this chapter, we will learn about the Linux **Filesystem Hierarchy Standard** (**FHS**) directories. Next, we will look for ways to obtain hardware information from the terminal environment. Finally, we will wrap this chapter with GRUB2.

Structure

This chapter will discuss the following topics:

- Filesystem hierarchy standard
- Finding hardware information
- Grand Unified Bootloader 2

Objectives

By the end of this chapter, you will be able to understand filesystem structure. You will become accustomed to directories and how they are used. It will be easier for you to locate hardware. Grand Unified Bootloader 2 will also be introduced to you through this chapter.

Filesystem hierarchy standard

When the term FHS is referenced, think of directory structure. You will find on almost all the Linux distributions, the FHS is identical. It defines the directory structure and content in UNIX-like operating systems such as Linux distros.

The need for a standardized filesystem hierarchy arose from the diverse nature of early Unix systems, where file organization varied significantly across different implementations. The FHS project was initiated in the early 1990s by the *Free Standards Group* (now the Linux Foundation) to address this issue. Its primary objective was to establish a typical structure simplifying software development, administration, and package management across Linux distributions.

Key principles of the FHS

The FHS is guided by several fundamental principles that shape its design and implementation. These principles include:

- **Filesystem hierarchy**: The FHS defines a consistent structure for organizing files and directories. It categorizes them based on purpose, ensuring each component has a specific role within the hierarchy.
- **Compatibility**: The FHS aims to maintain compatibility across different Linux distributions. By adhering to the standard, software developers and administrators can create and manage applications that work seamlessly across various systems.
- **Compliance**: Linux distributions seeking FHS compliance must adhere to specific guidelines to ensure consistent placement of files and directories. This promotes portability and simplifies the packaging and installation of software.

Across the various Linux Distros, the configuration files will usually be in the exact location regardless of Linux distribution (the **`/etc`** directory). This makes developing software for Linux much more accessible since software developers do not have to write different versions of applications for each distribution.

In the FHS environment, the files and directories appear under the root directory (**`/`**); this root directory is the parent of all other directories and sits at the top of the FHS.

A list of the commonly used directories across Linux distros is as follows:

- **`/`**: The root directory, this is the top-level directory. It is the parent for all other directories.
- **`/root/`**: The home directory of the root user.

- **`/boot`**: It contains files relating to system boot, such as the bootloader configurations, kernel images, and initial RAM disk (initram) files.
- **`/bin`**: The critical executable files that are necessary for system operations.
- **`/dev`**: It contains device files like hard disks or CD-ROMs.
- **`/sbin`**: It is similar to **`/bin`** but contains programs generally run by the root user.
- **`/etc`**: It contains configuration files.
- **`/home`**: This is the user's home directory.
- **`/lib`**: It contains libraries on 32-bit systems.
- **`/lib64`**: It contains libraries on 64-bit systems.
- **`/media`**: It is the mount point for removable media, usually used for auto-mounting external media.
- **`/mnt`:** It is the mount point for removable media manually.
- **`/usr`**: It contains most user utilities (**`/usr/bin/`**, **`/usr/lib/`**, etc.).
- **`/var`**: This contains the various files such as logs (auth, syslog), web server content, and package cache.
- **`/tmp`**: This contains temporary files created by programs.
- **`/proc`:** A virtual filesystem containing various runtime information such as devices, bus, interrupts, CPU, RAM, etc.

Now, let us look at the directories at the terminal using Ubuntu and Centos distributions.

The root (**/**) directory resides at the top of the directory structure. Using Ubuntu, let us check out our current working directory using the **`pwd`** command. Consider *Figure 1.1*:

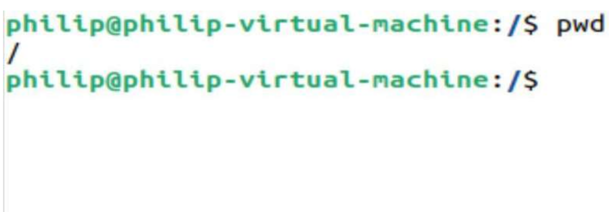

***Figure 1.1**: Present working directory*

Now, let us check the content of the root (**/**) directory using the **ls** command.

Consider *Figure 1.2*:

```
philip@philip-virtual-machine:/$ ls
bin    dev   lib   libx32      mnt   root  snap      sys  var
boot   etc   lib32 lost+found  opt   run   srv       tmp
cdrom  home  lib64 media       proc  sbin  swapfile  usr
philip@philip-virtual-machine:/$ ls -l
total 2191440
lrwxrwxrwx   1 root root          7 May 23 13:29 bin -> usr/bin
drwxr-xr-x   4 root root       4096 Jun  1 14:09 boot
drwxrwxr-x   2 root root       4096 May 23 13:30 cdrom
drwxr-xr-x  19 root root       4220 Aug  9 12:23 dev
drwxr-xr-x 129 root root      12288 Jun  1 14:10 etc
drwxr-xr-x   3 root root       4096 May 23 13:31 home
lrwxrwxrwx   1 root root          7 May 23 13:29 lib -> usr/lib
lrwxrwxrwx   1 root root          9 May 23 13:29 lib32 -> usr/lib32
lrwxrwxrwx   1 root root          9 May 23 13:29 lib64 -> usr/lib64
lrwxrwxrwx   1 root root         10 May 23 13:29 libx32 -> usr/libx32
drwx------   2 root root      16384 May 23 13:29 lost+found
drwxr-xr-x   3 root root       4096 May 23 13:57 media
drwxr-xr-x   2 root root       4096 Feb 22 23:57 mnt
drwxr-xr-x   2 root root       4096 Feb 22 23:57 opt
dr-xr-xr-x 345 root root          0 Aug  9 10:47 proc
drwx------   4 root root       4096 May 29 13:14 root
drwxr-xr-x  33 root root        860 Aug  9 10:51 run
lrwxrwxrwx   1 root root          8 May 23 13:29 sbin -> usr/sbin
drwxr-xr-x  13 root root       4096 May 29 10:45 snap
drwxr-xr-x   2 root root       4096 Feb 22 23:57 srv
-rw-------   1 root root 2243952640 May 23 13:29 swapfile
dr-xr-xr-x  13 root root          0 Aug  9 10:47 sys
drwxrwxrwt  19 root root       4096 Aug  9 12:25 tmp
drwxr-xr-x  14 root root       4096 Feb 22 23:57 usr
drwxr-xr-x  14 root root       4096 Feb 23 00:02 var
philip@philip-virtual-machine:/$
```

***Figure 1.2**: Output of the ls command*

We can see that the root (**/**) directory contains a range of other directories. For a hierarchical view, let us leverage the **tree** with the **-d** option (directory) command.

Consider *Figure 1.3*:

```
philip@philip-virtual-machine:/$ tree -d
.
├── bin -> usr/bin
├── boot
│   ├── efi  [error opening dir]
│   └── grub
│       ├── fonts
│       ├── i386-pc
│       ├── locale
│       └── x86_64-efi
├── cdrom
├── dev
│   ├── block
│   ├── bsg
│   ├── bus
│   │   └── usb
│   │       ├── 001
│   │       └── 002
│   ├── char
│   ├── cpu
│   │   ├── 0
│   │   └── 1
│   ├── disk
│   │   ├── by-id
│   │   ├── by-label
│   │   ├── by-partlabel
│   │   ├── by-partuuid
│   │   ├── by-path
│   │   └── by-uuid
│   ├── dma_heap
│   ├── dri
│   │   └── by-path
│   ├── fd -> /proc/self/fd
```

***Figure 1.3**: Output of the tree command with -d option*

Switch to our Centos system and check the root (**/**) directory.

Consider *Figure 1.4:*

```
philip@localhost:/
[philip@localhost /]$ pwd
/
[philip@localhost /]$ ls
afs  boot  etc   lib    media  opt   root  sbin  sys  usr
bin  dev   home  lib64  mnt    proc  run   srv   tmp  var
[philip@localhost /]$ 
```

Figure 1.4*: Output of the ls command with a similar structure*

As shown above, the structure is similar. Let us use the **tree** command with the **-d** option for a hierarchical view. Consider *Figure 1.5*:

```
.
├── afs
├── bin -> usr/bin
├── boot
│   ├── efi
│   │   └── EFI
│   │       └── centos [error opening dir]
│   ├── grub2 [error opening dir]
│   └── loader
│       └── entries [error opening dir]
├── dev
│   ├── block
│   ├── bsg
│   ├── bus
│   │   └── usb
│   │       ├── 001
│   │       └── 002
│   ├── char
│   ├── cpu
│   │   ├── 0
│   │   └── 1
│   ├── cs
│   ├── disk
│   │   ├── by-diskseq
│   │   ├── by-id
│   │   ├── by-label
│   │   ├── by-partuuid
│   │   ├── by-path
│   │   └── by-uuid
│   ├── dma_heap
:
```

Figure 1.5*: Output of the tree command with the -d option for a hierarchical view*

Now, look at the content of the **`/bin`** directory using the **`ls`** command. We will see a variety of commands which are a part of the operating system. Consider *Figure 1.6*:

```
philip@philip-virtual-machine:/bin$ pwd
/bin
philip@philip-virtual-machine:/bin$ ls
'['                                     mkzftree
 aa-enabled                             mmcli
 aa-exec                                mokutil
 aa-features-abi                        monitor-sensor
 aconnect                               more
 acpi_listen                            mount
 add-apt-repository                     mountpoint
 addpart                                mousetweaks
 airscan-discover                       mscompress
 alsabat                                msexpand
 alsaloop                               mt
 alsamixer                              mt-gnu
 alsatplg                               mtr
 alsaucm                                mtr-packet
 amidi                                  mv
 amixer                                 namei
 apg                                    nano
 apgbfm                                 nautilus
 aplay                                  nautilus-autorun-software
 aplaymidi                              nautilus-sendto
 apport-bug                             nawk
 apport-cli                             nc
 apport-collect                         nc.openbsd
 apport-unpack                          neqn
 appres                                 netcat
 appstreamcli                           networkctl
 apropos                                networkd-dispatcher
 apt                                    newgrp
 apt-add-repository                     ngettext
```

***Figure 1.6**: Output of ls command with the bin directory*

In contrast, in the **`/usr`** directory, you will notice some similarities inside the **`/usr/bin`** directory. However, the **`/bin`** is part of the core operating system and must be accessible before the **`/usr`** directory gets mounted. Consider *Figure 1.7*:

```
philip@philip-virtual-machine:/usr$ pwd
/usr
philip@philip-virtual-machine:/usr$ ls
bin  games  include  lib  lib32  lib64  libexec  libx32  local  sbin  share  src
philip@philip-virtual-machine:/usr$ ls /usr/bin
'['                                     mkzftree
 aa-enabled                             mmcli
 aa-exec                                mokutil
 aa-features-abi                        monitor-sensor
 aconnect                               more
 acpi_listen                            mount
 add-apt-repository                     mountpoint
 addpart                                mousetweaks
 airscan-discover                       mscompress
 alsabat                                msexpand
 alsaloop                               mt
 alsamixer                              mt-gnu
 alsatplg                               mtr
 alsaucm                                mtr-packet
 amidi                                  mv
 amixer                                 namei
 apg                                    nano
 apgbfm                                 nautilus
 aplay                                  nautilus-autorun-software
 aplaymidi                              nautilus-sendto
 apport-bug                             nawk
 apport-cli                             nc
 apport-collect                         nc.openbsd
 apport-unpack                          neqn
 appres                                 netcat
 appstreamcli                           networkctl
 apropos                                networkd-dispatcher
```

***Figure 1.7**: Contents of the /usr/bin directory*