Linux Fundamentals: A Comprehensive Guide

From Virtualization to Containerization — Master Linux from Scratch

Table of Contents

Introduction

This comprehensive guide covers Linux fundamentals from installation to advanced system administration. Whether you're a beginner looking to understand the basics or an aspiring system administrator, this course provides practical knowledge with real-world examples.

Author: Fouenang Miguel Bruce

Contact: miguelfouenanf@gmail.com | bfouenang237@gmail.com

1. Virtualization

What is Virtualization?

Virtualization is the process of running an operating system inside a virtual machine (VM) using virtualization software. This allows you to:

- Test different operating systems without affecting your host machine
- Create isolated development environments
- Study and practice system administration safely

Popular Virtualization Software

Software	Type	Best For
VirtualBox	Free, Open Source	Personal use, learning
VMware	Commercial	Enterprise environments
Virt-Manager	Linux-based	Cloud computing, KVM

Key Concepts

- A VM is a **separate system** from the host machine with its own IP and configurations
- VMs can share files with the host using **Guest Additions** (VirtualBox)
- Network engineers often configure VMs on the same network as the host for better communication

2. VirtualBox Setup

Requirements

1. VirtualBox software
2. ISO image of your desired Linux distribution (Ubuntu LTS, Linux Mint, Fedora, etc.)

Installation Steps

1. Download VirtualBox from virtualbox.org
2. Download your preferred Linux ISO from the official distribution website
3. Create a new VM in VirtualBox and follow the installation wizard
4. Install Guest Additions for file sharing between host and VM

Note for Linux Users

On Linux, you can install VMs without mixing data with your host machine. Always remember that a VM operates independently with its own network configuration.

3. Terminal Basics

Opening the Terminal

- **Ubuntu/GNOME:** GNOME Terminal
- **Xfce:** Xfce Terminal

Essential Terminal Commands

Command	Description
`clear`	Clear terminal screen
`Ctrl + L`	View previous commands
`Ctrl + U`	Clear line from cursor to beginning
`Ctrl + K`	Clear line from cursor to end

Command	Description
`reset`	Reset terminal after crashes
`whatis <command>`	Display command description
`history`	Show last ~700 commands
`!n`	Execute command number n from history
`Ctrl + R`	Search command history

Pro Tip

Use `history 25` to view only the last 25 commands instead of the entire history.

4. Navigation Commands

Basic Navigation

Command	Description	Example
`pwd`	Print working directory	`pwd`
`cd <dir>`	Change directory	`cd Documents`
`cd ..`	Go to parent directory	`cd ..`
`cd~` or `cd`	Go to home directory	`cd`
`cd /<dir>`	Absolute path navigation	`cd /Downloads`

Listing Files

Command	Description
`ls`	List files in current directory
`ls -a`	List all files (including hidden)
`ls -l`	Detailed list with permissions and ownership
`ls /path`	List files without changing directory

5. File and Directory Operations

Creating Directories

```
# Create single directory

mkdir foldername

# Create multiple directories
```

```
mkdir folder1 folder2 folder3

# Create nested directories

mkdir -p parent/child/grandchild
```

Creating Files

Method	Command	Example
touch	`touch <file>`	`touch hello.txt`
echo	`echo > <file>`	`echo > file.txt`
cat	`cat > <file>`	`cat > info.txt`

> **Note:** Linux doesn't require file extensions to function. A file without an extension works perfectly fine.

Copying Files

```
# Copy single file

cp file.txt destination/

# Copy multiple files

cp file1 file2 file3 destination/

# Copy with wildcards

cp *.txt destination/
```

Moving and Renaming

```
# Move file

mv file.txt destination/

# Rename file

mv oldname.txt newname.txt
```

Deleting Files and Directories

```
# Delete file

rm filename
```

```
# Delete empty directory

rmdir foldername

# Delete directory with contents (recursive)

rm -r foldername

# Delete with wildcard

rm M* -r
```

WARNING: Never use `rm -rf /` — this will delete your entire system!

Note: Files deleted from the terminal do NOT go to the trash can.

6. Viewing Files

Downloading Files

```
wget http://example.com/file.txt
```

Viewing Commands

Command	Description	Example
`more`	View file page by page	`more filename`
`more -10`	Show from line 10	`more -10 filename`
`more +150`	Start at line 150	`more +150 filename`
`less`	Advanced file viewer	`less filename`
`less -p "word"`	Search for word	`less -p "Romeo" file.txt`
`head`	Show first lines	`head -5 filename`
`tail`	Show last lines	`tail -3 filename`

7. Text Editors

Vim

Vim is a powerful, modal text editor available on virtually all Linux systems.

Vim Modes

1. **Normal Mode** — Navigate and execute commands
2. **Insert Mode** — Edit text
3. **Command Mode** — Execute commands (save, quit, etc.)

Essential Vim Commands

Command	Action
`i`	Enter insert mode
`Esc`	Return to normal mode
`:w`	Save file
`:q`	Quit
`:wq`	Save and quit
`:q!`	Quit without saving
`/word`	Search for “word”
`n`	Next search result

Learning Vim

```
vimtutor # Interactive Vim tutorial
```

Nano

Nano is a simpler, more beginner-friendly text editor.

Shortcut	Action
`Ctrl + O`	Write (save) file
`Ctrl + S`	Save
`Ctrl + X`	Exit

Creating Backups with Nano

```
nano -B filename # Creates automatic backups
```

8. Linux Directory Structure

Key Directories

Directory	Purpose
`/bin`	Essential system binaries
`/boot`	Boot loader files
`/home`	User home directories
`/lib`	System libraries
`/opt`	Optional/additional software
`/proc`	Process information
`/root`	Root user home

Directory	Purpose
`/tmp`	Temporary files (cleared on reboot)
`/usr`	User programs and utilities
`/var/log`	System logs

Disk Usage Commands

```
df -h # Show disk space (human-readable)
df -hT / # Show disk space for root partition
du -sh ~ # Show directory size
du -s ~ # List all files with sizes
```

9. File Compression

Tar Archives

```
# Create tar archive

tar -cf archive.tar filename

# Create compressed tar.gz

tar -zcf archive.tar.gz filename

# Extract tar archive

tar -xf archive.tar

# Extract tar.gz

tar -zxf archive.tar.gz
```

Zip Archives

```
# Create zip

zip archive.zip filename

# Extract zip

unzip archive.zip
```

10. Finding Files

The `find` Command

```
# Find by name

find -name "filename"

# Find with wildcards

find -name "*.txt"

# Find by size

find -size +10k # Larger than 10KB
find -size 100k # Exactly 100KB
find -size 10M # Exactly 10MB
```

11. User Management

Getting User Information

Command	Purpose
`whoami`	Show current user
`id`	Show user ID and group ID
`groups`	Show groups user belongs to

Switching Users

```
su username # Switch to user
sudo su # Switch to root
```

Note: Root users see **#** prompt; regular users see .

Creating Users

```
# Using useradd (basic)

sudo useradd username -m

# With custom home directory and shell

sudo useradd username -m -d /home/custom -s /bin/bash

# Using adduser (interactive, more detailed)

sudo adduser username
```

Modifying Users

```
# Lock user account

sudo usermod -L username

# Unlock user account

sudo usermod -U username

# Change home directory

sudo usermod -d /home/newdir username

# Add to group

sudo usermod -aG groupname username

# Remove from group

sudo deluser username groupname
```

Deleting Users

```
sudo deluser username
```

User Types in Linux

1. **Root (Superuser)** — Full system access
2. **Normal Users** — Standard user accounts
3. **Service Accounts** — For running services
4. **Daemon Users** — For system services (UID 1-4096)

12. Group Management

Creating Groups

```
sudo groupadd groupname
```

Viewing Groups

```
tail /etc/group # View last entries in groups file
```

Adding Users to Groups

```
sudo usermod -aG groupname username
```

Deleting Groups

```
sudo groupdel groupname
```

13. File Permissions

Understanding Permission Notation

When you run `ls -l`, you see permissions like `-rw-rw-r--`:

Position	Meaning
1st character	File type (`-` file, `d` directory)
2-4	Owner permissions (rwx)
5-7	Group permissions (rwx)
8-10	Others permissions (rwx)

Permission Values

Permission	Value	Description
`r`	4	Read
`w`	2	Write
`x`	1	Execute

chmod Command

```
# Symbolic notation

chmod o+w filename # Add write for others
chmod go-rw filename # Remove read/write for group and others
chmod a=rw filename # Set read/write for all

# Numeric notation

chmod 755 filename # rwxr-xr-x
chmod 644 filename # rw-r--r--
```

chown Command (Change Owner)

```
# Change owner

sudo chown user filename

# Change owner and group
```

```
sudo chown user:group filename

# Recursive change

sudo chown -R $USER:$USER directory/
```

chgrp Command (Change Group)

```
sudo chgrp newgroup filename
```

Special Permissions

```
# Set executable for owner

chmod u+x filename

# Set SUID bit

chmod u+s filename

# Set sticky bit (prevent deletion)

chmod +t directory/
```

14. Links and Inodes

Hard Links

```
ln original.txt link.txt
```

- Points directly to the file's inode
- Same content, same disk space
- If original is deleted, link still works

Symbolic (Soft) Links

```
ln -s original.txt softlink.txt
```

- Points to the filename, not inode
- If original is deleted, link breaks
- Can link across filesystems

  Recommendation: Prefer soft links to avoid data loss.

15. Input / Output Streams

The `cat` Command

```
# View file

cat filename

# View multiple files

cat file1 file2 file3

# Concatenate to new file

cat file1 file2 > combined.txt

# Create file

cat > filename

# Append to file

cat >> filename
```

File Command

```
file filename # Determine file type
```

16. Pipelines and grep

The Pipe Operator (`|`)

Pipes send the output of one command as input to another.

```
# Count lines in file

cat file.txt | wc -l

# Count files in directory

ls | wc -l

# Find and show first 5 text files

find -name "*.txt" | head -5

# Sort and get unique entries

find -name "*.txt" | sort | uniq
```

The `wc` Command

Option	Description
`-l`	Count lines
`-w`	Count words
`-c`	Count bytes

The `grep` Command

```
# Search for word in file

grep "word" filename

# Search with cat pipeline

cat file.txt | grep "word"

# Count occurrences

cat file.txt | grep "word" | wc -l

# Show only matches

grep -o "word" filename
```

Analyzing Logs

```
# View system logs

cat /var/log/syslog | grep systemd
cat /var/log/syslog | grep cron
cat /var/log/syslog | grep cron | grep root | wc -l
```

Logical Operators

```
# AND - execute both

echo "Hello" && echo "World"

# OR - execute second if first fails

head file.txt || echo "Error␣reading␣file"
```

17. Process Management

The `top` Command

System task manager showing:

- Running processes
- CPU usage
- Memory usage
- Process IDs (PID)

The `htop` Command

Modern alternative to `top` with:

- Color-coded display
- Mouse support
- CPU core visualization
- Function key shortcuts (F1-F10)

Process Commands

Command	Description
`top`	Basic process viewer
`htop`	Enhanced process viewer
`btop`	Modern resource monitor
`ps`	Static process snapshot
`pidof <program>`	Find process ID of program

Killing Processes

```
kill PID # Kill by process ID
sudo kill PID # Kill any process as root
```

Swap Memory

Swap is virtual RAM stored on disk, used when physical RAM is full.

```
# View swap usage

free -h

# View swap file location

swapon -s
```

systemctl — Service Management

```
# Check service status

systemctl status ssh

# Start/stop/enable/disable services

systemctl start ssh
systemctl stop ssh
systemctl enable ssh
systemctl disable ssh

# List all services

systemctl | wc -l
```

18. Package Management

APT (Advanced Package Tool)

Used on Debian-based systems (Ubuntu, Linux Mint).

```
# Update package lists

sudo apt update

# Upgrade installed packages

sudo apt upgrade

# Install package

sudo apt install packagename

# Remove package

sudo apt remove packagename

# Purge package (remove with config)

sudo apt purge packagename

# Search for packages

apt search keyword

# Show package info

apt show packagename
```

```
# List installed packages

apt list --installed
```

DPKG (Debian Package)

For installing `.deb` files directly:

```
# Install .deb package

sudo dpkg -i package.deb

# List installed packages

dpkg -l

# Remove package

sudo dpkg -r packagename
```

Adding Repositories

```
# Check system codename

lsb_release -a

# Repositories are stored in

/etc/apt/sources.list.d/
```

Other Package Managers

System	Command
Red Hat / Fedora	`dnf` / `yum`
Arch Linux	`pacman`
Universal	`snap`, `flatpak`

19. Cron Jobs

What is Cron?

Cron is a time-based job scheduler for automating tasks.

Crontab Commands

```
# List cron jobs
```

```
crontab -l

# Edit cron jobs

crontab -e

# Remove all cron jobs

crontab -r
```

Cron Syntax

```
* * * * * command

| | | | |
| | | | +-- Day of week (0-7, Sunday = 0 or 7)
| | | +---- Month (1-12)
| | +------ Day of month (1-31)
| +-------- Hour (0-23)

+---------- Minute (0-59)
```

Special Strings

String	Description
`@reboot`	Run at startup
`@hourly`	Run every hour
`@daily`	Run once a day
`@weekly`	Run once a week
`@monthly`	Run once a month
`@yearly`	Run once a year

Cron Logs

```
# Check cron service status

systemctl status cron

# Configure cron logging in

/etc/rsyslog.d/50-default.conf
```

20. Firewall (iptables)

Basic Commands

```
# List all rules

sudo iptables -L

# List with line numbers

sudo iptables -L --line-numbers

# Flush all rules

sudo iptables -F
```

Managing Rules

```
# Add rule to INPUT chain

sudo iptables -A INPUT -s "192.168.1.122" -j DROP
sudo iptables -A INPUT -s "192.168.1.122" -j ACCEPT

# Insert rule at position

sudo iptables -I INPUT 1 -s "192.168.1.122" -j ACCEPT

# Delete rule by number

sudo iptables -D INPUT 1
```

Port Rules

```
# Allow SSH (port 22)

sudo iptables -I INPUT -p tcp --dport 22 -j ACCEPT

# Allow HTTP (port 80)

sudo iptables -I INPUT -p tcp --dport 80 -j ACCEPT
```

Default Policies

```
# Set default INPUT policy to DROP

sudo iptables -P INPUT DROP
```

Saving Rules

```
# Save rules

sudo iptables-save > iptables_rules

# Restore rules

sudo iptables-restore < iptables_rules
```

Note: Rules are lost on reboot unless saved and restored.

21. Bash Scripting

What is Bash?

Bash (Bourne Again SHell) is a command interpreter and scripting language for automating tasks.

Creating Scripts

```
# Shebang line (first line of script)

#!/bin/bash

# Or for Python

#!/usr/bin/python3
```

Making Scripts Executable

```
chmod +x script.sh
```

Finding Interpreter Paths

```
whereis bash
whereis python3
```

Viewing Available Shells

```
cat /etc/shells
```

Example: Ping Script

```
#!/bin/bash

ping -c 4 google.com
```

22. Docker Fundamentals

What is Docker?

Docker is a containerization platform for building, shipping, and running applications in isolated environments.

Containers **Virtual Machines** — Containers are lighter and share the host OS kernel.

Installation

```
# Ubuntu/Debian

sudo apt install docker.io -y
```

Essential Commands

Command	Description
`docker pull <image>`	Download image from Docker Hub
`docker run <image>`	Run a container
`docker run -d <image>`	Run detached (background)
`docker run -it <image>`	Run interactive with TTY
`docker run -p 8080:80 <image>`	Map ports
`docker ps`	List running containers
`docker ps -a`	List all containers
`docker rm <container>`	Remove container
`docker rmi <image>`	Remove image
`docker exec <container> <cmd>`	Execute command in container

First Container

```
# Hello World

docker run hello-world

# Nginx web server

docker pull nginx
docker run -d -p 8080:80 nginx
```

Running Without Sudo

```
# Add user to docker group
```

```
sudo usermod -aG docker $USER
sudo systemctl restart docker
```

Dockerfile

A Dockerfile automates image creation:

```
FROM nginx:latest
COPY . /usr/share/nginx/html
EXPOSE 80
```

Build and run:

```
docker build -t myimage .
docker run -d -p 8080:80 myimage
```

Docker Compose

For multi-container applications:

```
# Install

sudo apt install docker-compose

# Start services

docker-compose up -d

# Stop services

docker-compose down -v
```

Cleanup Commands

```
# Remove all containers

docker rm $(docker ps -aq)

# Remove all images

docker rmi $(docker images -q)

# Force remove running container

docker rm -f <container-id>
```

Conclusion

This guide covered the essential skills for Linux system administration:

- Setting up virtualized Linux environments
- Navigating and managing the filesystem
- Understanding users, groups, and permissions
- Managing processes and services
- Automating tasks with cron and bash
- Deploying applications with Docker

Continue practicing these commands and explore the man pages (`man <command>`) for deeper understanding.

Author: Fouenang Miguel Bruce

Contact: miguelfouenanf@gmail.com | bfouenang237@gmail.com

Happy Linuxing!