Comprehensive Linux Study Guide

This guide provides a thorough overview of key Linux concepts, designed to help you prepare for your assessment.

1. Introduction to the Linux Operating System

What is Linux?

Linux is not technically an operating system itself, but rather the **kernel**—the core component of an OS. An operating system is the full software suite that manages computer hardware and software resources, and provides common services for computer programs. A **Linux distribution** (often called a "distro") combines the Linux kernel with other software (like the GNU toolset, graphical user interfaces, and utilities) to create a complete, usable OS.

Examples of Linux Distributions:

- Ubuntu
- Debian
- Fedora
- CentOS / Red Hat Enterprise Linux (RHEL)
- Arch Linux

Key Features of Linux:

- **Open Source:** The source code is freely available for anyone to view, modify, and distribute.
- **Multi-User:** Multiple users can access system resources like memory, CPU, and storage simultaneously.
- **Multi-Tasking:** The system can run multiple processes (applications) at the same time.
- **Security:** Features a robust user/group permission model and advanced security features like SELinux.

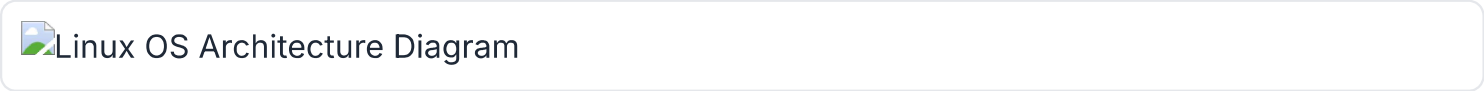
Portability: Can run on a vast range of hardware, from supercomputers and servers to mobile phones and embedded devices.

Stability & Reliability: Linux systems are known for being able to run for long periods without needing a reboot.

---

2. Linux OS Architecture

The Linux operating system architecture is structured in layers. This design separates different functionalities, allowing the system to be modular and stable.



The main layers are:

Hardware: This is the physical layer, which consists of all the peripheral devices attached to the system, such as RAM, CPU, hard disk drives, and network interface cards.

Kernel: This is the core of the operating system. It acts as the central manager for all hardware resources. The kernel is responsible for process management, memory management, and device management. It provides a bridge between the hardware and the applications running on top of it.

Shell: The shell is the command-line interface (CLI) that allows a user to interact with the kernel. It takes commands from the user, interprets them, and instructs the kernel to perform the corresponding tasks. Popular shells include Bash (Bourne Again Shell), Zsh, and Fish.

System Utilities & Applications: This is the outermost layer where user applications and system utilities run. These are the programs that users interact with directly, such as text editors, web browsers, and command-line tools like `ls` or `cp`. These applications make requests to the kernel (via system calls) to access hardware or perform other system-level functions.

---

3. The Linux Kernel and Hypervisors

The Linux Kernel

As described in the architecture, the kernel is the heart of the operating system. It acts as the primary bridge between the computer's hardware and the software applications running on it.

Core Responsibilities of the Kernel:

- **Process Management:** Manages the execution of applications (processes), including scheduling when each process gets to use the CPU.
- **Memory Management:** Allocates and manages the system's memory (RAM), ensuring processes don't interfere with each other.
- **Device Drivers:** Acts as an intermediary between the hardware (like disks, graphics cards, network cards) and software.
- **System Calls & Security:** Provides a secure interface for applications to request services from the kernel (e.g., opening a file, creating a new process). This is how the kernel enforces security rules.

Linux uses a **monolithic kernel**, which means that most of the core OS services (like the ones listed above) run in a single, large block of code within the kernel's special, protected memory space.

Hypervisors

A **hypervisor**, or Virtual Machine Monitor (VMM), is software that creates and runs virtual machines (VMs). It allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.

Types of Hypervisors:

- **Type 1 (Bare-Metal):** Runs directly on the host's hardware to control the hardware and to manage guest operating systems. Examples: VMware ESXi, Microsoft Hyper-V, KVM.
- **Type 2 (Hosted):** Runs on a conventional operating system just like other computer programs. Examples: VMware Workstation, VirtualBox, Parallels.

Linux and Hypervisors: KVM Linux has a built-in, Type 1 hypervisor technology called **KVM (Kernel-based Virtual Machine)**. Because KVM is part of the Linux kernel, it can leverage the

kernel's existing scheduler and memory management, making it highly efficient and powerful.

4. The Linux File System

In Linux, a core principle is **"everything is a file."** This means that not just documents and programs are treated as files, but also directories, hardware devices (like your mouse or hard drive), and system information streams.

Filesystem Hierarchy Standard (FHS)

Linux organizes files in a hierarchical, tree-like structure. The FHS is a standard that defines the names, locations, and contents of the main directories.

Directory	Purpose
`/`	**Root Directory:** The top-level directory of the entire file system.
`/bin`	**Essential User Binaries:** Contains executable programs needed for all users (e.g., `ls`, `cp`, `ping`).
`/sbin`	**System Binaries:** Contains essential executables for system administration (e.g., `reboot`, `fdisk`).
`/etc`	**Configuration Files:** Contains system-wide configuration files for all programs.
`/dev`	**Device Files:** Contains special files that represent hardware devices (e.g., `/dev/sda`, `/dev/null`).
`/proc`	**Process Information:** A virtual filesystem providing information about system processes and the kernel.
`/var`	**Variable Files:** For files whose content is expected to grow, like logs (`/var/log`), caches, etc.
`/tmp`	**Temporary Files:** For temporary files created by users and applications. Often cleared on reboot.
`/usr`	**User Programs:** Contains the majority of multi-user utilities and applications.
`/home`	**User Home Directories:** Contains personal directories for each user (e.g., `/home/johndoe`).
`/boot`	**Boot Loader Files:** Contains files needed to boot the system, including the Linux kernel itself.
`/lib`	**Essential Shared Libraries:** Contains library files needed by the binaries in `/bin` and `/sbin`.
`/opt`	**Optional Add-on Software:** For installing third-party applications.
`/mnt`	**Mount Point:** A temporary mount point for mounting filesystems manually.

Directory	Purpose
`/media`	**Removable Media:** A mount point for removable media like USB drives and CD-ROMs.
`/srv`	**Service Data:** Contains site-specific data served by the system, for services like FTP or web servers.

5. Essential Linux Commands (~80)

Commands are case-sensitive.

File & Directory Management

Command	Description	Syntax Example
`ls`	Lists directory contents.	`ls -l /home`
`cd`	Changes the current directory.	`cd /var/log`
`pwd`	Prints the name of the current working directory.	`pwd`
`mkdir`	Creates a new directory.	`mkdir new_folder`
`rmdir`	Removes an *empty* directory.	`rmdir old_folder`
`rm`	Removes files or directories.	`rm file.txt` or `rm -r directory`
`cp`	Copies files or directories.	`cp source.txt destination.txt`
`mv`	Moves or renames files or directories.	`mv old_name.txt new_name.txt`
`touch`	Creates an empty file or updates its timestamp.	`touch new_file.txt`
`cat`	Concatenates and displays the content of files.	`cat /etc/passwd`
`less`	Views file content one page at a time (allows backward navigation).	`less large_log_file.log`
`more`	Views file content one page at a time (only forward navigation).	`more file.txt`
`head`	Displays the first few lines of a file.	`head -n 5 file.txt`

Command	Description	Syntax Example
`tail`	Displays the last few lines of a file.	`tail -f /var/log/syslog`
`find`	Searches for files in a directory hierarchy.	`find /home -name "*.txt"`
`locate`	Finds files by name (uses a pre-built database).	`locate httpd.conf`
`updatedb`	Updates the database for `locate`.	`sudo updatedb`
`file`	Determines the type of a file.	`file my_script.sh`

File Permissions

Command	Description	Syntax Example
`chmod`	Changes the permissions of a file or directory.	`chmod 755 script.sh`
`chown`	Changes the owner of a file or directory.	`sudo chown user:group file.txt`
`chgrp`	Changes the group ownership of a file.	`sudo chgrp developers file.txt`

How `chmod` Octal (Numeric) Notation Works

Permissions in Linux are assigned to three categories of users:

Owner: The user who owns the file.

Group: The group that owns the file.

Others: All other users on the system.

Each category can be granted three types of permissions, which are represented by a numeric value:

Read (r) = 4

Write (w) = 2

Execute (x) = 1

To set permissions, you add the values for the desired access level for each user category. The `chmod` command uses a three-digit number representing the permissions for **Owner, Group, and Others**, respectively.

Example 1: `chmod 777` **(Full Access for Everyone)**

Owner: 7 → 4 + 2 + 1 = Read + Write + Execute (`rwx`)

Group: 7 → 4 + 2 + 1 = Read + Write + Execute (`rwx`)

Others: 7 → 4 + 2 + 1 = Read + Write + Execute (`rwx`)

Resulting permissions: `rwxrwxrwx`

Example 2: `chmod 755` **(Common for Web Files & Directories)**

Owner: 7 → 4 + 2 + 1 = Read + Write + Execute (`rwx`)

Group: 5 → 4 + 0 + 1 = Read + Execute (`r-x`)

Others: 5 → 4 + 0 + 1 = Read + Execute (`r-x`)

Resulting permissions: `rwxr-xr-x`

Example 3: `chmod 644` **(Common for Files)**

Owner: 6 → 4 + 2 + 0 = Read + Write (`rw-`)

Group: 4 → 4 + 0 + 0 = Read (`r--`)

Others: 4 → 4 + 0 + 0 = Read (`r--`)

Resulting permissions: `rw-r--r--`

Example 4: `chmod 700` **(Private for Owner)**

Owner: 7 → 4 + 2 + 1 = Read + Write + Execute (`rwx`)

Group: 0 → 0 + 0 + 0 = No permissions (`---`)

Others: 0 → 0 + 0 + 0 = No permissions (`---`)

Resulting permissions: `rwx------`

Text Processing

Command	Description	Syntax Example
`grep`	Searches for patterns in text.	`grep "error" /var/log/syslog`
`sed`	A stream editor for filtering and transforming text.	`sed 's/old/new/g' file.txt`
`awk`	A powerful pattern scanning and processing language.	`awk '{print $1}' file.txt`
`sort`	Sorts lines of text files.	`sort names.txt`
`uniq`	Reports or omits repeated lines.	`sort names.txt
`wc`	Prints newline, word, and byte counts for each file.	`wc -l file.txt`
`cut`	Removes sections from each line of files.	`cut -d':' -f1 /etc/passwd`
`tr`	Translates or deletes characters.	`cat file.txt
`diff`	Compares two files line by line.	`diff file1.txt file2.txt`

System Information

Command	Description	Syntax Example
`uname`	Prints system information (kernel name, version, etc.).	`uname -a`
`hostname`	Shows or sets the system's host name.	`hostname`
`df`	Reports file system disk space usage.	`df -h`
`du`	Estimates file and directory space usage.	`du -sh /home/user`
`free`	Displays amount of free and used memory.	`free -m`
`top`	Displays running processes dynamically.	`top`
`ps`	Reports a snapshot of current processes.	`ps aux`
`who`	Shows who is logged on.	`who`
`whoami`	Prints the effective user ID.	`whoami`
`date`	Prints or sets the system date and time.	`date`
`cal`	Displays a calendar.	`cal`
`uptime`	Shows how long the system has been running.	`uptime`

Process Management

Command	Description	Syntax Example
`kill`	Sends a signal to a process (usually to terminate it).	`kill 1234`
`killall`	Kills processes by name.	`killall firefox`
`pkill`	Kills processes based on name and other attributes.	`pkill -u username`
`bg`	Puts a suspended process into the background.	`bg %1`
`fg`	Brings a background process into the foreground.	`fg %1`
`jobs`	Lists active jobs (background processes).	`jobs`

Networking

Command	Description	Syntax Example
`ping`	Sends ICMP ECHO_REQUEST packets to network hosts.	`ping google.com`
`ip`	Shows/manipulates routing, devices, policy routing and tunnels.	`ip addr show`
`ifconfig`	(Legacy) Configures network interfaces.	`ifconfig`
`netstat`	(Legacy) Prints network connections, routing tables, etc.	`netstat -tuln`
`ss`	(Modern replacement for `netstat`) Dumps socket statistics.	`ss -tuln`
`wget`	A non-interactive network downloader.	`wget https://example.com/file.zip`
`curl`	A tool to transfer data from or to a server.	`curl -O https://example.com/file.zip`
`ssh`	Secure Shell client for remote login.	`ssh user@remote_host`
`scp`	Securely copies files between hosts on a network.	`scp file.txt user@remote:/home/user/`

User & Group Management

Command	Description	Syntax Example
`useradd`	Creates a new user.	`sudo useradd newuser`
`userdel`	Deletes a user account.	`sudo userdel olduser`
`usermod`	Modifies a user account.	`sudo usermod -aG sudo newuser`
`groupadd`	Creates a new group.	`sudo groupadd developers`
`groupdel`	Deletes a group.	`sudo groupdel staff`
`passwd`	Changes a user's password.	`passwd`
`su`	Switches user identity.	`su - username`
`sudo`	Executes a command as another user (usually root).	`sudo apt update`

Archiving & Compression

Command	Description	Syntax Example
`tar`	An archiving utility.	`tar -cvf archive.tar /path/to/dir`
`gzip`	Compresses files (uses `.gz` extension).	`gzip file.txt`
`gunzip`	Decompresses files compressed by `gzip`.	`gunzip file.txt.gz`
`zip`	A package and compression utility for zip files.	`zip archive.zip file1 file2`
`unzip`	Extracts files from a ZIP archive.	`unzip archive.zip`

Package Management (Debian/Ubuntu Example)

Command	Description	Syntax Example
`apt update`	Resynchronizes the package index files from their sources.	`sudo apt update`
`apt upgrade`	Installs the newest versions of all packages currently installed.	`sudo apt upgrade`
`apt install`	Installs a new package.	`sudo apt install htop`
`apt remove`	Removes a package.	`sudo apt remove htop`
`apt search`	Searches for a package.	`apt search browser`

Shell & Scripting

Command	Description	Syntax Example
`echo`	Displays a line of text.	`echo "Hello, World!"`
`history`	Displays the command history.	`history`
`man`	Displays the on-line manual pages for a command.	`man ls`
`alias`	Creates a shortcut for a longer command.	`alias ll='ls -alF'`
`export`	Sets an environment variable.	`export VAR="value"`
`source`	Executes commands from a file in the current shell.	`source ~/.bashrc`
`which`	Shows the full path of shell commands.	`which bash`
`clear`	Clears the terminal screen.	`clear`

I/O Redirection & Pipes

Operator	Description	Syntax Example
`>`	Redirects standard output to a file (overwrites).	`ls -l > file_list.txt`
`>>`	Appends standard output to a file.	`echo "New log" >> app.log`
`<`	Redirects standard input from a file.	`sort < names.txt`
`	`	`	**Pipe:** Redirects the output of one command to the input of another.
`2>`	Redirects standard error to a file.	`bad_command 2> error.log`

6. Basic Terminal Shortcuts

Shortcut	Action
`Ctrl + C`	Interrupt (kill) the current foreground process.
`Ctrl + Z`	Suspend the current foreground process.
`Ctrl + D`	Log out of the current session (sends End-of-File, EOF).
`Ctrl + L`	Clear the terminal screen (same as the `clear` command).
`Ctrl + A`	Move the cursor to the beginning of the line.
`Ctrl + E`	Move the cursor to the end of the line.
`Ctrl + R`	Search backward through the command history.
`Ctrl + U`	Clear the line from the cursor to the beginning.
`Ctrl + K`	Clear the line from the cursor to the end.
`Tab`	Autocomplete a command, filename, or directory name.
`Up Arrow`	Show the previous command from history.
`Down Arrow`	Show the next command from history.
`!!`	Execute the very last command again.
`!n`	Execute the nth command from `history`.
`Shift + PageUp`	Scroll up in the terminal.
`Shift + PageDown`	Scroll down in the terminal.