CyberGuardians: Penetration Tester Development Program

PRIME DIGITAL CONSULTANT

depa

หลักสูตร

นักทดสอบการเจาะระบบ

Penetration Tester

ภายใต้โครงการพัฒนานักทดสอบการเจาะระบบ เพื่อเสริมสร้างทักษะ และการจ้างงานสำหรับนักศึกษาจบใหม่

Speaker

- นาย เจษฎา ทองก้านเหลือง (ต้อม)
- กรรมการผู้จัดการ (Managing Director)
- บริษัท ที-เน็ต ไอที โซลูชัน จำกัด
- Certificate: CCNA, CEH, CHFI ,ECSA, CISO
- CompTIA Security+, CompTIA Pentest+, CompTIA Project+, CompTIA Network+, CompTIA CySA+, CompTIA Cloud+
- Peplink Certified Engineer (PCE), Peplink Sales Specialist (PSS)
- Fortinet NSE1, Fortinet NSE2
- IT Specialist Certification (ITS): Cyber Security, Network Security
- Certificate Name: Certificate of Competence in Zero Trust, Certificate of Cloud Security Knowledge
- E-Mail : Jedsada@tnetitsolution.co.th
- Facebook: ผู้ดูแลกลุ่มสอนแฮกแบบหมูๆ
- Website : www.tnetitsolution.co.th

Agenda

- ❑ Introduction to Linux and System Architecture
- ❑ User and Permission Management
- ❑ Service and Process Management
- ❑ Core Linux Security Concepts
- ❑ Patch Management and Hardening
- ❑ Intrusion Detection and Auditing

Introduction to Linux and System Architecture

Linux Security Tools

Hardening & Auditing

- Lynis
- OpenSCAP
- Bastille Linux
- grsecurity

Vulnerability Scanning

- Nessus
- Rapid7 InsightVM
- OpenVAS
- Nmap + NSE Scripts

Authentication & Access Control

- Fail2ban
- SSHGuard
- PAM Modules
- Google Authenticator PAM

Network Security & Monitoring

- iptables/nftables/firewalld/ufw
- tcpdump/Wireshark/Tshark
- Suricata/Snort
- Security Onion

EDR

- Wazuh
- Osquery
- Auditd/Auditbeat

SIEM

- Elastic Stack (ELK)
- Syslog-ng / rsyslog
- Prometheus + Grafana

Agenda

❑ Introduction to Linux and System Architecture

- Linux distributions overview (Debian, RHEL, Ubuntu, etc.)
- Linux boot process and system architecture
- Filesystem hierarchy (/etc, /var, /home, /usr, etc.)
- Basic command-line tools (ls, cd, grep, cat, less, etc.)
- **Hands-on:** Navigating the filesystem, using help and man pages

Linux distributions

What Are Linux Distributions?

- A **Linux distribution (distro)** is an operating system built around the **Linux kernel**, bundled with software like package managers, desktop environments, utilities, and system tools. Each distro caters to different goals: **stability, performance, user-friendliness, or cutting-edge tech**.
- **Different needs:**
  - Ease of use
  - Stability vs bleeding-edge software
  - Security
  - Enterprise support
  - Specialized purposes (e.g., penetration testing, IoT)

Enterprise Distributions

Distribution	Description	Key Features	Use Cases
Red Hat Enterprise Linux (RHEL)	Commercial enterprise distro with support from Red Hat (IBM).	SELinux, FIPS, system roles, certified cloud & hardware support	Enterprises, finance, healthcare
SUSE Linux Enterprise Server (SLES)	Enterprise distro by SUSE with strong YaST tools and SAP support.	YaST, Snapper (btrfs snapshots), AutoYaST	Enterprises, SAP workloads
Oracle Linux	RHEL-compatible distro with Oracle enhancements.	Unbreakable Enterprise Kernel, Ksplice (live patching)	Oracle DB environments, enterprise servers

Server & Cloud Distributions

Distribution	Description	Key Features	Use Cases
Ubuntu Server	Free and LTS-supported server version of Ubuntu.	Cloud-init, Snap support, MAAS, Landscape	Cloud, containers, CI/CD
CentOS Stream	Rolling release preview of RHEL.	Continuous updates, close to RHEL ABI	RHEL development/testing
AlmaLinux / Rocky Linux	RHEL-compatible community distros (post-CentOS 8).	Binary-compatible with RHEL	Enterprises needing free RHEL alternative
Amazon Linux	AWS-optimized distro, based on Fedora/RHEL.	Tight AWS integration, preconfigured for cloud	EC2 instances, AWS-native apps

Desktop Distributions

Distribution	Description	Key Features	Use Cases
Ubuntu Desktop	Popular desktop distro, user-friendly.	GNOME, Snap Store, live USB support	Beginners, developers
Linux Mint	Ubuntu-based distro with a Windows-like UI.	Cinnamon desktop, multimedia codecs included	Windows migrants, home use
Fedora Workstation	Developer-focused desktop with latest GNOME.	Flatpak, Wayland, bleeding edge kernel	Developers, testers
Zorin OS / Elementary OS	Ubuntu-based with modern, clean UI.	Mac/Windows-like desktop environments	Casual users, design-focused workflows

Security-Focused Distributions

Distribution	Description	Key Features	Use Cases
Kali Linux	Debian-based, built for penetration testing.	Metasploit, Nmap, over 600 tools	Ethical hacking, security audits
Parrot OS	Lightweight, secure distro with privacy features.	Tor, anonymization tools, sandboxing	Privacy-focused users, hackers
Qubes OS	Security-by-isolation using Xen-based VMs.	Compartmentalization, Qubes Manager	High-security environments
Tails OS	Live distro focused on anonymity and privacy.	Tor by default, no persistence	Whistleblowers, journalists

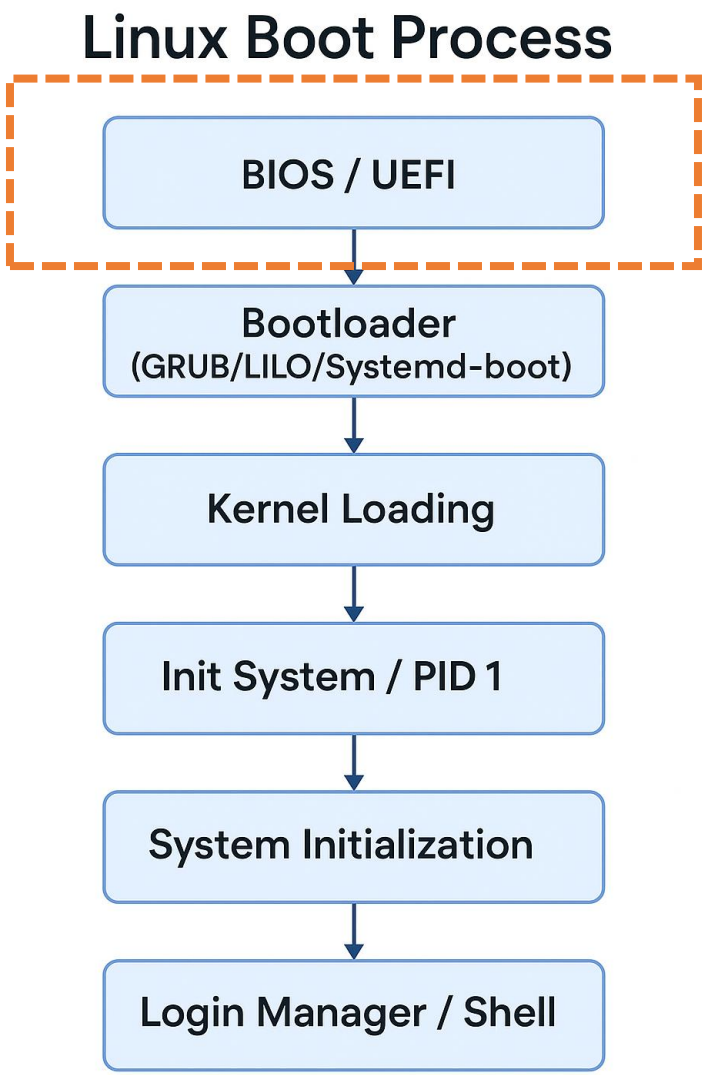
Developer & Rolling-Release Distros

Distribution	Description	Key Features	Use Cases
Arch Linux	Minimal, rolling release. DIY setup.	Pacman, AUR, systemd	Advanced users, custom setups
Manjaro	Arch-based with user-friendly installer.	GUI tools, preconfigured desktop	Intermediate users
Gentoo	Source-based distro for complete control.	Portage, compile-everything	Performance tuning, embedded systems
Void Linux	Independent distro with runit init system.	Musl/libc options, minimalist	Lightweight systems, advanced users

Linux Boot Process

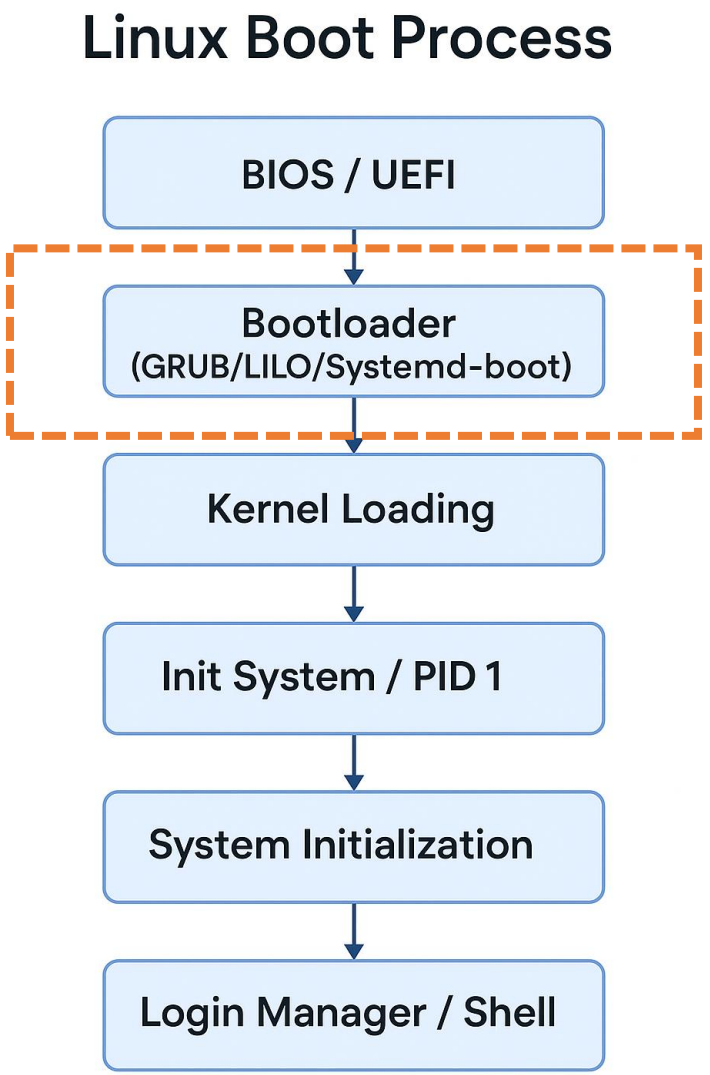
Linux Boot Process [1]

- The Linux boot process follows a **well-defined sequence**, transitioning from hardware to a fully running OS:
  - **Step 1: BIOS / UEFI Initialization**
  - **Role**: Initializes hardware and performs POST (Power-On Self-Test).
  - **Details**: Checks CPU, RAM, disk controllers.
  - **UEFI**: More modern than BIOS, supports Secure Boot.



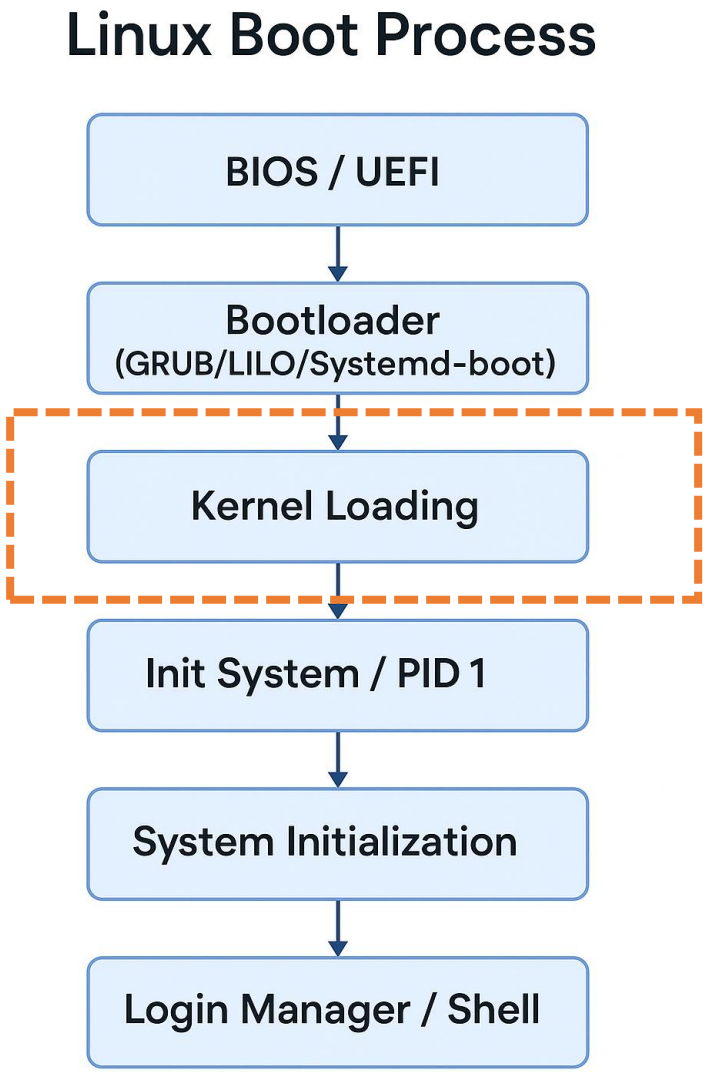
Linux Boot Process [2]

- The Linux boot process follows a **well-defined sequence**, transitioning from hardware to a fully running OS:
  - **Step 2: Bootloader (GRUB/LILO/Systemd-boot)**
  - **Role**: Loads the Linux kernel into memory.
  - **GRUB** (GNU GRUB) is the most common bootloader.
  - Can load multiple OS kernels or kernel versions.



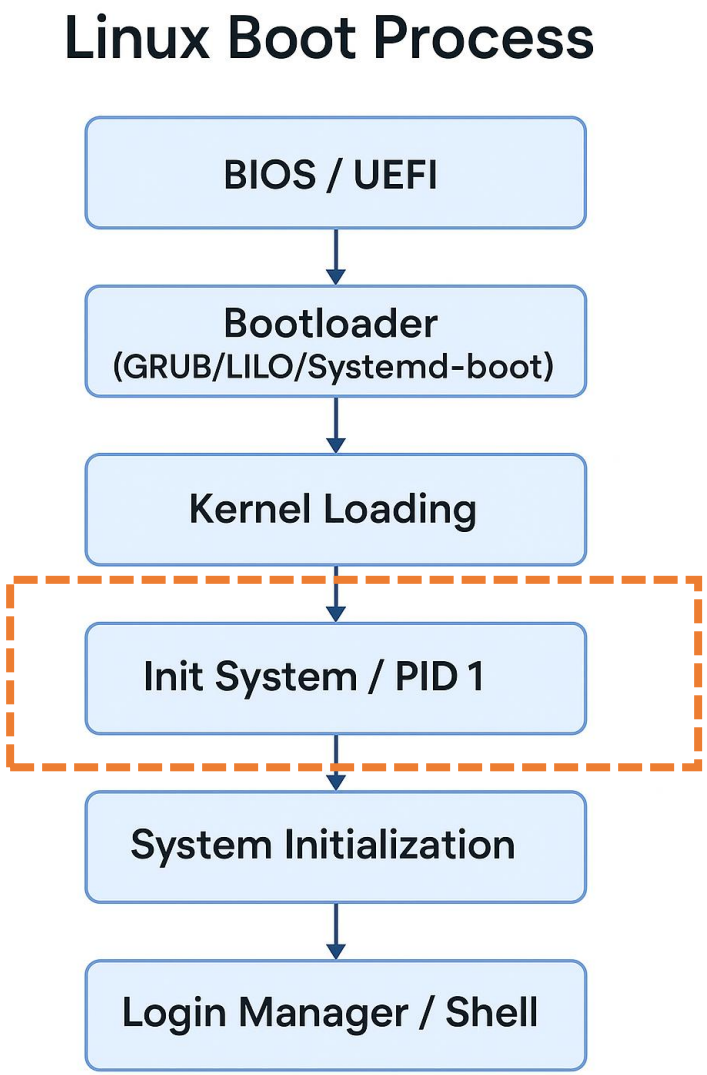
Linux Boot Process [3]

- The Linux boot process follows a **well-defined sequence**, transitioning from hardware to a fully running OS:
  - **Step 3: Kernel Loading**
  - **Role**: The kernel is uncompressed and initialized.
  - **Initial RAM disk (initrd/initramfs)** is loaded to mount essential filesystems and drivers.
  - Detects hardware, sets up memory and device trees.



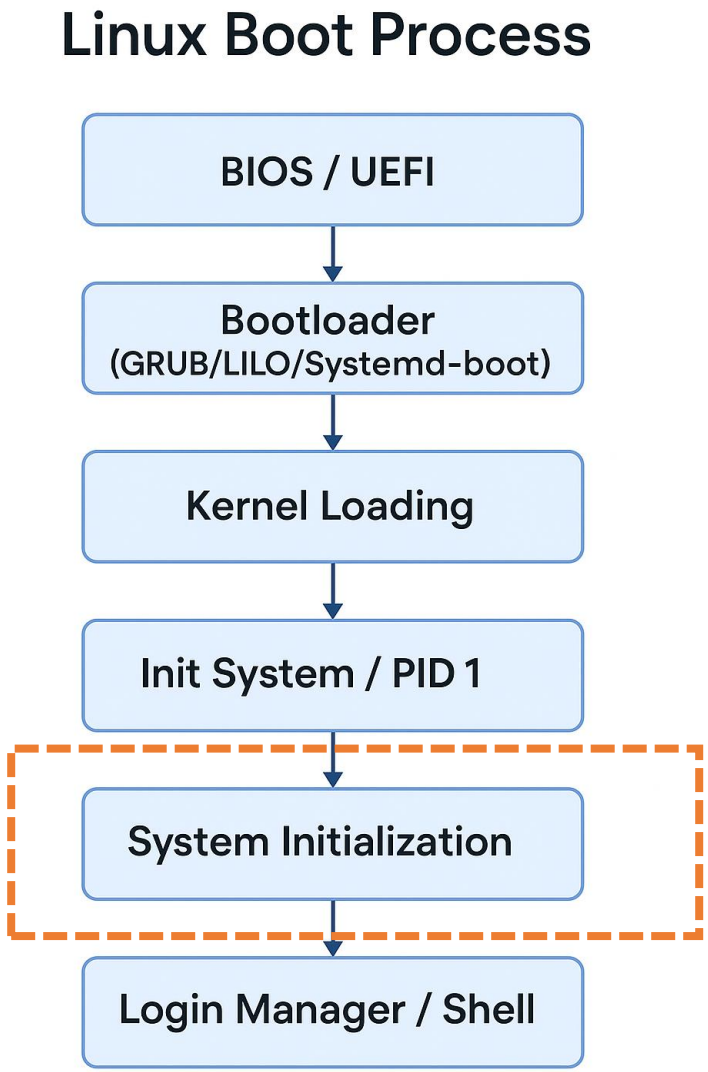
Linux Boot Process [4]

- The Linux boot process follows a **well-defined sequence**, transitioning from hardware to a fully running OS:
  - **Step 4: Init System / PID 1**
  - **Role**: Hands control from kernel to user space.
  - **Common Init Systems**:
    - systemd (modern distros: RHEL 7+, Ubuntu 15+)
    - SysVinit (older systems)
    - Upstart (older Ubuntu versions)



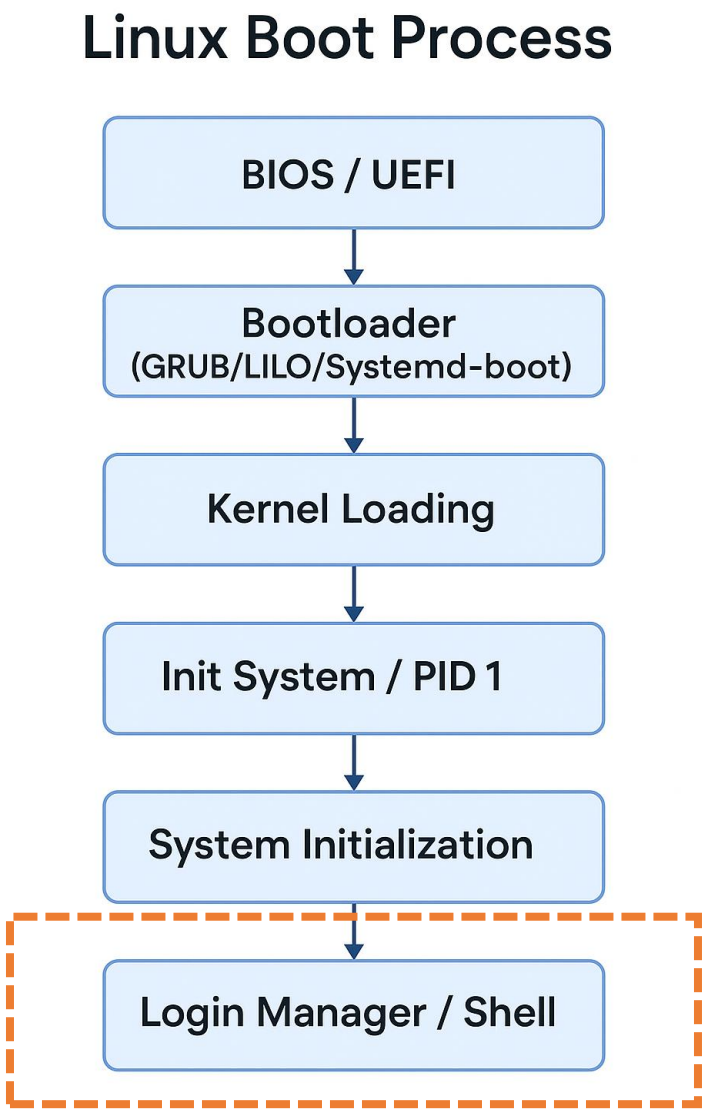
Linux Boot Process [5]

- The Linux boot process follows a **well-defined sequence**, transitioning from hardware to a fully running OS:
  - **Step 5: System Initialization**
  - **Role**: Init system launches:
    - Mounts file systems
    - Sets hostname
    - Loads kernel modules
    - Starts background services (daemons)
    - Enables networking



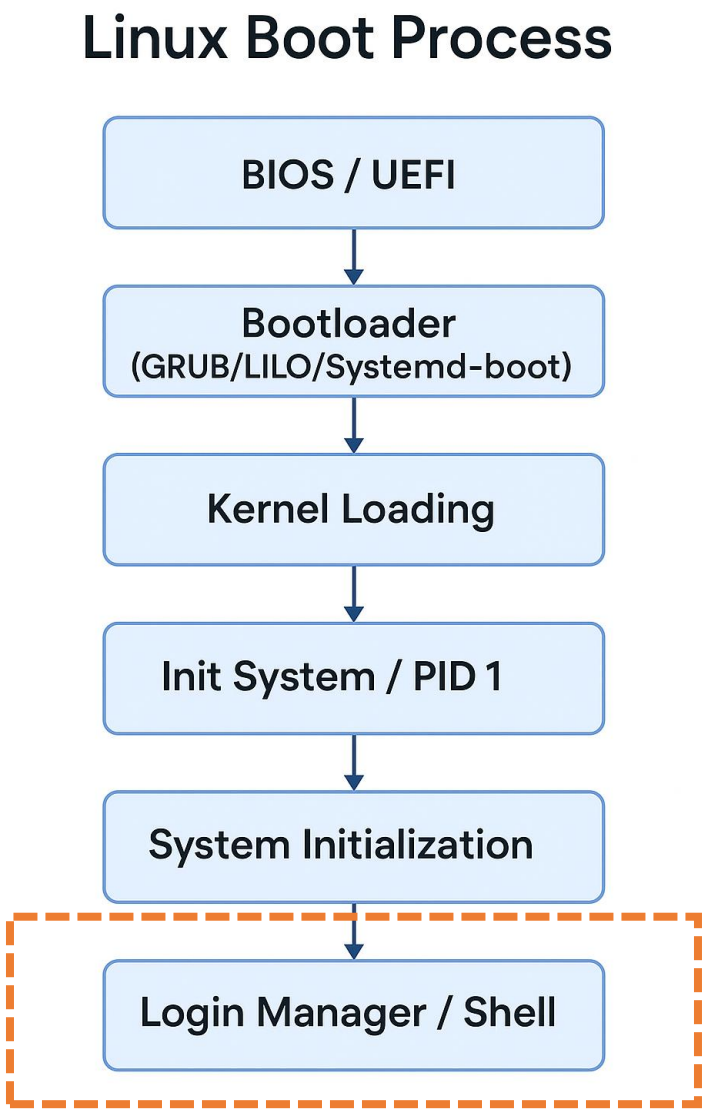
Linux Boot Process [6]

- The Linux boot process follows a **well-defined sequence**, transitioning from hardware to a fully running OS:
  - **Step 6: Login Manager / Shell**
  - **CLI systems**: getty spawns a login prompt
  - **GUI systems**: Display managers like GDM, LightDM start the graphical environment



Linux Boot Process [6]

- The Linux boot process follows a **well-defined sequence**, transitioning from hardware to a fully running OS:
  - **Step 6: Login Manager / Shell**
  - **CLI systems**: getty spawns a login prompt
  - **GUI systems**: Display managers like GDM, LightDM start the graphical environment



Linux System Architecture

Linux System Architecture [1]

- The **Linux operating system** is built using a **modular architecture** that separates core system functions into distinct layers. This layered model allows Linux to be secure, flexible, and portable across various hardware platforms.
- **Layered Architecture:**
  - Hardware Layer
  - Kernel Layer (Linux Kernel)
  - System Libraries
  - System Utilities
  - Shell (Command-Line Interface)
  - User Applications

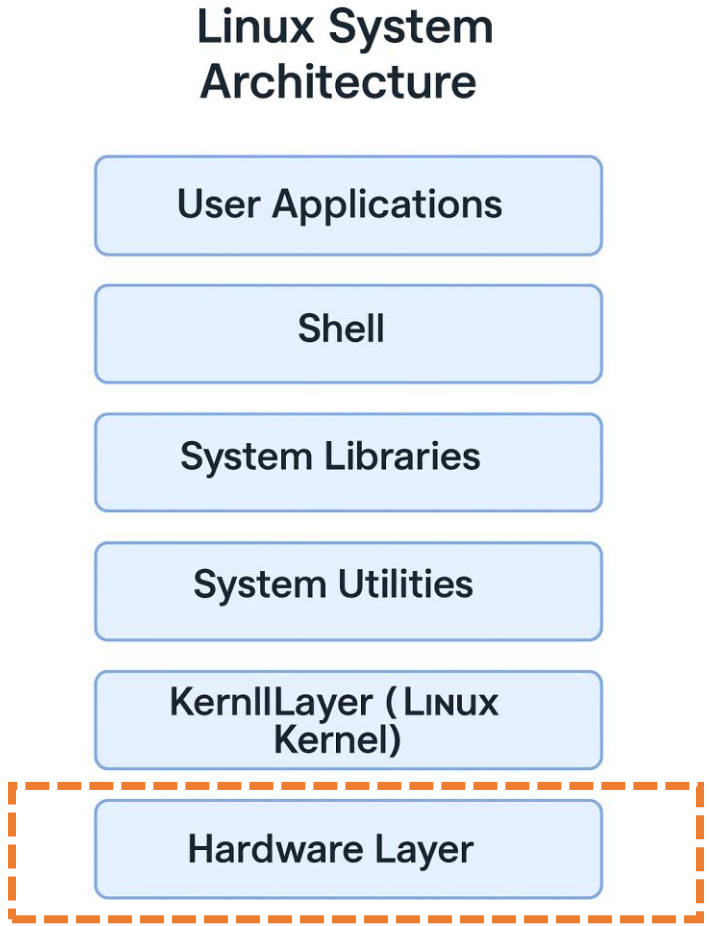
Linux System Architecture [2]

❑Hardware Layer

- This is the physical component of the computer system. It includes:
- CPU (Central Processing Unit)
- RAM (Memory)
- Hard drives
- Network interfaces
- Peripheral devices

❑Role in the OS:

The Linux kernel interacts directly with this layer using device drivers to manage and control the hardware.



Linux System Architecture [3]

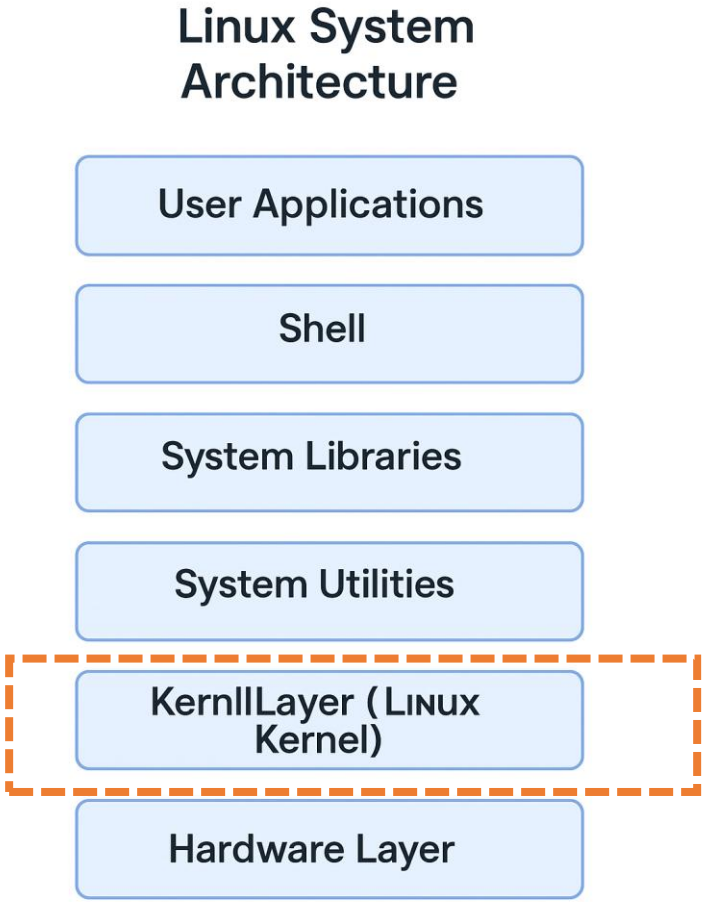
❑Kernel Layer

- The **kernel** is the heart of the Linux operating system. It's responsible for managing communication between hardware and software.

❑Key Responsibilities:

- **Process Management:** Scheduling and executing processes
- **Memory Management:** Allocation and deallocation of RAM
- **Device Management:** Managing device drivers and hardware access
- **File System Management:** Reading/writing to storage
- **Networking:** Managing data transmission over networks
- **Security & Access Control:** Enforcing user permissions and isolation (e.g., SELinux, namespaces)

❑The Linux kernel is a **monolithic kernel**, meaning all core services run in a single address space but with the ability to load/unload modules dynamically.



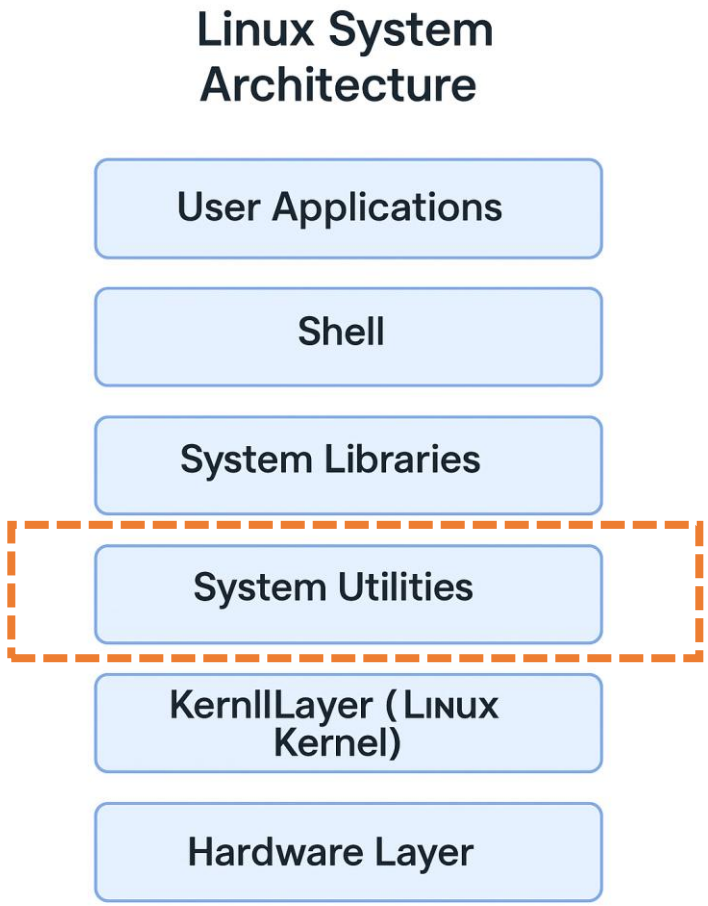
Linux System Architecture [4]

❑ **System Utilities**

❑ These are essential software tools and programs that manage the Linux system. They can be:

- **Core utilities**: ls, cp, mv, mkdir, etc.
- **Service managers**: systemd, init
- **File systems**: mount, umount, fsck
- **Networking tools**: ip, ping, netstat

❑ Some of these run in the background as **daemons** (e.g., sshd, cron) and handle tasks like scheduled jobs, logging, and network connections.



Linux System Architecture [5]

❑ **System Libraries**

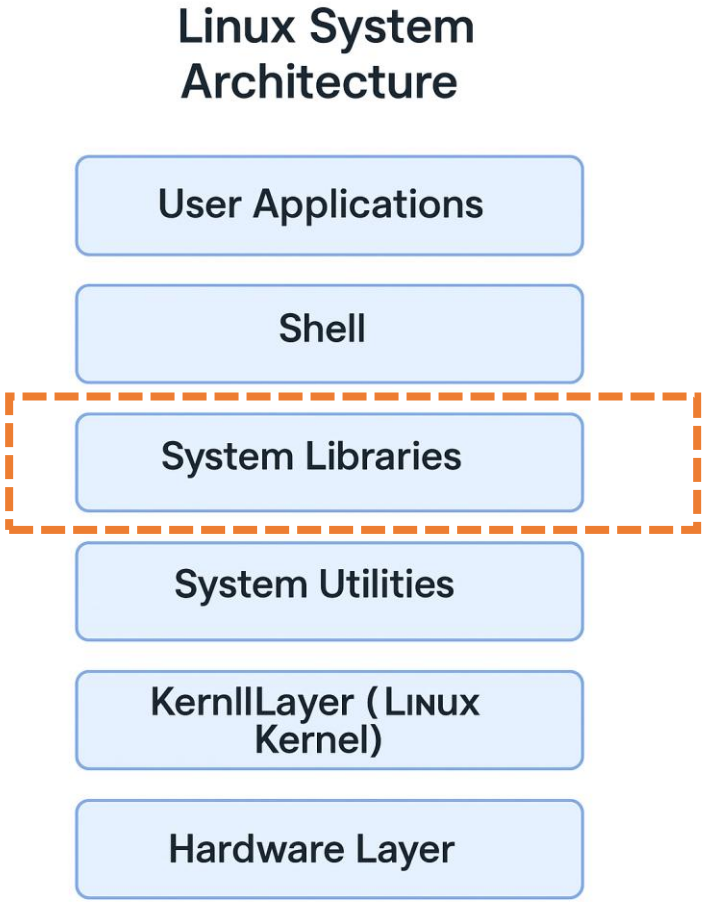
❑ These are special functions or programs that provide the interface for user applications to interact with the kernel.

❑ **Example:**

- glibc (GNU C Library) – handles standard functions like input/output, memory allocation, and string operations.

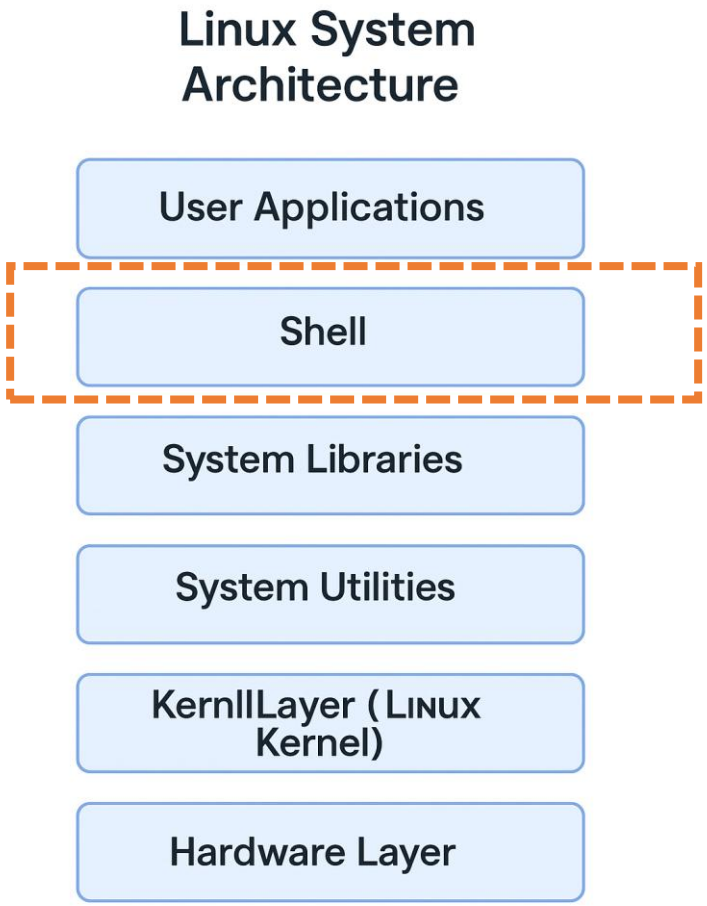
❑ These libraries **abstract system calls** so that developers don't have to write low-level code to interact with hardware or the kernel directly.

.



Linux System Architecture [6]

- **Shell (Command-Line Interface)**
- The shell acts as a **command interpreter**. It reads user input and translates it into commands the kernel can understand via system libraries.
- **Common Shells:**
  - bash (Bourne Again Shell)
  - zsh (Z Shell)
  - sh (original Bourne Shell)
  - fish (Friendly Interactive Shell)
- The shell allows scripting, automation, and direct control of system functions.



Linux System Architecture [7]

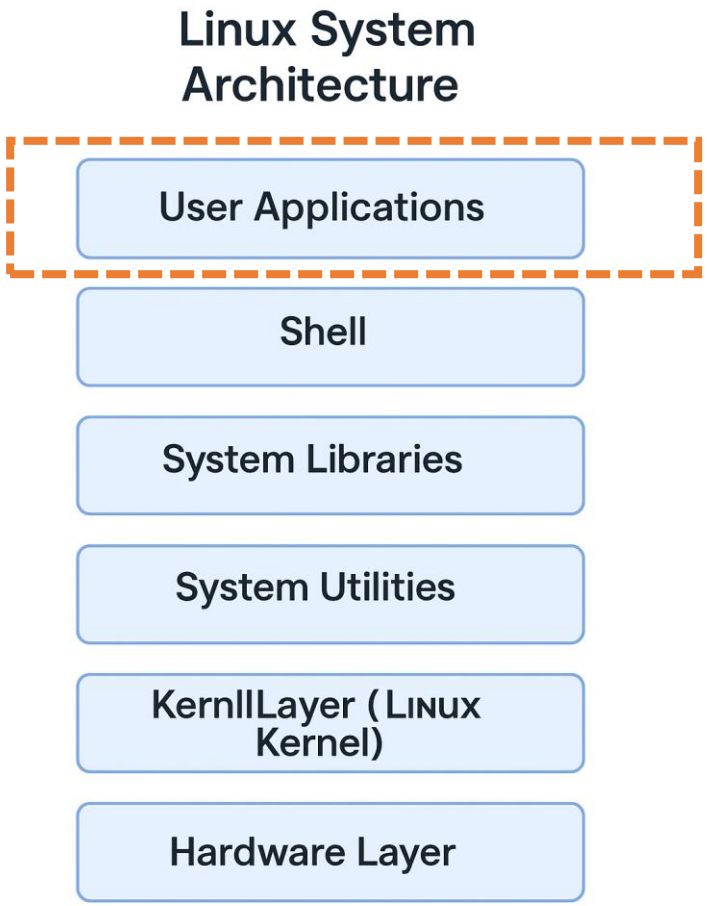
❑ **User Applications**

❑ These are the programs users interact with directly. They operate in **user space** (not kernel space) and perform tasks through system calls.

❑ **Examples:**

- Text editors: nano, vim, gedit
- Browsers: Firefox, Chromium
- Terminals: gnome-terminal, xterm
- Package managers: apt, dnf, pacman

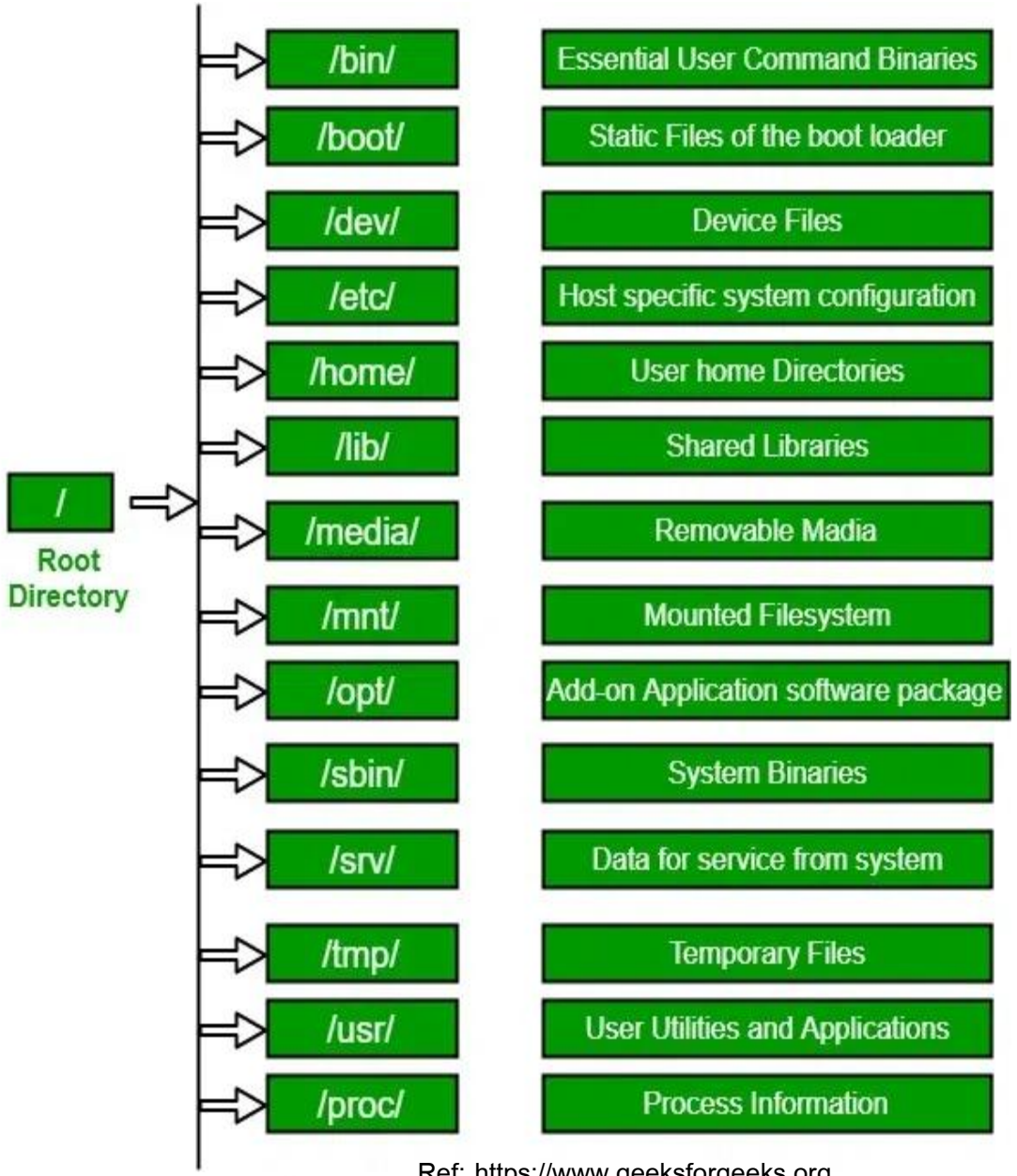
❑ Applications **do not access hardware directly** — they rely on the kernel via libraries and system utilities.



Filesystem hierarchy

Filesystem hierarchy [1]

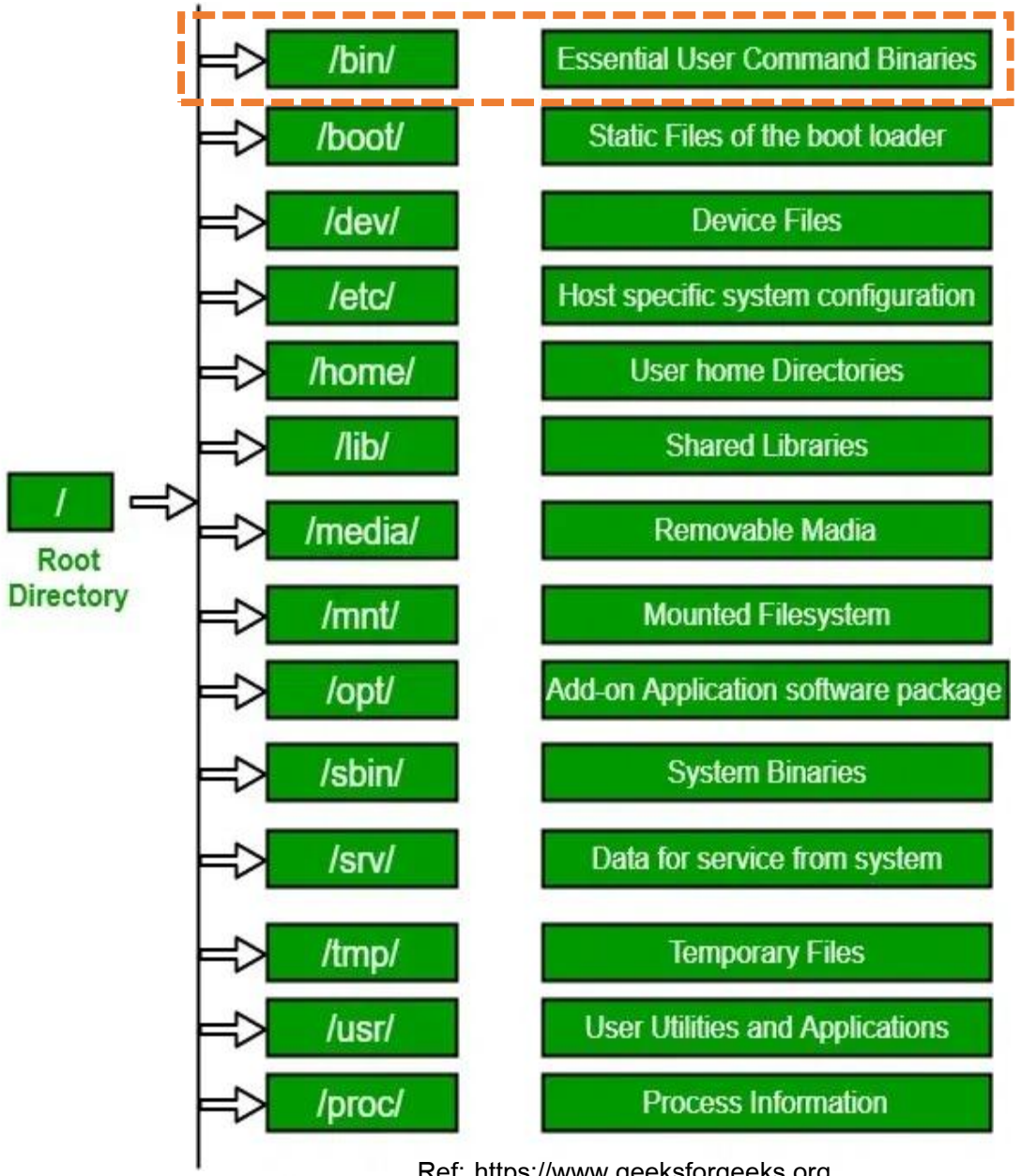
- The **Linux File Hierarchy Structure** or the **Filesystem Hierarchy Standard (FHS)** defines the directory structure and directory contents in Unix-like operating systems. It is maintained by the Linux Foundation.



Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [2]

- **/bin :**
- The /bin directory contains essential commands and binaries needed by all users, including cp, ls, ssh, and kill. These commands are universally available across user types.
  - Contains binary executables.
  - Common linux commands you need to use in single-user modes are located under this directory.
  - Commands used by all the users of the system are located here e.g. ps, ls, ping, grep, cp



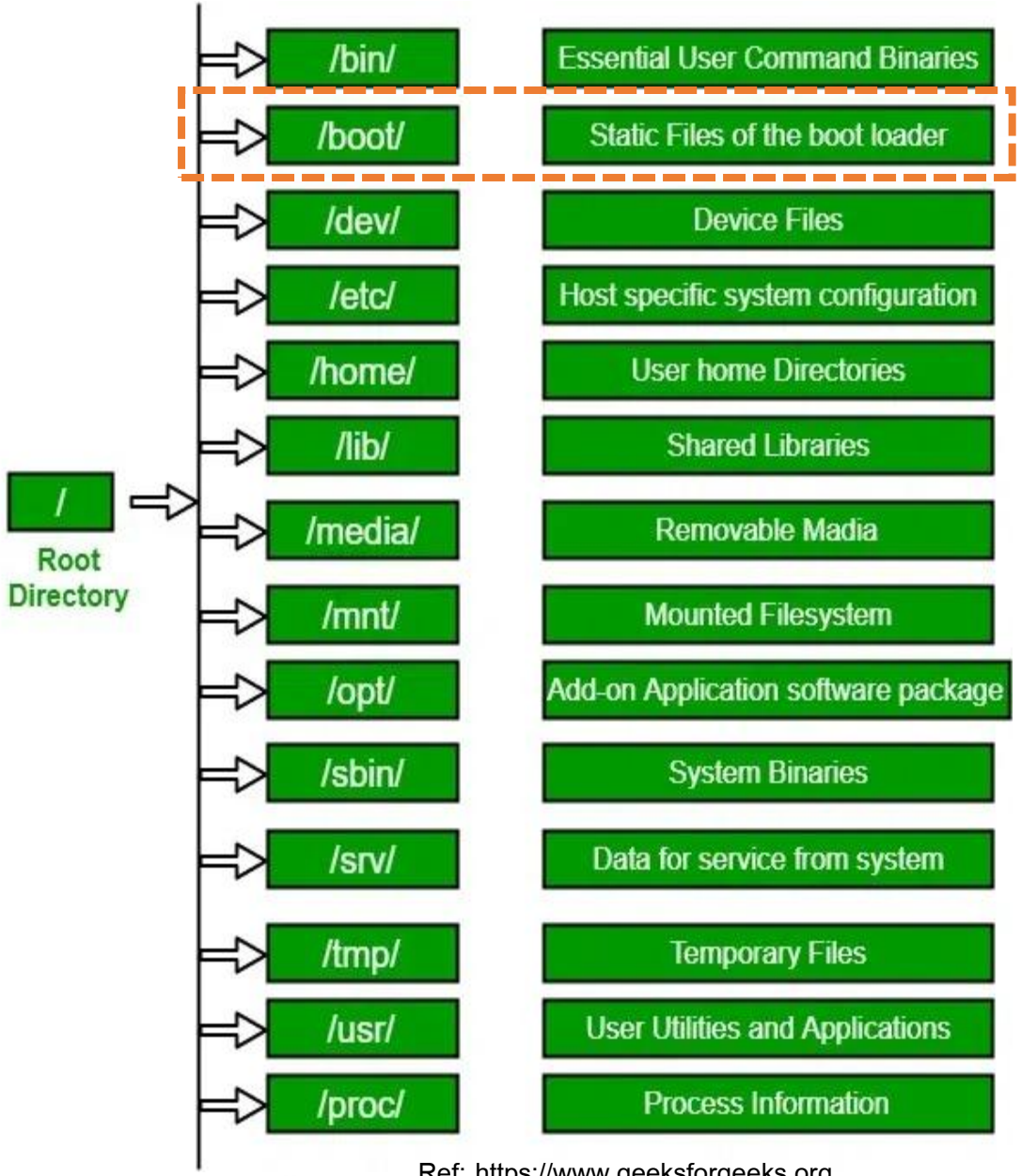
Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [3]

❑**/boot :**

❑This directory stores all files required for booting the system. It includes the GRUB bootloader configuration and essential kernel files that are loaded during startup.

- Kernel initrd, vmlinux, grub files are located under /boot
- Example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic



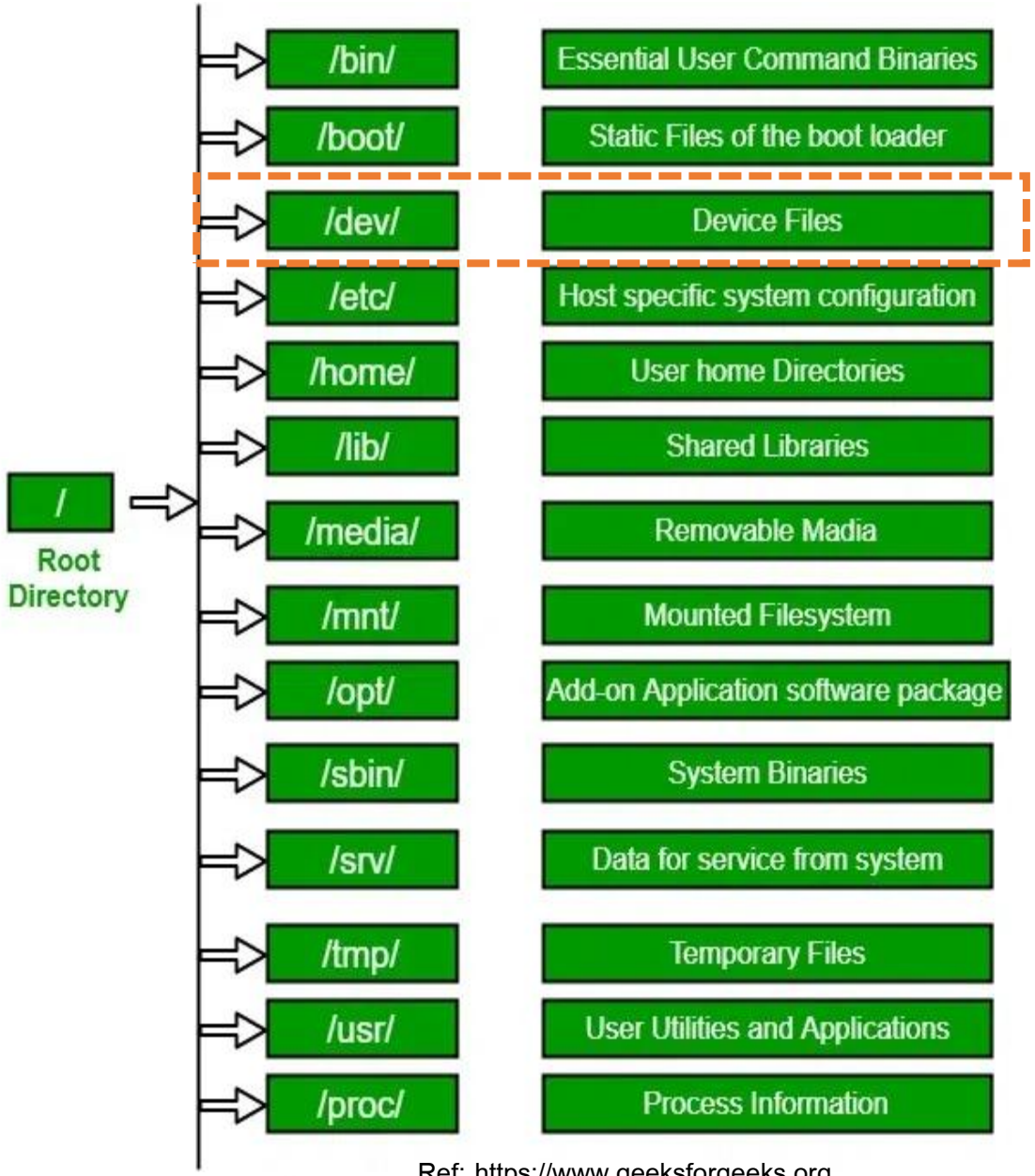
Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [4]

❑ **/dev :**

❑Device files in Linux are stored in the /dev directory. These are special files that act as interfaces between hardware and software. Device files are of two types: **block devices** (e.g., hard drives) and **character devices** (e.g., microphones and speakers). Examples include /dev/sda1 for disk partitions.

- These include terminal devices, usb, or any device attached to the system.
- Example: /dev/tty1, /dev/usbmon0



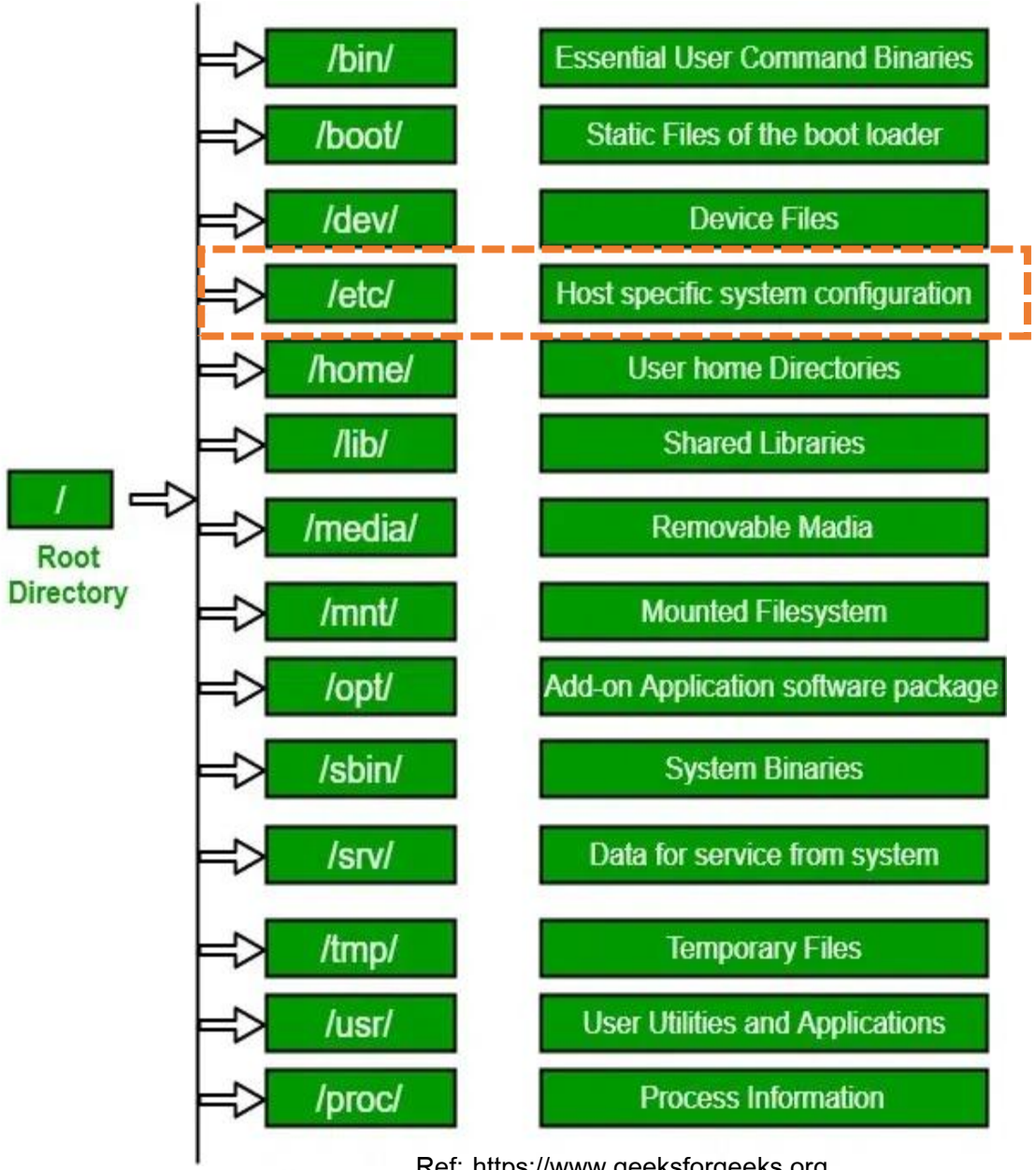
Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [5]

❑ **/etc :**

❑ Short for "Editable Text Configuration," /etc contains configuration files for system applications, users, services, and tools or it contains the Host-specific system-wide configuration files. For example, user details like UID and local addresses are defined here.

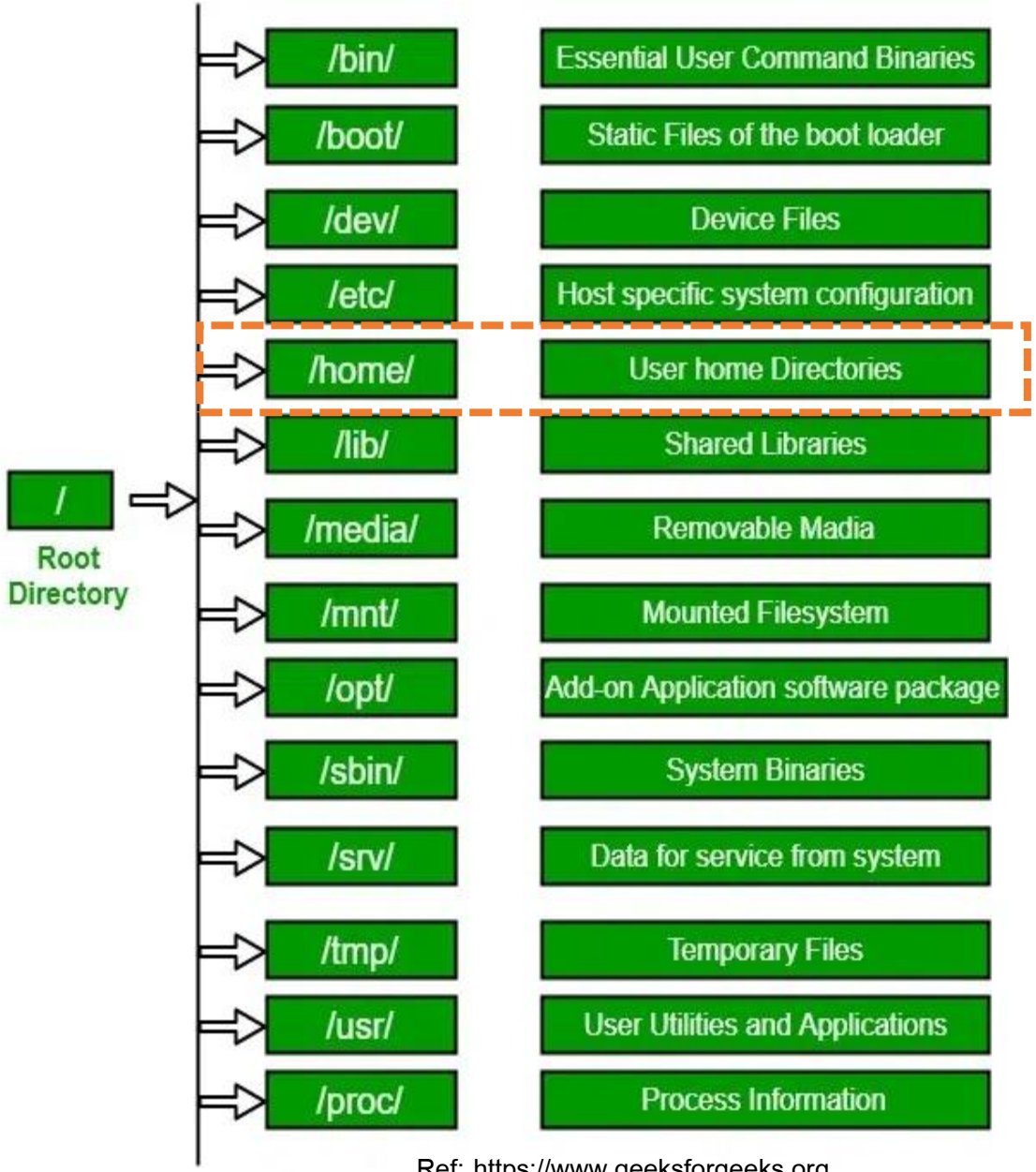
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- Example: /etc/resolv.conf, /etc/logrotate.conf.



Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [6]

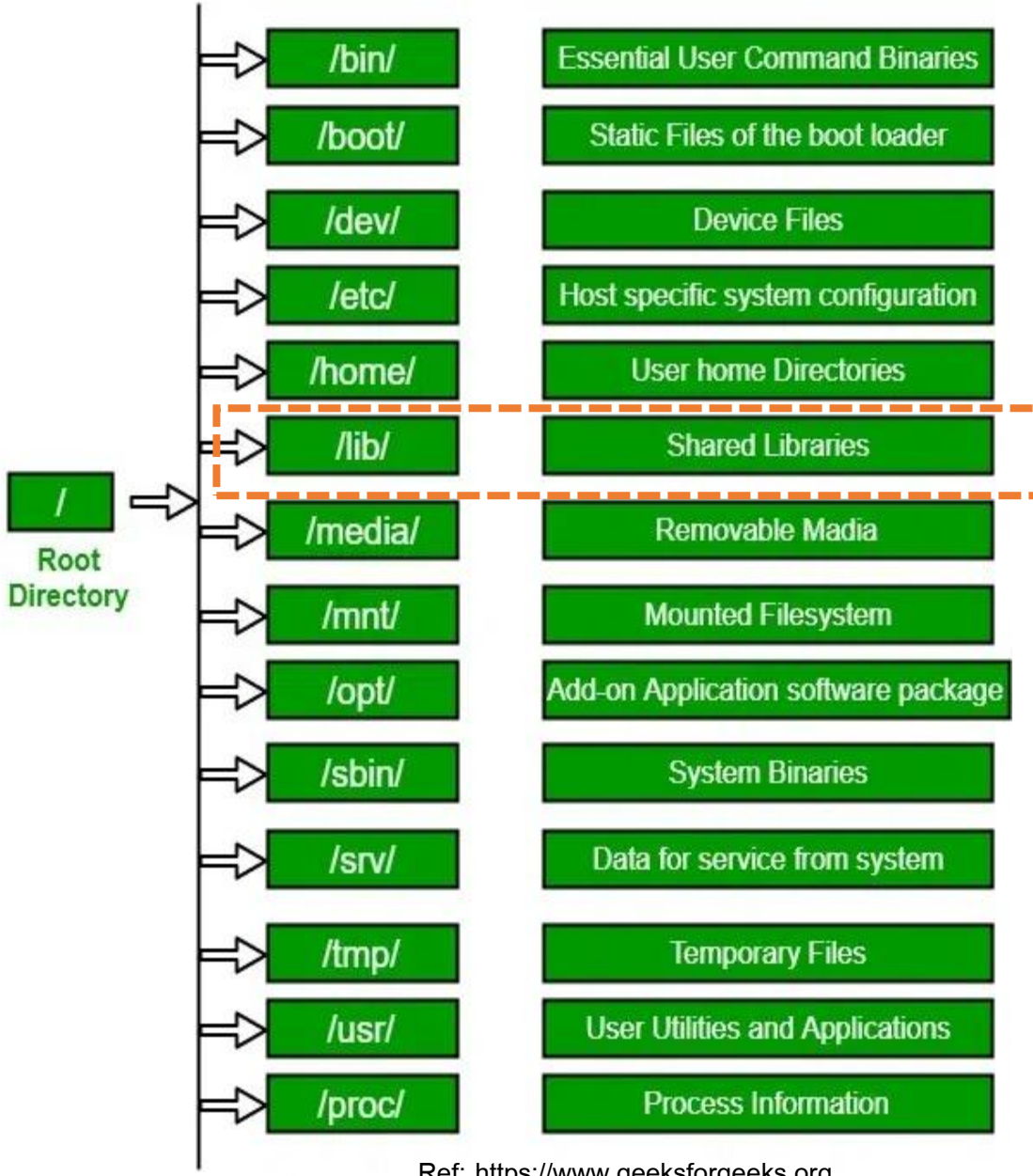
- **/home :**
- Every non-root user has a personal directory inside /home. For example, if your username is anshu, your personal directory would be /home/anshu. Each user can create, delete, or modify files only in their own home directory and cannot access others' home directories.
  - Home directories for all users to store their personal files, containing saved files, personal settings, etc..
  - example: /home/kishlay, /home/kv



Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [7]

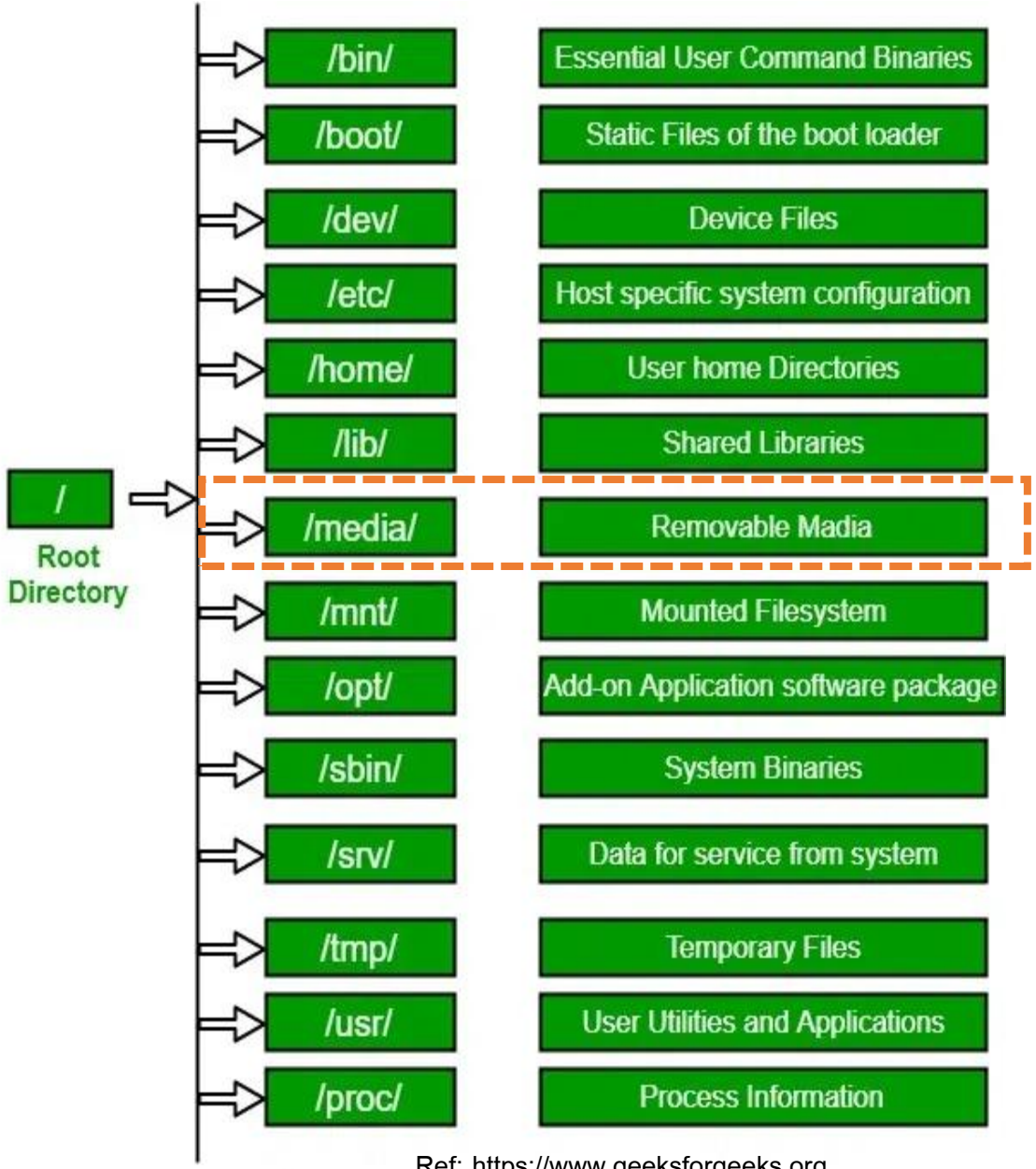
- **/lib:**
- Applications require shared libraries to run, which are stored in /lib. These include dynamic libraries needed during runtime. For example, Apache server libraries are available here.
  - Library filenames are either ld* or lib*.so.*
  - Example: ld-2.11.1.so, libncurses.so.5.7



Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [8]

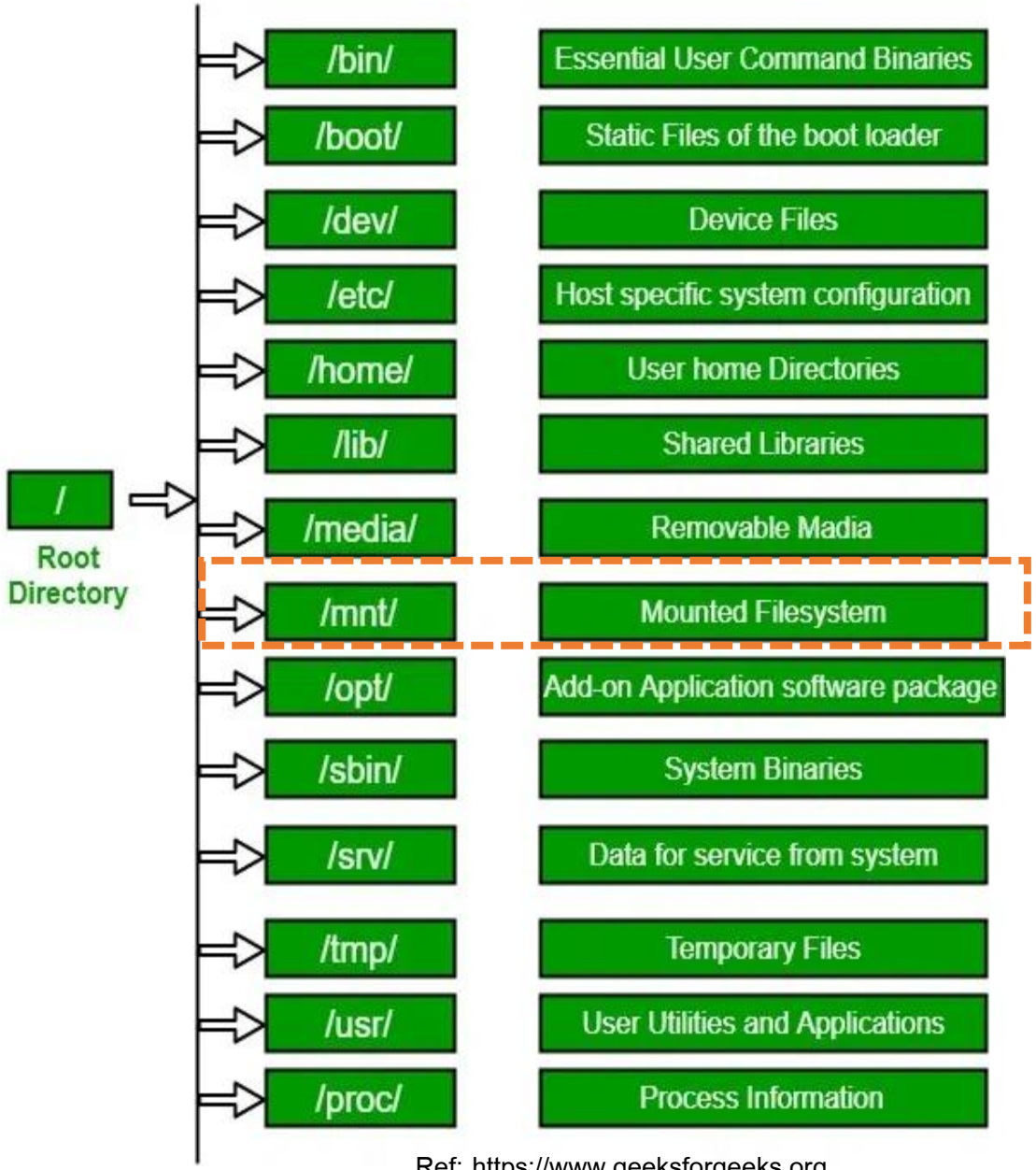
- **/media:**
- Devices like USBs, CDs, and pen drives are mounted under /media. For example, when a CD-ROM is inserted (appeared in FHS-2.3), its details will appear here.
  - Temporary mount directory for removable devices.
  - Examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer



Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [9]

- **/mnt :**
- When external drives are connected, they are temporarily mounted in /mnt. This is where their contents become accessible to the system.
  - Temporary mount directory where sysadmins can mount filesystems



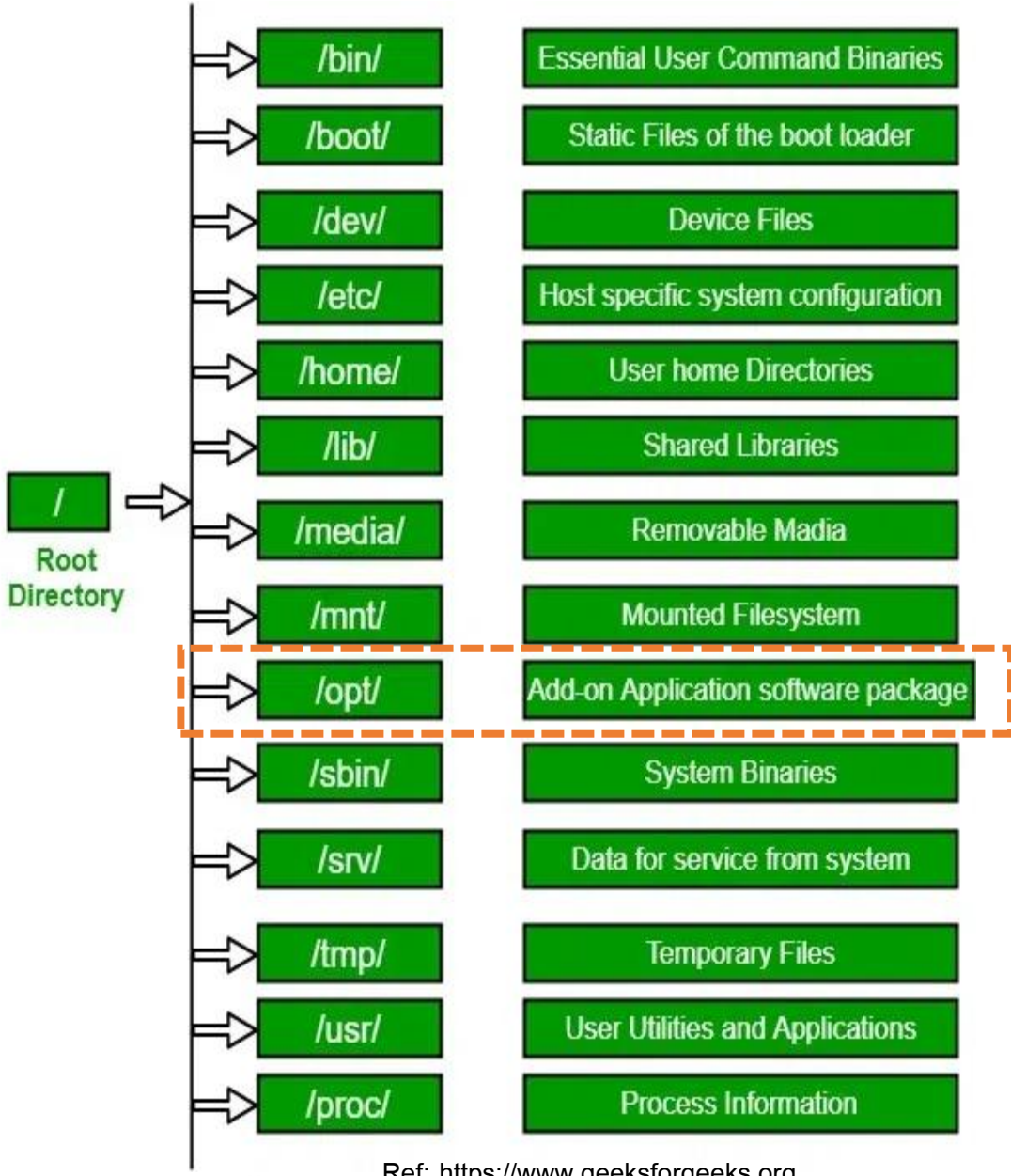
Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [10]

❑**/opt:**

❑Third-party software and packages not part of the default system installation are stored in /opt. It includes their configuration and data files.

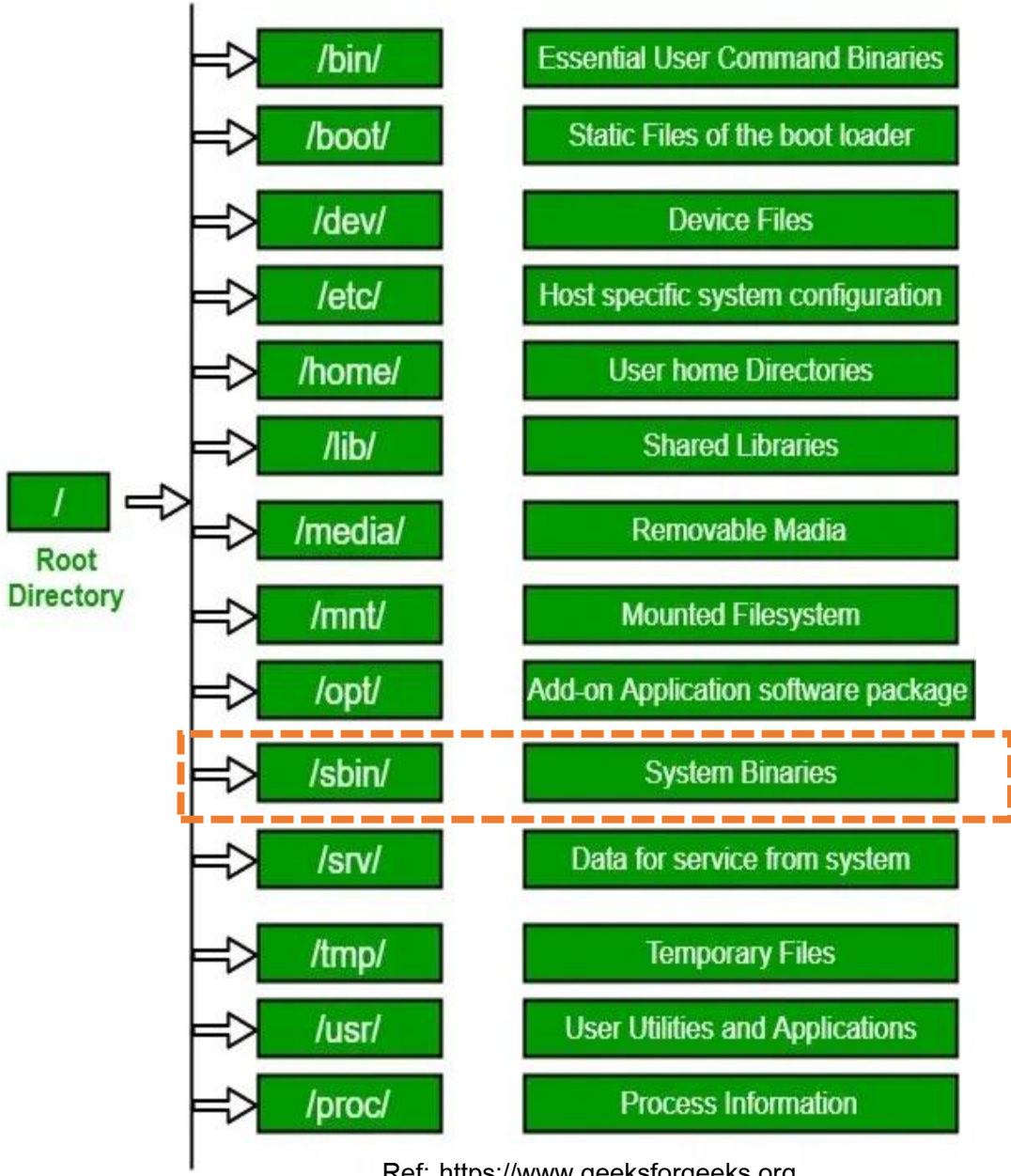
- Contains add-on applications from individual vendors.
- Add-on applications should be installed under either /opt/ or /opt/ sub-directory.



Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [11]

- **/sbin :**
- Essential system binaries, e.g.,
- This directory holds administrative binaries like iptables, firewall management tools, fsck, init, route etc. These binaries are primarily for system administrators and typically require root privileges to execute.
  - Just like /bin, /sbin also contains binary executables.
  - The linux commands located under this directory are used typically by system administrators, for system maintenance purposes.
  - Example: iptables, reboot, fdisk, ifconfig, swapon



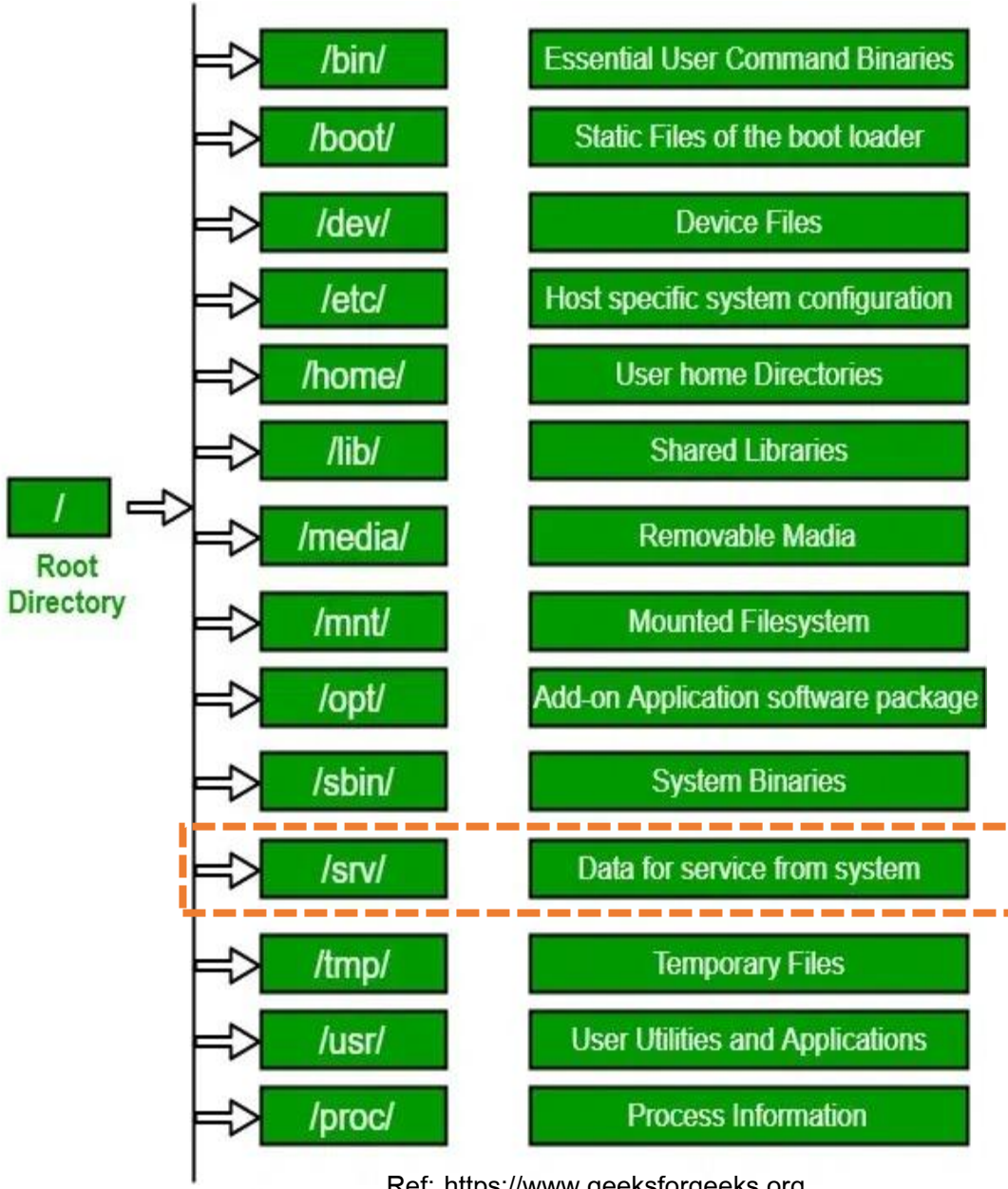
Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [12]

❑**/srv :**

❑Site-specific data served by this system, such as data and scripts for web servers, data offered by FTP servers, and repositories for version control systems.

- srv stands for service.
- Contains server specific services related data.
- Example, /srv/cvs contains CVS related data.



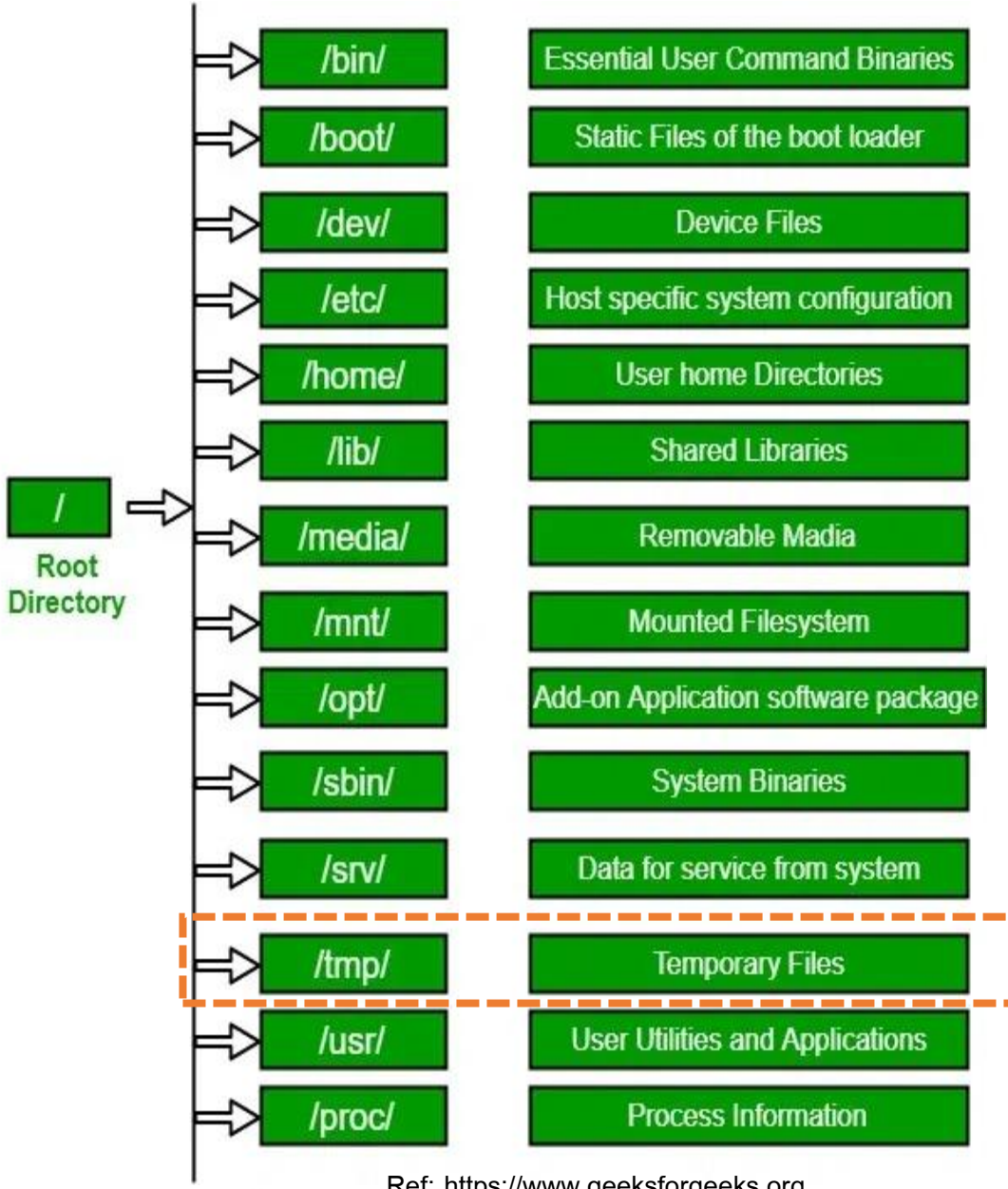
Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [13]

❑**/tmp :**

❑Programs create temporary files during execution, and these are stored in /tmp. These files are deleted automatically after the program finishes or when the system is restarted.

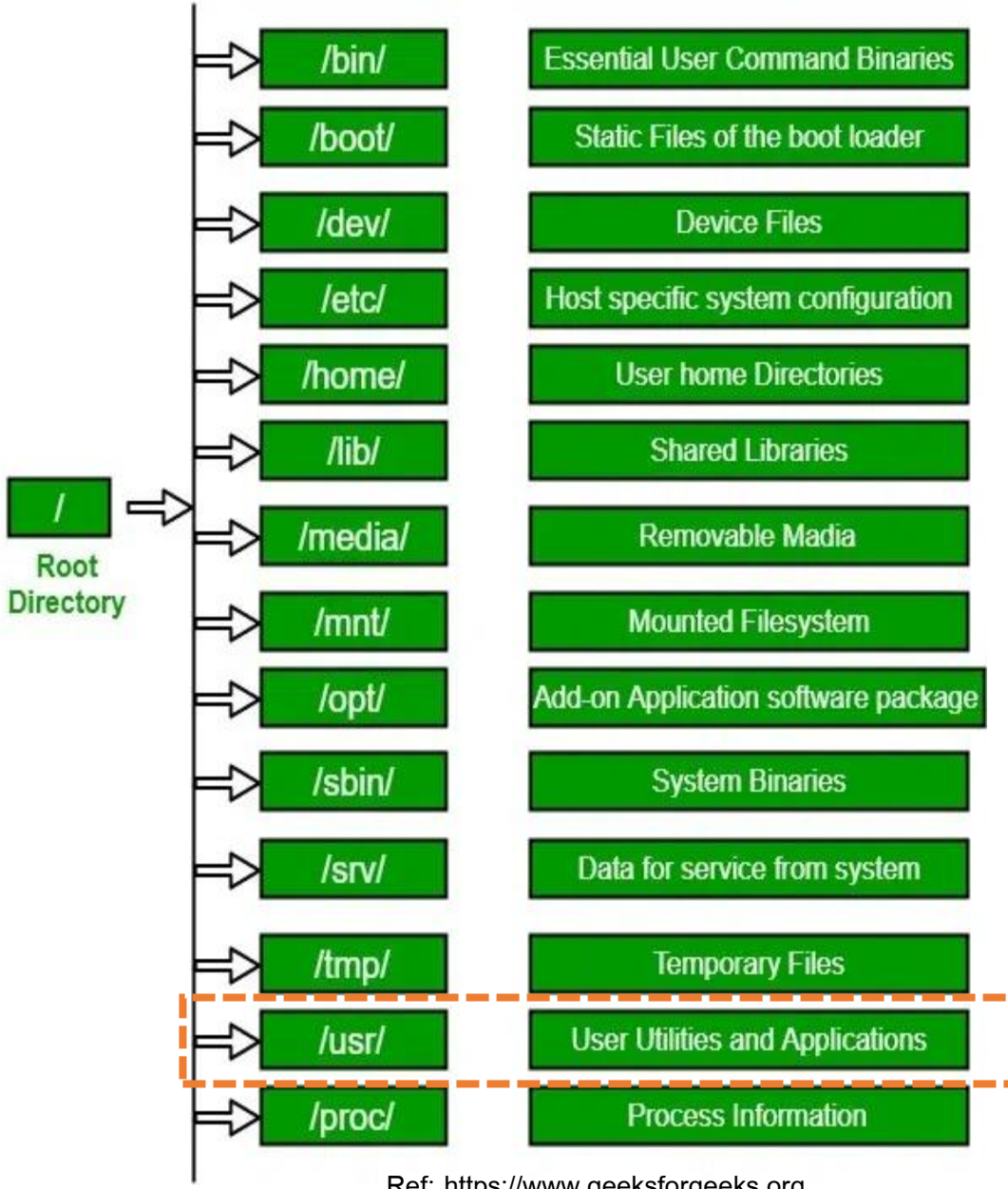
- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when the system is rebooted.



Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [14]

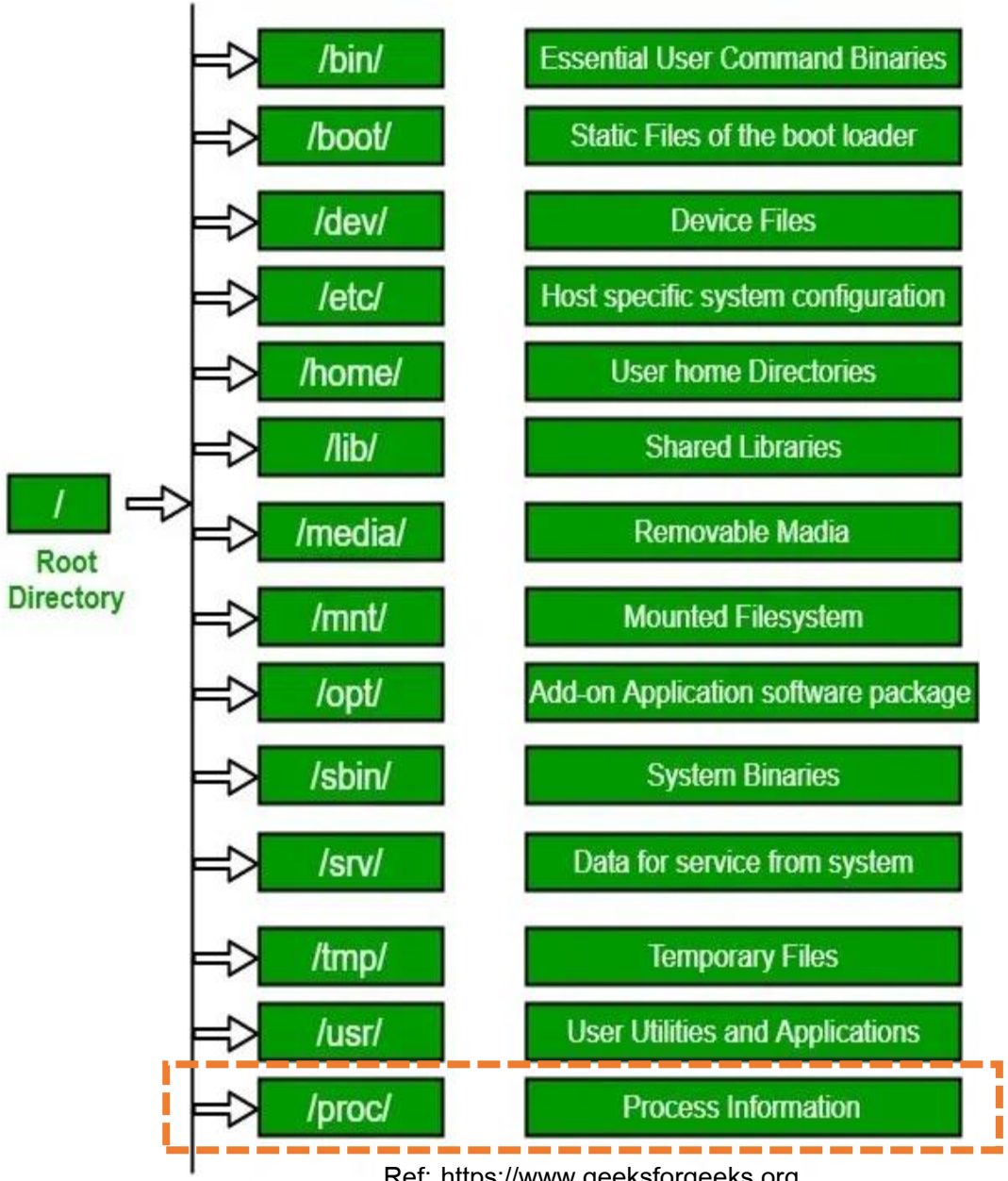
- **/usr :**
- Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications.
  - Contains binaries, libraries, documentation, and source-code for second level programs.
  - /usr/bin contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
  - /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
  - /usr/lib contains libraries for /usr/bin and /usr/sbin
  - /usr/local contains user's programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2
  - /usr/src holds the Linux kernel sources, header-files and documentation.



Ref: https://www.geeksforgeeks.org

Filesystem hierarchy [15]

- **/proc:**
- The /proc directory provides detailed information about system processes. Each process is assigned a unique ID and represented as a directory inside /proc. For example, /proc/meminfo gives real-time data about memory usage including total, free, buffer, and cache statistics.
  - Contains information about system process.
  - This is a pseudo filesystem that contains information about running processes. For example: /proc/{pid} directory contains information about the process with that particular pid.
  - This is a virtual filesystem with text information about system resources. For example: /proc/uptime



Ref: https://www.geeksforgeeks.org

Basic Command-Line Tools

Basic Command-Line Tools [1]

- In Linux, the **command-line interface (CLI)** is a powerful way to interact with the system using text-based commands. These tools allow users to **navigate, manage files, inspect the system, and automate tasks**.
- **Most commonly used basic command-line tools**:
  - ls, cd, cat, grep, less
  - echo, touch, mkdir, rm, cp
  - mv, clear

Basic Command-Line Tools [2]

- ls – List Directory Contents
  - ls # List files in the current directory
  - ls -l # Long listing (permissions, owner, size, etc.)
  - ls -a # Show hidden files (starting with .)

- cd – Change Directory
  - cd /home/user # Change to a specific directory
  - cd .. # Move up one level
  - cd ~ # Go to home directory

Basic Command-Line Tools [3]

- cat – Concatenate and Display File Content
  - cat file.txt # View contents of a file
  - cat file1 file2 > all # Merge two files into one

- grep – Search Text Using Patterns
  - grep "error" logfile.txt # Search for "error" in file
  - grep -i "error" logfile.txt # Case-insensitive search
  - grep -r "todo" /project # Recursively search in dir

Basic Command-Line Tools [4]

- less – View File Content One Page at a Time
  - less largefile.log # View file with navigation (PgUp, PgDn, /search)
- echo – Display a Line of Text
  - echo "Hello, world!" # Print a string
  - echo $HOME # Show the value of a variable
- touch – Create an Empty File / Update Timestamp
  - touch file.txt # Create empty file
  - touch file{1..3}.txt # Create file1.txt, file2.txt, file3.txt

Basic Command-Line Tools [5]

- mkdir – Create Directories
  - mkdir newdir # Create a directory
  - mkdir -p dir1/dir2 # Create nested directories

- rm – Remove Files and Directories
  - rm file.txt # Delete file
  - rm -r dir/ # Delete directory and contents
  - rm -f file.txt # Force delete (no prompt)

Basic Command-Line Tools [6]

- cp – Copy Files and Directories
  - cp file1.txt file2.txt # Copy file
  - cp -r dir1 dir2 # Copy directory recursively
- mv – Move or Rename Files
  - mv oldname.txt newname.txt # Rename file
  - mv file.txt /tmp/ # Move file to /tmp
- clear – Clear the Terminal Screen
  - clear # Clears everything visible in terminal

Power Tools

Tool	Description	Example
head	View the **first lines** of a file	head -n 10 file.txt
tail	View the **last lines** of a file	tail -f /var/log/syslog (live view)
wc	Word/line/char count	wc -l file.txt
sort	Sort lines in a file	sort names.txt
uniq	Remove duplicate lines	`sort file
find	Search for files	find /etc -name "*.conf"
man	View manual pages	man ls
history	Show command history	`history

Demo

Navigating the Filesystem [1]

- **Navigating the Filesystem**
- **Goal:** Practice moving between directories, listing files, and understanding the structure.
- **Commands to Use:**
  - pwd – Show current directory
  - ls – List files
  - cd – Change directory
  - tree (optional) – Show directory tree

Navigating the Filesystem [2]

- **Start in your home directory:**
  - cd ~
  - pwd
- **List contents of the home directory:**
  - ls -l
- **Create and move into a new directory:**
  - mkdir linux_practice
  - cd linux_practice

Navigating the Filesystem [5]

- **Create some files and subdirectories:**
  - touch file1.txt file2.txt
  - mkdir notes
  - cd notes
  - touch todo.txt
  - cd ..

Navigating the Filesystem [6]

- **Navigate back and forth:**
- cd notes
- cd ..
- cd ..

Navigating the Filesystem [7]

- **Using man Pages**
- **Goal:** Understand how to get help for any command.
- Basic manual usage:
  - man ls
  - Scroll with ↑ ↓ or PgUp/PgDn
  - Search: press /, type a word, press Enter
  - Quit: q

Navigating the Filesystem [8]

- **Explore specific command options:**
  - man mkdir
- **Check file format documentation:**
  - man 5 passwd
- **Search for a command by keyword:**
  - man -k "copy"

Navigating the Filesystem [9]

- **Using --help and Shell help**
- Use these to get quick syntax help:
- Commands with --help:
  - cp --help
  - grep --help

Navigating the Filesystem [10]

- **Built-in shell commands (help command):**
  - help cd
  - help exit
- Distinguish between binary and built-in commands:
  - type cd
  - type ls

User and Permission Management

Agenda

❑User and Permission Management

- Managing users and groups (useradd, passwd, usermod, etc.)
- File and directory permissions (chmod, chown, umask)
- SUID, SGID, Sticky Bit explained
- sudo and privilege escalation
- **Hands-on:** Creating users, setting file permissions, sudo configuration

Managing Users and Groups

Managing users and groups [1]

- Managing users and groups is a core part of Linux system administration. This includes **creating, modifying, and deleting** users and groups, as well as **managing permissions** and access control.
- **Users and Groups in Linux**
- **Users**
  - Normal users: Have limited privileges (e.g., developers, admins).
  - Root user: Superuser with full system access.
  - System users: Used by services (e.g., nginx, mysql).

Managing users and groups [2]

❑Users and Groups in Linux

❑Groups

- Used to **group users** with common access needs.
- Users can belong to **multiple groups**.
- **Primary group**: Assigned on user creation.
- **Supplementary groups**: Additional groups a user belongs to.

Basic User and Group Management Commands

Command	Purpose
useradd	Create a new user
usermod	Modify a user account
userdel	Delete a user account
groupadd	Create a new group
groupmod	Modify a group
groupdel	Delete a group
passwd	Set or change user password
id	Show user and group IDs
groups	Show groups a user belongs to
who / w	Show logged-in users
su / sudo	Switch user / run as superuser

Common Administrative Tasks [1]

❑Create a User

- sudo useradd -m john #-m: Create home directory /home/john
- sudo passwd john #Sets password for john

❑Create a Group

- sudo groupadd developers

❑Add User to a Group

- sudo usermod -aG developers john #-aG: Append user to group (don’t remove from others)

❑Change a User’s Primary Group

- sudo usermod -g developers john

Common Administrative Tasks [2]

❑Delete a User

- sudo userdel -r john #-r: Remove user and their home directory

❑Delete a Group

- sudo groupdel developers

Common Administrative Tasks [3]

- **Viewing User and Group Info**
- **Check Current User:**
  - whoami
- **View Current User's Groups:**
  - groups
- **List All Users:**
  - cut -d: -f1 /etc/passwd
- **List All Groups:**
  - cut -d: -f1 /etc/group

File and Directory Permissions

File and Directory Permissions

- Understanding **file and directory permissions** is essential for securing and managing Linux systems. Permissions determine **who can read, write, or execute** a file or access a directory.
- **File and Directory Permissions Overview**
- Each file or directory in Linux has:
  - **Owner**: The user who owns the file.
  - **Group**: A group assigned to the file.
- **Permissions** for:
  - **User (u)** – the owner
  - **Group (g)** – users in the group
  - **Others (o)** – all other users

Example

- ls -l
- -rwxr-xr-- 1 alice devs 1234 Nov 6 script.sh

Symbol	Meaning
-	Regular file (d = directory, l = symlink)
rwx	Owner: read, write, execute
r-x	Group: read, execute
r--	Others: read only

Permission	File	Directory
r (read)	View content	List files in directory
w (write)	Modify content	Create, delete, or rename files
x (execute)	Run the file	Enter (cd) or traverse directory

Managing Permissions [1]

❑View Permissions

- ls -l filename

❑Change Permissions with chmod

- chmod u+x script.sh # Add execute for owner
- chmod g-w file.txt # Remove write for group
- chmod o=r file.txt # Set others to read-only
- chmod 755 script.sh # rwxr-xr-x
- chmod 644 file.txt # rw-r--r--

Managing Permissions [2]

- **Numeric Representation:**
- Each permission has a number:
  - r = 4
  - w = 2
  - x = 1

User	rwx	Value
Owner	rwx	7
Group	r-x	5
Others	r-x	5

Managing Permissions [3]

- Change Ownership with chown
  - sudo chown user:group file.txt
- Example:
  - sudo chown alice:devs report.log

SUID, SGID, Sticky Bit Explained

Special Permission Bits in Linux

- These are **special permission bits** that control advanced behavior for files and directories in Linux. They influence who can execute or delete files under certain conditions.
- **Types of File Permission:**
  - SUID (Set User ID)
  - SGID (Set Group ID)
  - Sticky Bit
- **How to Identify Them**
- Use ls -l:
  - **SUID** ⟶ s or S in **owner** exec bit
  - **SGID** ⟶ s or S in **group** exec bit
  - **Sticky** ⟶ t or T in **others** exec bit

SUID (Set User ID)

❑Definition:

- When the SUID bit is set on an **executable file**, the file runs with the **permissions of the file owner** (usually root), **not the user executing it**.

❑Purpose:

- Allows regular users to execute programs with elevated privileges when needed.

❑Example Use Case:

- The passwd command needs to modify /etc/shadow (a root-owned file), but it must be executable by normal users.

Example

- **Check it:**
  - ls -l /usr/bin/passwd
- Expected Output:
  - -rwsr-xr-x 1 root root ... /usr/bin/passwd
  - Note the **s** in the owner's execute position (rws).
- **Set SUID:**
  - chmod u+s filename
- **Remove SUID:**
  - chmod u-s filename

SGID (Set Group ID)

❑Definition:

- On a **file**: SGID makes the file run with the permissions of the file's **group**, not the user running it.
- On a **directory**: SGID makes **new files/folders inherit the group ownership** of the directory.

❑Purpose:

- On files: Allows programs to run with group-level privileges.
- On directories: Ensures consistent group collaboration.

❑Example Use Case (Directory):

- Shared group directory /shared where all users need to create files that remain in the group devteam.

Example [1]

❑Set SGID on directory:

- sudo mkdir /shared
- sudo chown root:devteam /shared
- sudo chmod 2775 /shared

❑Check it:

- ls -ld /shared

❑Expected Output:

- drwxr-sr-x 2 root devteam ...
- Note the **s** in the group execute position.

Example [2]

- **Set SGID on file:**
  - chmod g+s filename
- **Remove SGID:**
  - chmod g-s filename

Sticky Bit

❑Definition:

- When set on a **directory**, only the **file owner**, the **directory owner**, or **root** can delete or rename the files within it — even if other users have write access.

❑Purpose:

- Prevents users from deleting each other's files in shared directories like /tmp.

❑Example Use Case:

- Protecting user files in /tmp, which is writable by all users.

Example

- **Check it:**
  - ls -ld /tmp
- **Expected Output:**
  - drwxrwxrwt 10 root root ...
  - Note the **t** at the end.
- **Set Sticky Bit:**
  - chmod +t /sharedfolder
- **Remove Sticky Bit:**
  - chmod -t /sharedfolder

Summary of Commands

Action	Command
Set SUID	chmod u+s file
Remove SUID	chmod u-s file
Set SGID (file/dir)	chmod g+s file_or_dir
Remove SGID	chmod g-s file_or_dir
Set Sticky Bit	chmod +t dir
Remove Sticky Bit	chmod -t dir

sudo and Privilege Escalation in Linux

What is sudo?

- sudo (short for **"superuser do"**) is a command-line utility in Linux that allows a permitted user to **run specific commands with superuser (root) privileges** — without needing to log in as the root user.
- **Purpose:**
  - Perform administrative tasks (install software, restart services, modify system files)
  - Limit full root access while still allowing necessary actions
  - Improve security by controlling who can do what

Why Use sudo Instead of Logging in as Root?

- **Accountability**: All sudo commands are logged in /var/log/auth.log
- **Granular control**: You can limit what commands a user can run
- **Security**: Avoids giving full root shell access
- **Temporary access**: Only escalates privileges for the current command

What is Privilege Escalation?

- **Privilege escalation** refers to **gaining higher access rights than normally permitted**. This can be:
  - **Legitimate**: A sysadmin using sudo to perform system tasks
  - **Malicious**: An attacker exploiting a vulnerability to gain root access
- **Types:**
  - **Vertical escalation**: Standard user becomes root (e.g., via sudo)
  - **Horizontal escalation**: User gains access to another user's data or processes

Example: Use sudo

- **How to Use sudo**
- Run a command with elevated privileges:
  - sudo apt update
  - sudo systemctl restart sshd
- Edit a system file (e.g., /etc/hosts) using a text editor:
  - sudo nano /etc/hosts
- Get a root shell (if permitted):
  - sudo -i

Example: Configuration

- **sudo Configuration – /etc/sudoers**
  - This file defines which users or groups can use sudo and what commands they can run.
- **Best practice:** Edit this file safely using:
  - sudo visudo
- Example entry in /etc/sudoers:
  - alice ALL=(ALL) ALL
  - This allows user alice to run any command as any user, on any host.
- Limited permission:
  - bob ALL=(ALL) NOPASSWD: /usr/sbin/reboot
  - This allows bob to reboot the system without a password, but nothing else.

Example: Configuration

- **Check sudo privileges**
- To see what you're allowed to do with sudo:
  - sudo –l
- A junior sysadmin has limited access. Their sudo -l shows:
  - (ALL) NOPASSWD: /usr/bin/systemctl restart apache2

DEMO

Hands-On Lab [1]

❑User Management, File Permissions, and Sudo Configuration

❑Objective

- By the end of this lab, participants will be able to:
- Create and manage Linux user accounts
- Set and modify file permissions and ownership
- Grant and restrict sudo privileges

Creating Users and Groups

- ❑ Create a New User
  - sudo useradd -m alice
  - -m: Creates a home directory /home/alice
- ❑ Set a Password
  - sudo passwd alice
  - Choose a secure password when prompted.
- ❑ Create a New Group and Add User to It
  - sudo groupadd developers
  - sudo usermod -aG developers alice
  - -aG: Adds alice to the developers group **without removing other groups**

Creating Users and Groups

- Verify Group Membership
  - groups alice

Setting File Permissions

- ❑ Create a Sample File and Directory
  - mkdir /home/alice/testdir
  - touch /home/alice/testdir/testfile.txt
- ❑ Change Ownership of the File
  - sudo chown alice:developers /home/alice/testdir/testfile.txt
- ❑ Set Read/Write Permissions
  - chmod 640 /home/alice/testdir/testfile.txt
- ❑ **Explanation of 640:**
  - **6** = read + write for owner
  - **4** = read for group
  - **0** = no access for others

Setting File Permissions

- Make Directory Group-Writable with SGID
  - chmod 2775 /home/alice/testdir
  - The **2** sets the **SGID bit**, so files created in this folder inherit the group developers.

Configuring Sudo Access

- Add User to sudo Group (Debian/Ubuntu)
  - sudo usermod -aG sudo alice
- **Test sudo Access**
  - su – alice
  - sudo whoami
  - root

Create a Custom Sudo Rule

- ❑ Edit the sudoers file **safely**
  - sudo visudo
- ❑ Add a line at the end:
  - alice ALL=(ALL) NOPASSWD: /usr/sbin/reboot
  - Now, alice can run the reboot command **without a password**, but no other root commands**.**
- ❑ **Test It:**
  - sudo /usr/sbin/reboot

Service and Process Management

Agenda

❑ Service and Process Management

- Systemd overview (systemctl, journalctl)
- Managing services (start, stop, enable, disable)
- Monitoring system resources (top, htop, ps, kill)
- Log file management (/var/log, logrotate)
- **Hands-on:** Managing services and viewing logs

Systemd overview

What is systemd?

- systemd is a **system and service manager** for Linux operating systems. It is the **first process** (PID 1) that starts at boot and is responsible for initializing the system — including mounting filesystems, starting services (daemons), and managing processes.
- **Used in:** Most modern distros like **Ubuntu (16.04+), RHEL/CentOS 7+, Debian 8+, Fedora**, and others.
- **Purpose of systemd**
  - Bootstraps the system (replaces traditional init)
  - Manages background services (starting, stopping, restarting)
  - Tracks system state and logs with **journal**
  - Handles dependencies between services
  - Speeds up boot time using parallelization

Key Components of systemd

Component	Purpose
unit files	Configuration files for services, sockets, targets
systemctl	CLI tool to manage units and system state
journald	Logging system used by systemd
targets	Group of services representing runlevels (e.g., multi-user.target)

Managing services

Managing Services in Linux

- In modern Linux systems (Ubuntu, RHEL, Fedora, Debian), services are managed using the **systemd** init system, and the primary tool for interacting with it is **systemctl**.
- A **service** (also called a **daemon**) is a background process that performs tasks such as running a web server, a database, or managing logs.
- **Objectives of Service Management**
  - Start or stop services
  - Enable or disable them on boot
  - Restart or reload them when configuration changes
  - Check their current status
  - View their logs

Common systemctl Commands

❑ Managing Services

- sudo systemctl start nginx # Start a service
- sudo systemctl stop nginx # Stop a service
- sudo systemctl restart nginx # Restart a service
- sudo systemctl reload nginx # Reload config without full restart

❑ Enabling/Disabling at Boot

- sudo systemctl enable nginx # Start service at boot
- sudo systemctl disable nginx # Disable service from boot

Common systemctl Commands

- Status and Logs
  - sudo systemctl status nginx # Show active status
  - sudo journalctl -u nginx # Show logs for nginx service
- Check All Services
  - systemctl list-units --type=service

Common systemctl Commands

- List and Monitor
  - systemctl list-units --type=service # List active services
  - systemctl list-unit-files --type=service # List all (enabled/disabled)
- View detailed info:
  - systemctl show nginx
- Check logs with journald:
  - journalctl -u nginx # Show logs for nginx
  - journalctl -xe # Show recent systemd errors
  - journalctl -b # Logs since last boot

Summary

Command	Function
systemctl start <service>	Start the service now
systemctl stop <service>	Stop the service
systemctl restart <service>	Restart the service
systemctl reload <service>	Reload config without full restart
systemctl enable <service>	Enable service at boot
systemctl disable <service>	Disable service from boot
systemctl status <service>	Show current status
journalctl -u <service>	Show logs for the service

Monitoring System Resources

Monitoring System Resources in Linux

- Monitoring system resources in Linux is essential for performance tuning, troubleshooting, and ensuring system stability. Here's a comprehensive breakdown of **how to monitor CPU, memory, disk, and network usage** using built-in and common tools.

- **Key System Resources to Monitor**
  - CPU Usage
  - Memory (RAM)
  - Disk I/O and Space
  - Network Usage
  - Running Processes

Essential Monitoring Tools

❑Real-time process and system usage

- top
- Shows CPU, memory, load average, and process usage.
- Press Shift + P to sort by CPU, Shift + M to sort by memory.
- Press q to quit.

❑Enhanced interactive process viewer (needs to be installed)

- htop
- Colorful, user-friendly version of top.
- Allows process management with function keys (e.g., kill with F9)

❑Install with:

- sudo apt install htop # Debian/Ubuntu

Essential Monitoring Tools

- **Memory, CPU, I/O statistics**
  - vmstat 2 5
  - Reports system performance at intervals.
  - Columns include: procs, memory, swap, I/O, CPU.
- **Memory usage**
  - free –h
  - Shows used, free, and cached memory in a human-readable format.
- **CPU and disk I/O (from sysstat package)**
  - iostat -xz 1 5
  - Reports per-device disk I/O and CPU usage.
  - Install with: sudo apt install sysstat # Debian/Ubuntu

Essential Monitoring Tools

❑Disk space usage

- df -h
- Shows available and used disk space on all mounted filesystems.

❑Directory and file space usage

- du -sh /var/log
- Shows size of a specific directory or file.

❑Real-time disk I/O per process (needs to be installed)

- sudo iotop
- Shows which processes are using the disk.
- Requires root privileges.

Essential Monitoring Tools

❑Network traffic monitoring

- ip -s link
- Shows network interface statistics.

❑Real-time:

- sudo apt install iftop nload
- sudo iftop # Real-time traffic between hosts
- sudo nload # Real-time traffic per interface

❑Process snapshot

- ps aux | less
- List all running processes with CPU and memory usage.
- Combine with grep to filter: ps aux | grep nginx

Essential Monitoring Tools

- **Network connections**
  - ss -tuln # Show listening ports and services
- **Load average and uptime**
  - uptime
  - Shows how long the system has been running and load averages over 1, 5, and 15 minutes.
- **System activity reports over time (from sysstat)**
  - sar -u 1 5 # CPU usage
  - sar -r 1 5 # Memory
  - sar -n DEV 1 5 # Network interface stats

Log file management

What Are Log Files?

- Log files are **records of system and application events**, created to track activity, errors, warnings, and user actions.
- **Purpose:**
  - Diagnose system or application issues
  - Monitor system health and performance
  - Detect unauthorized access or suspicious activity
  - Provide an audit trail for compliance and forensics

Common Log File Locations

- Most logs are stored in the /var/log/ directory.

File	Purpose
/var/log/syslog or /var/log/messages	General system logs (depends on distro)
/var/log/auth.log	Authentication attempts (Ubuntu/Debian)
/var/log/secure	Authentication and security logs (RHEL/CentOS)
/var/log/kern.log	Kernel messages
/var/log/dmesg	Hardware and boot messages
/var/log/boot.log	Boot process messages
/var/log/faillog	Failed login attempts
/var/log/httpd/ or /var/log/apache2/	Web server logs
/var/log/journal/	Binary logs from systemd-journald

Basic Commands for Log File Management

❑View Logs

- cat /var/log/syslog # Full content
- less /var/log/syslog # Scrollable view
- tail -n 50 /var/log/syslog # Last 50 lines

❑Real-Time Monitoring

- tail -f /var/log/auth.log # Follow auth log in real-time

❑Search in Logs (e.g., for failed login)

- grep "Failed password" /var/log/auth.log

❑Filter Logs by Date or IP (using grep and awk)

- grep "2025-11-05" /var/log/syslog
- grep "192.168.1.10" /var/log/auth.log

Log Rotation with logrotate

- logrotate is a utility that **automatically rotates, compresses, and removes old log files** to save space and manage logs efficiently.
- **logrotate** is used to automatically:
  - Rotate logs (rename current log file and start a new one)
  - Compress old logs (gzip)
  - Delete logs older than a certain time
  - Keep disk usage under control
- **Config File Locations:**
  - Global config: /etc/logrotate.conf
  - Service-specific: /etc/logrotate.d/

Example Configuration

- Example config (/etc/logrotate.d/nginx):
  - /var/log/nginx/*.log {
  - daily
  - rotate 7
  - compress
  - missingok
  - notifempty
  - create 0640 nginx adm
  - }
- **Run logrotate manually:**
  - sudo logrotate -f /etc/logrotate.conf

systemd Journal (journald)

- Modern Linux systems using systemd use **journald** to log messages in binary format.
- **View logs:**
  - journalctl # View all logs
  - journalctl -xe # Show recent errors with context
  - journalctl -u ssh.service # Logs for SSH service
  - journalctl --since today # Logs from today
- **Clear old journal logs:**
  - sudo journalctl --vacuum-time=7d

Demo

Hands-On Lab

❑Hands-On Lab: Managing Services and Viewing Logs

❑Lab Objectives

- By the end of this lab, you will be able to:
- Start, stop, enable, and restart services using systemctl
- Check the status of a service
- View and monitor service logs using journalctl and tail
- Understand the connection between service management and system logging

Managing Services [1]

❑ Check if the SSH Service is Running

- sudo systemctl status ssh
- Look for "Active: active (running)" in the output.

❑ Start the Service

- sudo systemctl start ssh

❑ Stop the Service

- sudo systemctl stop ssh
- sudo systemctl status ssh
- You should now see "Active: inactive (dead)"

Managing Services [2]

- **Restart the Service**
  - sudo systemctl restart ssh
- Enable SSH to Start on Boot
  - sudo systemctl enable ssh
- To test:
  - systemctl is-enabled ssh
- Disable SSH from Starting at Boot
  - sudo systemctl disable ssh

Viewing Logs [1]

❑View Logs for SSH Service

- sudo journalctl -u ssh
- Scroll through logs using the arrow keys, press q to exit.

❑View Logs in Real Time

- sudo journalctl -u ssh –f
- This follows the log as new entries are added (like tail -f).

❑In another terminal, restart SSH:

- sudo systemctl restart ssh

Viewing Logs [2]

- **View Recent Log Entries Only**
  - sudo journalctl -u ssh --since "10 minutes ago“
- Search for a Keyword in Logs
  - sudo journalctl -u ssh | grep "Failed"
- Use tail to View Traditional Log Files
  - sudo tail -n 50 /var/log/auth.log

Core Linux Security Concepts

Agenda

❑ Core Linux Security Concepts

- Understanding attack surfaces
- File Integrity Checking (AIDE)
- Linux firewall basics (iptables, firewalld, or nftables)
- Disabling unnecessary services
- SSH security: disabling root login, using key-based auth
- **Hands-on:** Securing SSH, configuring simple firewall rules

Understanding attack surfaces

What is an Attack Surface?

- An **attack surface** is the **total set of points** where an unauthorized user (attacker) could try to enter, exploit, or extract data from a system.
- In Linux, the attack surface includes:
  - Running services
  - Open ports
  - Software packages
  - User accounts
  - Files with weak permissions
  - Kernel interfaces
  - External devices or connected networks
- **Goal of a secure system:**
  Minimize the attack surface to reduce the risk of exploitation.

Categories of Attack Surfaces in Linux [1]

- **Network Attack Surface**
- These are **services listening on open ports** that can be reached over the network.
- Examples:
  - SSH (port 22)
  - Web server (port 80/443)
  - FTP, Samba, DNS
- **Reduce it by:**
  - Disabling unused services
  - Using firewalls (ufw, iptables, nftables)
  - Using secure protocols (e.g., SSH instead of Telnet)

Categories of Attack Surfaces in Linux [2]

- **Software and Application Attack Surface**
- Every installed **package or app** increases potential vulnerabilities.
- Examples:
  - Outdated web apps (e.g., PHP apps)
  - Vulnerable versions of OpenSSL, Apache, etc.
- **Reduce it by:**
  - Uninstalling unnecessary packages
  - Regular patching and updates
  - Using minimal, hardened Linux distributions

Categories of Attack Surfaces in Linux [3]

- **User and Privilege Attack Surface**
- User accounts with **weak passwords**, **sudo access**, or **excessive permissions** are common attack vectors.
- Examples:
  - Unused accounts
  - sudo given to too many users
  - Default or weak passwords
- **Reduce it by:**
  - Enforcing strong password policies
  - Reviewing and limiting sudo access
  - Disabling or removing unused accounts

Categories of Attack Surfaces in Linux [4]

- **File System and Permission Attack Surface**
- Files or directories with **improper permissions** can be misused to escalate privileges or exfiltrate data.
- Examples:
  - World-writable files or directories
  - SUID/SGID misuse
  - Sensitive files without correct access control
- **Reduce it by:**
  - Auditing file permissions (find / -perm -o+w)
  - Using tools like AIDE, tripwire for integrity checks
  - Disabling unnecessary SUID/SGID bits

Categories of Attack Surfaces in Linux [5]

- **Kernel and System Interface Attack Surface**
- Linux exposes **kernel modules**, **syscalls**, and **/proc/sys interfaces** which can be abused by attackers.
  - Examples:
    - Loading malicious kernel modules
    - Abusing /proc or /sys to manipulate kernel behavior
- **Reduce it by:**
    - Disabling unneeded kernel modules
    - Enforcing kernel hardening with sysctl settings
    - Using Mandatory Access Control (MAC) like SELinux or AppArmor

Minimize the Attack Surface

Area	Action
Services	Disable or remove unused services
Packages	Install only what is needed (minimal install)
Firewall	Block unused ports
Users	Remove inactive accounts, use least privilege
Sudo	Limit sudo access and audit usage
Permissions	Set correct file/directory permissions
Auditing	Use tools like auditd, fail2ban, logwatch
Security Layers	Use SELinux, AppArmor, and file integrity monitoring

File Integrity Checking (AIDE)

File Integrity Checking [1]

- **File Integrity Checking with AIDE (Advanced Intrusion Detection Environment)** is a powerful way to detect **unauthorized changes** to important files and directories on a Linux system — a crucial part of system hardening and compliance (e.g., PCI-DSS, CIS Benchmarks).

- **What Is AIDE?**
  - **AIDE** is a **host-based intrusion detection system (HIDS)**. It works by:
  - **Creating a database** of file hashes, permissions, ownership, etc.
  - **Later comparing** the current state of the filesystem to the saved baseline.
  - **Alerting** you to any unauthorized changes.

File Integrity Checking [2]

❑Why Use AIDE?

- Detect tampering with system binaries (e.g., /bin/ls, /usr/bin/ssh)
- Monitor config files (/etc/passwd, /etc/ssh/sshd_config)
- Identify rootkits or unauthorized file changes
- Meet compliance/security audit requirements

Install AIDE [1]

- **Install AIDE On Debian/Ubuntu:**
  - sudo apt update
  - sudo apt install aide
- Initialize the AIDE Database
  - sudo aide --init
- This creates a file like:
  - /var/lib/aide/aide.db.new.gz
- Now move it to become the active database:
  - sudo mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz

Install AIDE [2]

- **Run a Check (Detect Changes)**
  - sudo aide --check
- It compares current file state against the database and shows any differences (modified, added, deleted files).
- Example output:
  - Changed files:
  - /etc/ssh/sshd_config
  - Added files:
  - /tmp/hack.sh
  - Removed files:
  - /etc/myapp/config.conf

Install AIDE [3]

- **Update the Database (After Legitimate Changes)**
- If you made authorized changes (e.g., updated system packages or changed config files), rebuild the database:
  - sudo aide --update
- This creates a new database:
  - /var/lib/aide/aide.db.new.gz
- Replace the old one:
  - sudo mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz

Install AIDE [4]

- **Configure What AIDE Monitors**
  - /etc/aide/aide.conf
- It defines what to check, including directories and rules.
- Example:
  - /etc NORMAL
  - /bin NORMAL
  - /sbin NORMAL
  - /usr NORMAL
- You can define custom rules like:
  - NORMAL = p+i+n+u+g+s+m+c+md5+sha512

Install AIDE [5]

- **Automate AIDE with Cron**
  - sudo crontab –e
  - @daily /usr/bin/aide --check | mail -s "AIDE Integrity Check" admin@example.com
- AIDE Output Example
- After running aide --check, sample output:
  - AIDE found differences between database and filesystem!!
  - Summary:
  - Total number of entries: 34579
  - Added files: 2
  - Removed files: 0
  - Changed files: 3

Linux firewall basics

What Is a Firewall?

- A **firewall** is a system or software that controls **incoming and outgoing network traffic** based on defined security rules. It helps **protect your system from unauthorized access, scanning, or attacks**.
- In Linux, firewalls can be **configured at the host level**, allowing you to control traffic to/from that specific machine.
- **Common Linux Firewall Tools:**
  - Iptables
  - nftables
  - ufw
  - firewalld

What is iptables?

- iptables is a command-line utility used to configure the **Linux kernel firewall** provided by **Netfilter**. It lets you **define rules** to control incoming and outgoing traffic on your system.

- **Purpose of iptables**
  - Allow or block specific traffic based on IP, port, protocol, interface, etc.
  - Act as a **packet filter** for firewall functionality.
  - Control **access to services**, **mitigate attacks**, or **isolate** parts of a system.

Basic Structure of an iptables Rule

- iptables [CHAIN] [CONDITIONS] -j [TARGET]
- CHAIN: The traffic flow hook (e.g. INPUT, OUTPUT, FORWARD)
- CONDITIONS: Match criteria (IP, port, protocol, interface)
- TARGET: Action to take (ACCEPT, DROP, REJECT)
- Built-in Chains

Chain	Purpose
INPUT	Incoming packets to the local system
OUTPUT	Outgoing packets from the system
FORWARD	Packets routed through the system

Common iptables Targets

Target	Description
ACCEPT	Allow the packet
DROP	Silently discard the packet
REJECT	Reject the packet with error message
LOG	Log the packet to syslog

•-n: don’t resolve names (faster)
•-v: verbose

Basic iptables Commands [1]

- Allow Incoming Traffic (e.g., SSH on port 22)
  - sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
- Block All Incoming Traffic (Set Default Policy)
  - sudo iptables -P INPUT DROP
  - sudo iptables -P FORWARD DROP
  - sudo iptables -P OUTPUT ACCEPT
- List Current Rules
  - sudo iptables -L -n -v
  - -L: list rules
  - -n: don’t resolve names (faster)
  - -v: verbose

Basic iptables Commands [2]

- Delete a Rule
  - sudo iptables -D INPUT -p tcp --dport 22 -j ACCEPT
- Flush (Clear) All Rules
  - sudo iptables -F
- Save Rules (So They Persist After Reboot)
  - sudo apt install iptables-persistent
  - sudo netfilter-persistent save
- Log dropped packets (careful not to fill disk):
  - sudo iptables -A INPUT -j LOG --log-prefix "IPTables-Dropped: "

Common Use Cases

- ❑ Allow SSH
  - sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
  - sudo iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
- ❑ Allow HTTP/HTTPS
  - sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
  - sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
- ❑ Allow Loopback Interface
  - sudo iptables -A INPUT -i lo -j ACCEPT
- ❑ Allow Established Connections
  - sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

Common Use Cases

- Drop Everything Else
  - sudo iptables -A INPUT -j DROP

What Is firewalld?

- firewalld is a **dynamic firewall manager** that provides a **high-level, zone-based interface** for managing network traffic using **iptables/nftables** under the hood — but without the need to reload all rules when changes are made.

- **Purpose and Advantages**
  - **Dynamic changes** — no need to restart the firewall daemon
  - **Zone-based security** — easier to manage different trust levels for interfaces
  - **Rich rule support** — more advanced than simple iptables
  - Integrates with **services**, **ports**, **protocols**, and **interfaces**

Installing firewalld

❑ On RHEL/CentOS/Fedora:

- sudo dnf install firewalld
- sudo systemctl enable firewalld
- sudo systemctl start firewalld

❑ On Ubuntu/Debian:

- sudo apt install firewalld
- sudo systemctl enable firewalld
- sudo systemctl start firewalld

Firewalld Concepts

- **Zones**
- Define **trust levels** for different network connections.

Zone	Trust Level
drop	All incoming denied
block	Similar to drop, with rejection
public	Default, untrusted zone
external	For router/NAT setups
internal	Trusted for LAN use
home, work, trusted	Increasing levels of trust

Basic firewalld Commands

❑Check Firewall Status

- sudo firewall-cmd --state

❑List All Zones and Services

- sudo firewall-cmd --get-zones
- sudo firewall-cmd --get-services

❑View Active Zone and Rules

- sudo firewall-cmd --get-active-zones
- sudo firewall-cmd --list-all

❑Assign an Interface to a Zone

- sudo firewall-cmd --zone=public --change-interface=eth0 --permanent

Basic firewalld Commands

❑ Allow a Service (e.g., SSH, HTTP)

- sudo firewall-cmd --zone=public --add-service=ssh --permanent
- sudo firewall-cmd --zone=public --add-service=http --permanent
- sudo firewall-cmd --reload

❑ Open a Specific Port

- sudo firewall-cmd --zone=public --add-port=8080/tcp --permanent
- sudo firewall-cmd --reload

❑ Remove a Service or Port

- sudo firewall-cmd --zone=public --remove-service=http --permanent
- sudo firewall-cmd --zone=public --remove-port=8080/tcp --permanent
- sudo firewall-cmd --reload

What is nftables?

- nftables is the **successor to iptables, ip6tables, arptables, and ebtables**, providing a **unified, more efficient, and flexible firewall management** system in Linux.
- **Backend:** Built into the Linux kernel since version **3.13+**, replacing the older iptables backend (xtables)

Why Use nftables Instead of iptables?

Benefit	Description
Unified Framework	One tool (nft) for IPv4, IPv6, ARP, and bridge traffic
Faster & Efficient	Uses a pseudo-state machine; better performance
Safer Syntax	Easier to read, less error-prone
Rule Sets	Rules stored in a single atomic configuration
Reusable Sets	Define IP sets, port sets, and apply them to rules

Installing nftables

- Most modern distros include nftables by default.
- On Debian/Ubuntu:
  - sudo apt install nftables
  - sudo systemctl enable nftables
  - sudo systemctl start nftables
- On RHEL/CentOS/Rocky 8+:
  - sudo dnf install nftables
  - sudo systemctl enable nftables
  - sudo systemctl start nftables

Basic Concepts in nftables

Concept	Description
Table	Container for rules (e.g., inet filter)
Chain	Holds the actual rules (e.g., input, output)
Rule	The filtering or NAT logic
Set	Reusable list of IPs/ports

Basic Concepts in nftables

- **Default Table Types**
- ip – IPv4 traffic
- ip6 – IPv6 traffic
- inet – Combined IPv4 and IPv6 (**recommended**)
- arp – ARP traffic
- bridge – Bridged traffic

Example: Basic nftables Firewall Rules

- sudo nft add table inet myfilter
- sudo nft add chain inet myfilter input { type filter hook input priority 0 \; policy drop \; }
- sudo nft add rule inet myfilter input iif "lo" accept
- sudo nft add rule inet myfilter input ct state established,related accept
- sudo nft add rule inet myfilter input tcp dport 22 accept
- sudo nft add rule inet myfilter input tcp dport 80 accept
- sudo nft add rule inet myfilter input tcp dport 443 accept

Example: Basic nftables Firewall Rules

- View Current Ruleset
  - sudo nft list ruleset
- Save and Load Rules
  - sudo nft list ruleset > /etc/nftables.conf
- Load Rules:
  - sudo nft -f /etc/nftables.conf

Disabling Unnecessary Services in Linux

Why Disable Unnecessary Services?

❑Why Disable Unnecessary Services?

❑Every running service:

- Uses system resources (CPU, RAM, ports)
- Increases the **attack surface** (e.g., vulnerable software, open ports)
- May expose the system to unauthorized access or data leakage

❑Goal:

Run only what you need. This is a core security best practice in Linux hardening.

Step-by-Step [1]

❑View Running Services

- systemctl list-units --type=service --state=running

❑Find the Service Name

- systemctl status servicename
- systemctl list-unit-files | grep enabled

❑Stop the Service

- sudo systemctl stop servicename

❑Disable the Service

- sudo systemctl disable servicename

Step-by-Step [2]

❑ Mask the Service

- sudo systemctl mask servicename
- sudo systemctl mask bluetooth

❑ Disable Telnet, FTP, and Other Legacy Services

- sudo systemctl stop telnet
- sudo systemctl disable telnet
- sudo systemctl stop vsftpd
- sudo systemctl disable vsftpd

Step-by-Step [3]

- ❑ **List all installed services (enabled/disabled)**
  - systemctl list-unit-files --type=service
- ❑ **Find services listening on ports**
  - sudo ss -tuln
- ❑ **Identify services using the network**
  - sudo netstat -tulpen # or use `ss -tulpen` if netstat isn't available

Common Services to Review or Disable

Service	Use Case	Safe to Disable If...
cups	Printing service	No printer is used
bluetooth	Bluetooth devices	No Bluetooth peripherals
avahi-daemon	Network discovery (mDNS)	No Bonjour/Apple devices
rpcbind	NFS/RPC services	Not using NFS or remote mounts
telnet	Remote shell (insecure)	Always — use SSH instead
vsftpd	FTP server	Not hosting an FTP server
postfix	Mail server	Not sending local mail
smb	Samba/Windows sharing	No Windows shares needed
httpd, nginx	Web servers	Not hosting websites

SSH Security

- Securing **SSH (Secure Shell)** is **crucial** for protecting remote access to your Linux systems. By default, SSH provides encrypted communication, but **default settings can still be vulnerable** to brute-force attacks, credential theft, or misconfiguration.
- **Why Secure SSH?**
- SSH is **a common attack vector**. If misconfigured, it can expose your system to:
  - Brute-force login attempts
  - Credential theft
  - Remote code execution
- Hardening SSH is critical to reducing the attack surface.

SSH Security Best Practices [1]

- ❑ **Use Key-Based Authentication (Disable Passwords)**
- ❑ Generate SSH Key Pair (Client Side)
  - ssh-keygen -t ed25519 -C "your_email@example.com"
- ❑ **Copy Key to Server**
  - ssh-copy-id user@server
  - Manually: Copy contents of ~/.ssh/id_ed25519.pub into the server's ~/.ssh/authorized_keys
- ❑ **Disable Password Authentication on Server**
- ❑ Edit /etc/ssh/sshd_config
  - PasswordAuthentication no
  - ChallengeResponseAuthentication no
- ❑ sudo systemctl restart sshd

SSH Security Best Practices [2]

- **Disable Root Login via SSH**
- Edit /etc/ssh/sshd_config
  - PermitRootLogin no
- sudo systemctl restart sshd
- **Set Login/Connection Limits**
- **Edit /etc/ssh/sshd_config**
  - MaxAuthTries 3
  - LoginGraceTime 30
  - MaxSessions 2

Demo

Hands-on

❑Hands-On Lab: Securing SSH & Configuring Simple Firewall Rules

❑Objective

- By the end of this lab, you will:
- Harden SSH access (disable root login, enable key-based auth)
- Restrict SSH access using the firewall (ufw or firewalld)
- Understand and apply security best practices for remote access

Secure SSH Configuration

❑ Step 1: Backup SSH Configuration

- sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak

❑ Step 2: Disable Root Login via SSH

- sudo nano /etc/ssh/sshd_config
- PermitRootLogin no

❑ Step 3: Restrict SSH Access to Specific Users

- sudo nano /etc/ssh/sshd_config
- AllowUsers alice bob

❑ Step 4: Restart SSH Service

- sudo systemctl restart sshd

Configure Basic Firewall Rules:ufw

❑Step 1: Enable UFW

- sudo ufw enable

❑Step 2: Set Default Policies

- sudo ufw default deny incoming
- sudo ufw default allow outgoing

❑Step 3: Allow SSH Access

- sudo ufw allow 22/tcp

❑Step 4: Allow Web Ports

- sudo ufw allow 80/tcp
- sudo ufw allow 443/tcp

❑Step 5: Check Firewall Status

- sudo ufw status verbose

Patch Management and Hardening

Agenda

❑Patch Management and Hardening

- Keeping the system updated (apt, dnf, yum, unattended-upgrades)
- Kernel updates and reboot strategies
- Hardening system defaults (sysctl settings)
- Minimizing installed packages
- **Hands-on:** Applying updates, using sysctl to harden networking

What is Patch Management?

- **Patch management** involves **regularly updating software packages, kernels, and configurations** to fix:
  - Security vulnerabilities
  - Bug fixes
  - Performance improvements
- **Why It Matters**
  - Unpatched systems are a top target for attackers.
  - Zero-day exploits often target known but unpatched software.
  - Patch compliance is mandatory in many regulated environments (e.g., PCI-DSS, HIPAA, ISO 27001).

Patch Management Tools by Distro

Distribution	Tool	Commands
Debian/Ubuntu	apt	sudo apt update && sudo apt upgrade
RHEL/CentOS	dnf or yum	sudo dnf update or sudo yum update
Fedora	dnf	sudo dnf upgrade
Arch Linux	pacman	sudo pacman -Syu

Keeping the system updated

Basic Patch Management Workflow

❑Checking for Updates

❑On Debian/Ubuntu:

- sudo apt update # Refresh package list
- sudo apt list --upgradable
- sudo apt upgrade # Apply all updates
- sudo apt full-upgrade # Apply including kernel changes
- sudo apt autoremove #Clean up unused packages

Basic Patch Management Workflow

❑Enable Automatic Security Updates (Ubuntu/Debian)

- sudo apt install unattended-upgrades
- sudo dpkg-reconfigure --priority=low unattended-upgrades

❑Configuration file:

- /etc/apt/apt.conf.d/50unattended-upgrades

❑Test Patch Management with Cron Job

- sudo crontab -e
- 0 3 * * 1 apt update && apt upgrade -y

Kernel updates and reboot strategies

What is the Linux Kernel?

- The **Linux kernel** is the core of the operating system. It manages:
  - Process scheduling
  - Memory management
  - Device drivers
  - Networking stack
  - Security and system calls
- **Kernel updates** are often released to:
  - Patch security vulnerabilities
  - Improve performance or stability
  - Add support for new hardware

Why Kernel Updates Matter

- Kernel vulnerabilities are often **critical**, and exploits can provide **root-level access**.
- Updates are **not always automatic**, even with regular apt or dnf upgrades.
- In most cases, **a reboot is required** to load the new kernel.

How to Check Your Current Kernel Version

- **How to Check**
  - uname –r
- **Example output:**
  - 5.15.0-105-generic
- **To list all installed kernels (Debian/Ubuntu)**
  - dpkg --list | grep linux-image
- **Install Updates:**
  - sudo apt update
  - sudo apt upgrade
  - sudo apt install --install-recommends linux-generic

Hardening system defaults

What is sysctl?

- sysctl is a Linux utility that allows you to **view and modify kernel parameters** at runtime.

  These parameters influence core behavior of the system, especially around:

  - Network security
  - Memory handling
  - Process limits
  - System performance

- By hardening these settings, you reduce your exposure to **network-based attacks**, **kernel exploits**, and **misuse of system resources**.

Where sysctl Settings Are Defined

- **View current settings:**
  - sudo sysctl -a
- **Apply changes temporarily:**
  - sudo sysctl net.ipv4.ip_forward=0
- **Apply changes permanently by editing:**
  - /etc/sysctl.conf
- **Or drop custom .conf files into:**
  - sudo sysctl -p

sysctl Hardening Settings [1]

❑ Disable IP Forwarding (unless routing)

- net.ipv4.ip_forward = 0
- net.ipv6.conf.all.forwarding = 0

❑ Block IP Spoofing

- net.ipv4.conf.all.rp_filter = 1
- net.ipv4.conf.default.rp_filter = 1

❑ Disable Source Routing

- net.ipv4.conf.all.accept_source_route = 0
- net.ipv4.conf.default.accept_source_route = 0
- net.ipv6.conf.all.accept_source_route = 0
- net.ipv6.conf.default.accept_source_route = 0

sysctl Hardening Settings [2]

- ❑ **Enable SYN Flood Protection**
  - net.ipv4.tcp_syncookies = 1
- ❑ **Enable Exec Shield / Address Space Protection**
  - kernel.randomize_va_space = 2
- ❑ **Log Suspicious Packets**
  - net.ipv4.conf.all.log_martians = 1
  - net.ipv4.conf.default.log_martians = 1
- ❑ **Disable IPv6 (if unused)**
  - net.ipv6.conf.all.disable_ipv6 = 1
  - net.ipv6.conf.default.disable_ipv6 = 1

How to Apply Hardening Settings

- **Create a new sysctl file:**
  - sudo nano /etc/sysctl.d/99-hardening.conf
- **Paste your hardening settings**
- Apply changes:
  - sudo sysctl --system
- **Verify individual parameters:**
  - sysctl net.ipv4.ip_forward
  - sysctl net.ipv4.tcp_syncookies

Minimizing installed packages

Minimizing installed packages

- Minimizing installed packages is a **core Linux hardening practice** that reduces the system's **attack surface**, improves **performance**, and simplifies **maintenance**.
- **Why Minimize Installed Packages?**
  - Each installed package is a potential:
  - **Vulnerability** (if it's unpatched or exposed)
  - **Target** (attackers look for unnecessary software)
  - **Maintenance burden** (more updates, more configs)
  - **Fewer packages = less risk**.

When to Minimize Packages

- **When to Minimize Packages**
- During initial OS installation (use “Minimal” or “Core” options)
- Before deploying to production or cloud
- After hardening legacy systems
- As part of container/base image creation

How to Minimize Installed Packages

- **List Installed Packages**
- **Debian/Ubuntu:**
  - dpkg -l
- **RHEL/CentOS/Fedora/Rocky:**
  - rpm -qa

Find and Remove Unused Packages

- **Find and Remove Unused Packages**
- **Start by identifying unnecessary components like:**
  - GUI packages on servers (gnome, xorg, etc.)
  - Compilers (gcc, make) if not needed
  - Mail servers (postfix, exim) if not used
  - FTP/Telnet servers
  - Games or documentation
- **Debian/Ubuntu:**
  - sudo apt remove --purge <package-name>
  - sudo apt autoremove

Use a Minimal Install Base

- ❑ **Use a Minimal Install Base**
- ❑ **Debian/Ubuntu Server**: Choose "Minimal Installation"
- ❑ **RHEL/CentOS**: Use the “Minimal” installation ISO or dnf groupremove:
  - sudo dnf groupremove "GNOME Desktop"
- ❑ **Remove Orphaned Packages**
  - sudo apt autoremove --purge
- ❑ **Disable Optional Repositories**
  - sudo dnf config-manager --set-disabled <repo-id>

Demo

Hands-On Lab [1]

❑Part 1: Apply System Updates

❑Ubuntu / Debian

- sudo apt update && sudo apt list --upgradable
- sudo apt upgrade -y # Apply available updates
- sudo apt full-upgrade -y # Apply all updates including kernel
- sudo reboot

Hands-On Lab [2]

- **Part 2: Harden Networking with sysctl**
- **Step 1: Create a Custom sysctl File**
- Create a new config file:
  - sudo nano /etc/sysctl.d/99-network-hardening.conf

Hands-On Lab [3]

- Paste the following **secure settings**:
  - # Disable IP forwarding
  - net.ipv4.ip_forward = 0
  - net.ipv6.conf.all.forwarding = 0
  - # Block source-routed packets
  - net.ipv4.conf.all.accept_source_route = 0
  - net.ipv4.conf.default.accept_source_route = 0
  - net.ipv6.conf.all.accept_source_route = 0
  - net.ipv6.conf.default.accept_source_route = 0

Hands-On Lab [4]

- Paste the following **secure settings**:
  - # Disable ICMP redirects
  - net.ipv4.conf.all.accept_redirects = 0
  - net.ipv4.conf.default.accept_redirects = 0
  - net.ipv6.conf.all.accept_redirects = 0
  - net.ipv6.conf.default.accept_redirects = 0

  - # Enable SYN cookies to prevent SYN flood attacks
  - net.ipv4.tcp_syncookies = 1

  - # Enable reverse path filtering (anti-spoofing)
  - net.ipv4.conf.all.rp_filter = 1
  - net.ipv4.conf.default.rp_filter = 1

Hands-On Lab [5]

- Paste the following **secure settings**:
  - # Log suspicious packets
  - net.ipv4.conf.all.log_martians = 1
  - net.ipv4.conf.default.log_martians = 1
  - # Enable IP spoofing protection
  - net.ipv4.conf.all.send_redirects = 0
  - net.ipv4.conf.default.send_redirects = 0

Hands-On Lab [6]

❑Step 2: Apply the New sysctl Settings

- sudo sysctl --system

❑Check individual values:

- sudo sysctl net.ipv4.ip_forward
- sudo sysctl net.ipv4.conf.all.accept_redirects

❑Step 3: Test Network Behavior (Optional)

- cat /proc/sys/net/ipv4/ip_forward
- ss -tuln
- ip route

Intrusion Detection and Auditing

Agenda

❑Intrusion Detection and Auditing

- Basics of SELinux / AppArmor
- Introduction to auditd
- Log monitoring and alerting tools (logwatch, fail2ban)
- **Hands-on:** Configuring fail2ban to protect SSH

Basics of SELinux / AppArmor

What Is MAC (Mandatory Access Control)?

- MAC is a security model that **enforces strict access policies**, regardless of user permissions. Unlike traditional DAC (Discretionary Access Control), users and processes **cannot override security rules** set by system policies.
- SELinux and AppArmor implement MAC at the kernel level to **limit what programs and users can do**, even if compromised.

SELinux (Security-Enhanced Linux)

- SELinux is a **kernel-level security module** developed by the NSA and maintained by Red Hat. It uses **security contexts (labels)** and **policy rules** to define what processes can access what resources.

- **Key Concepts**
  - **Subjects**: Users or processes (e.g., httpd_t)
  - **Objects**: Files, ports, devices (e.g., httpd_sys_content_t)
  - **Types**: Used to match subjects to objects (Type Enforcement)
  - **Policies**: Define allowed actions (read, write, execute)

SELinux (Security-Enhanced Linux)

❑SELinux Modes

Mode	Description
Enforcing	Policies are enforced — violations are blocked and logged
Permissive	Violations are only logged — not enforced
Disabled	SELinux is turned off

❑Check current mode:

- getenforce

❑Set mode temporarily:

- sudo setenforce 0 # Permissive
- sudo setenforce 1 # Enforcing

SELinux (Security-Enhanced Linux)

- **SELinux Logs**
- **Violations are logged to:**
  - /var/log/audit/audit.log
- **Use ausearch or sealert to analyze:**
  - sudo ausearch -m avc
  - sudo sealert -a /var/log/audit/audit.log
- **Apache to Use a Custom Port**
  - sudo semanage port -a -t http_port_t -p tcp 8080
  - sudo semanage port -l | grep http_port_t
  - sudo systemctl restart httpd

AppArmor (Application Armor)

- **What is AppArmor?**
- AppArmor is another MAC system used mainly in **Ubuntu and SUSE**. It uses **path-based profiles** to restrict what programs can access based on **file paths**, rather than labels.
- **AppArmor Modes**

Mode	Description
Enforce	Violations are blocked and logged
Complain	Violations are logged but not blocked

- **Check status:**
  - sudo aa-status
- sudo aa-enforce /etc/apparmor.d/usr.sbin.apache2
- sudo aa-complain /etc/apparmor.d/usr.sbin.apache2

AppArmor (Application Armor)

- **AppArmor Profiles**
- **Profiles are stored in:**
  - /etc/apparmor.d/
- **Each profile defines file access rules, like:**
  - /usr/sbin/apache2 {
  - /var/www/** r,
  - /etc/apache2/** r,
  - /usr/sbin/apache2 mr,
  - }
- **Load profiles:**
  - sudo apparmor_parser -r /etc/apparmor.d/usr.sbin.apache2

SELinux vs AppArmor

Feature	SELinux	AppArmor
Policy Type	Label-based (context-aware)	Path-based
Complexity	More complex, fine-grained	Simpler, easier to manage
Used by	RHEL, CentOS, Fedora	Ubuntu, Debian, SUSE
Control	Requires understanding of labeling	Easier for quick setups
Tooling	semanage, restorecon, audit2allow	aa-status, aa-complain, aa-enforce

auditd

What is auditd?

- auditd is the **user-space component** of the Linux Auditing System. It **records and logs detailed events** related to system activity, including:
  - File access
  - User logins and logouts
  - Privilege escalations (e.g., sudo usage)
  - Changes to critical files
  - Executed commands
- **Why Use auditd?**
  - Improve **security visibility**
  - Track **unauthorized changes**
  - Meet **compliance requirements** (PCI-DSS, HIPAA, etc.)
  - Support **forensics** after an incident

Key Components

Component	Description
auditd	The main daemon that logs audit events
auditctl	Tool to manage audit rules (temporary)
augenrules	Tool to build rules from /etc/audit/rules.d/ (persistent)
ausearch	Tool to search audit logs
aureport	Generates summary reports
audit.rules	File for runtime or persistent audit rules

Installing auditd

❑Debian/Ubuntu:

- sudo apt install auditd audispd-plugins

❑Enable and start the service:

- sudo systemctl enable --now auditd

❑Check status:

- sudo systemctl status auditd

❑auditd File Locations

File	Purpose
/etc/audit/auditd.conf	Main config file
/etc/audit/rules.d/	Custom audit rule files
/var/log/audit/audit.log	Audit event log

Basic auditd Commands

❑View the Audit Log

- sudo less /var/log/audit/audit.log

❑Search Events Using ausearch

- sudo ausearch -k keyname
- sudo ausearch -k passwd_watch

❑Generate Audit Reports with aureport

❑Show login activity:

- sudo aureport –au

❑Show file access activity:

- sudo aureport -f

Basic auditd Commands

- **Adding Audit Rules**
- **Audit rules tell auditd what to monitor.**
- **Edit or create a rules file:**
  - sudo nano /etc/audit/rules.d/audit.rules
- **Example Rules:**
- **Watch /etc/passwd for write access:**
  - -w /etc/passwd -p wa -k passwd_watch
  - -w: Watch file
  - -p wa: Watch for write and attribute changes
  - -k: Tag the event with a keyword

Basic auditd Commands

- **Monitor use of the sudo command:**
  - -w /usr/bin/sudo -p x -k sudo_exec
- **Watch login/logout logs:**
  - -w /var/log/faillog -p wa -k login_watch
- **After saving rules:**
  - sudo augenrules --load

Log Monitoring and Alerting

Why Monitor Logs?

- Linux logs contain **critical security and system event data**, including:
  - Login attempts
  - SSH activity
  - Authentication failures
  - Service errors
  - File or config changes
- Monitoring these logs **manually** is impractical on busy systems — automation is essential.

What is Logwatch?

- Logwatch is a **log analysis and summarization tool** that scans system logs and **generates daily email reports**. It's not real-time but excellent for **daily system audits**.
- **Installation**
- **Debian/Ubuntu:**
  - sudo apt install logwatch
- **Usage**
- **Run a one-time report:**
  - sudo logwatch --detail High --mailto you@example.com --range today --service all

Configure for Daily Reports

- ❑ **Edit the config file:**
  - sudo nano /etc/logwatch/conf/logwatch.conf
- ❑ **Ensure these options are set:**
  - MailTo = you@example.com
  - Detail = High
  - Range = yesterday
- ❑ **Set up a daily cron job (Logwatch usually configures this automatically on install):**
  - cat /etc/cron.daily/00logwatch
- ❑ **You can customize the services it reports on by editing:**
  - /etc/logwatch/conf/services/

Configure for Daily Reports

- **Example Logwatch Report Topics**
- SSH logins (accepted and denied)
- Sudo usage
- Failed login attempts
- Disk space and usage
- Service restarts

What is Fail2Ban?

- Fail2Ban scans log files for **suspicious patterns** (e.g., failed SSH logins) and **automatically bans offending IPs** using the system firewall.
- **Installation**
- **Debian/Ubuntu:**
  - sudo apt install fail2ban
- **Enable and start:**
  - sudo systemctl enable --now fail2ban

Basic Configuration

- **Copy default config to customize:**
  - sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
- **Edit:**
  - sudo nano /etc/fail2ban/jail.local
- **Example SSH section:**
  - [sshd]
  - enabled = true
  - port = ssh
  - logpath = %(sshd_log)s
  - backend = systemd
  - maxretry = 5
  - bantime = 3600
  - findtime = 600

Monitoring Fail2Ban

- **Check status:**
  - sudo fail2ban-client status
- **Status for specific jail:**
  - sudo fail2ban-client status sshd
- **Unban an IP:**
  - sudo fail2ban-client set sshd unbanip 192.168.0.100
- **Fail2Ban Email Alerts**
  - destemail = you@example.com
  - sender = fail2ban@yourdomain.com
  - mta = sendmail
  - action = %(action_mwl)s

Demo

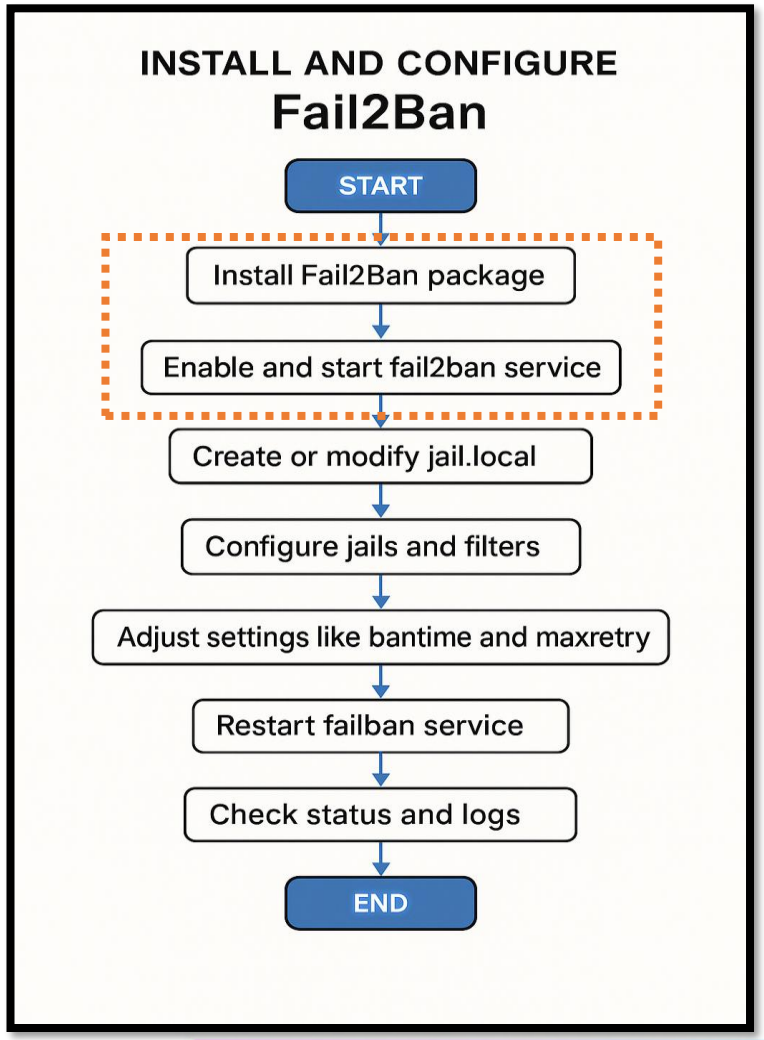
Hands-on Lab: Fail2Ban to Protect SSH

❑Install and Service Startup

- sudo apt update
- sudo apt install fail2ban –y
- sudo systemctl enable fail2ban
- sudo systemctl start fail2ban

❑Fail2Ban loads:

- **fail2ban.conf** (global settings)
- **jail.conf + jail.local** (which jails are enabled, maxretry, bantime, etc.)
- Filter definitions from **/etc/fail2ban/filter.d/** (regex patterns)
- Action definitions from **/etc/fail2ban/action.d/** (iptables, firewalld, sendmail, etc.)



Hands-on Lab

❑Fail2Ban to Protect SSH

❑Objective

- By the end of this lab, you will:
- Install and enable Fail2Ban
- Configure a **jail** to monitor SSH login attempts
- Ban IPs that trigger too many failed login attempts
- Review logs and unban IPs if needed

Step-by-Step [1]

- ❑ **Install Fail2Ban**
- ❑ **Debian/Ubuntu:**
  - sudo apt update
  - sudo apt install fail2ban -y
- ❑ **Enable and Start the Service**
  - sudo systemctl enable --now fail2ban
- ❑ **Verify status:**
  - sudo systemctl status fail2ban

Step-by-Step [2]

- **Create/Modify Jail Configuration for SSH**
- **Best practice: don’t edit the default jail.conf — create jail.local instead.**
  - sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
  - sudo nano /etc/fail2ban/jail.local
- **Find the [sshd] section and configure as follows**
  - [sshd]
  - enabled = true
  - port = ssh
  - logpath = %(sshd_log)s
  - backend = systemd

  - # Ban settings
  - maxretry = 5
  - findtime = 600
  - bantime = 3600

Step-by-Step [3]

❑Restart Fail2Ban to Apply Changes

- sudo systemctl restart fail2ban

❑Check Jail Status

- sudo fail2ban-client status

❑Check specific jail (e.g., sshd):

- sudo fail2ban-client status sshd

❑View Fail2Ban Logs

- sudo cat /var/log/fail2ban.log

❑Or follow in real time:

- sudo tail -f /var/log/fail2ban.log

Step-by-Step [4]

❑Unban a Banned IP

- sudo fail2ban-client set sshd unbanip <IP_ADDRESS>

Hands-on Lab: Lynis

❑Install and Service Startup

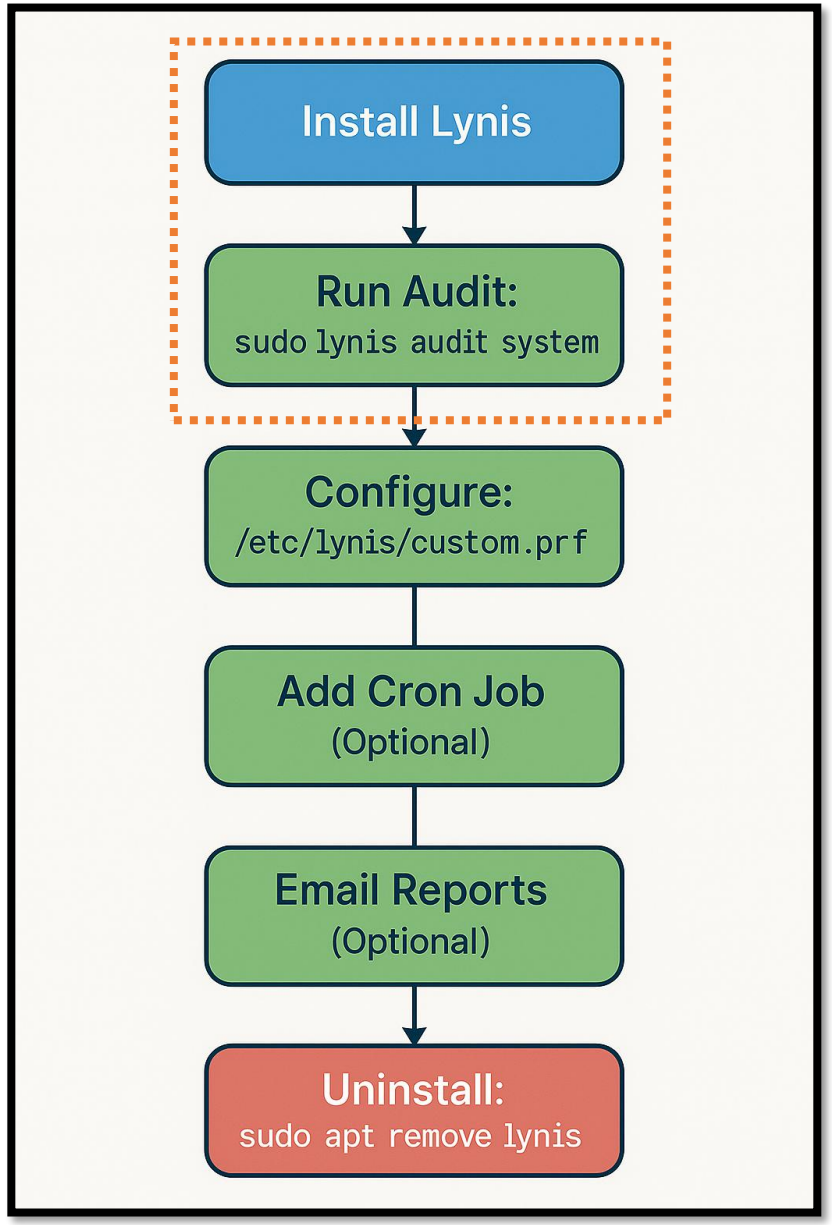
- sudo apt update
- sudo apt install lynis –y
- sudo lynis audit system

❑You will get output like:

- Hardening index score
- Warnings
- Suggestions
- Security recommendations

❑Logs and results are stored here:

- /var/log/lynis.log
- /var/log/lynis-report.dat



Hands-on Lab

- ❑ **Step 1: Install Lynis**
- ❑ **Option 1: Clone from GitHub**
  - ❑ sudo apt update
  - ❑ sudo apt install git -y
  - ❑ git clone https://github.com/CISOfy/lynis
  - ❑ cd lynis
- ❑ **Option 2: Install via Package Manager (Debian-based)**
  - ❑ sudo apt install lynis -y

Hands-on Lab

- **Step 2: Run Lynis Audit**
- Basic System Audit
  - sudo ./lynis audit system
- OR if installed via apt:
  - sudo lynis audit system

- **What Happens During the Audit?**
- **Lynis checks:**
  - Boot and kernel settings
  - User accounts and authentication
  - Services and daemons
  - File system and permission issues
  - Security frameworks (AppArmor, SELinux)
  - Logs and audit frameworks

Hands-on Lab

- **Step 4: Review the Report**
- After the scan, you'll see a summary like:
  - [+] Hardening index : 68 [##############   ]
  - [+] Suggestions   : 18
  - [+] Warning   : 4
- The report is saved at:
  - /var/log/lynis.log
- Suggestions are saved at:
  - /var/log/lynis-report.dat

Hands-on Lab

- **Step 4: Review the Report**
- After the scan, you'll see a summary like:
  - [+] Hardening index : 68 [##############   ]
  - [+] Suggestions   : 18
  - [+] Warning   : 4
- The report is saved at:
  - /var/log/lynis.log
- Suggestions are saved at:
  - /var/log/lynis-report.dat

Hands-on Lab

- ❑ **Step 5: Apply Hardening Measures**
- ❑ Enforce Password Policy
- ❑ Edit /etc/login.defs:
  - PASS_MAX_DAYS 90
  - PASS_MIN_DAYS 10
  - PASS_WARN_AGE 7
- ❑ Enable Auditd
  - sudo apt install auditd -y
  - sudo systemctl enable auditd
  - sudo systemctl start auditd

Hands-on Lab

- **Step 6: Re-run Lynis**
- After applying fixes, rerun the audit:
  - sudo lynis audit system
- Verify that the hardening score improves and that previous warnings are resolved.

Hands-on Lab

- **Step 7: Convert to PDF or HTML**
- Convert to PDF (Linux):
  - a2ps ~/lynis_detailed_report.log -o - | ps2pdf - lynis_report.pdf
- Convert to HTML (example using aha and lynis color output):
  - sudo lynis audit system --color | aha > lynis_report.html
- Install aha first:
  - sudo apt install aha


Q&A