Independent Skill Development Mission

Independent Skill Development Mission NXT
EMPOWERING FUTURES, SHAPING TOMORROW

ISDM (INDEPENDENT SKILL DEVELOPMENT MISSION

MANAGING USERS AND GROUPS IN UNIX

CHAPTER 1: INTRODUCTION TO USER AND GROUP MANAGEMENT

What is User and Group Management in UNIX?

User and group management is a fundamental part of **system administration** in UNIX and Linux. Since UNIX is a **multi-user operating system**, multiple users can access the system simultaneously. To maintain **security, access control, and proper system organization**, users are assigned **unique user accounts** and grouped under **specific groups**.

Proper user and group management helps in:

- **Controlling file access and permissions** – Users can only modify files they have permission for.
- **Enhancing system security** – Prevents unauthorized access to system files.
- **Improving collaboration** – Allows teams to work within shared directories and projects.
- **Managing system resources** – Controls CPU, memory, and disk usage for users.

Each user account is identified by a **User ID (UID)**, and each group has a **Group ID (GID)**. System administrators can **create, modify,**

delete, and manage users and groups using command-line tools like useradd, usermod, groupadd, and passwd.

Example

A company with different departments (HR, IT, and Finance) may assign **user groups** to each department. Employees in the IT group can access development files, while Finance employees are restricted from modifying IT-related files.

Exercise

1. List all users on a UNIX system using the /etc/passwd file.
2. Create a new user and assign them to an existing group.

Case Study: Implementing Access Control in a Corporate IT Infrastructure

A large company needs to **restrict access to financial records** to only authorized personnel. By implementing **user groups and permissions**, system administrators ensure that only employees in the finance group have access to payroll files, preventing unauthorized changes or data breaches.

CHAPTER 2: CREATING AND MANAGING USERS IN UNIX

Creating a New User

Administrators use the useradd command to **create user accounts** in UNIX.

Basic Syntax of useradd

sudo useradd -m -s /bin/bash username

- -m ⟶ Creates a home directory for the user.

- -s /bin/bash ⟶ Assigns the Bash shell as the default shell.

Setting a Password for a User

After creating a user, set a password using:

sudo passwd username

The system will prompt for a new password.

Modifying User Accounts

Use the usermod command to **modify an existing user**.

- **Change a user's home directory:**
- sudo usermod -d /new/home/directory username
- **Lock a user account:**
- sudo usermod -L username
- **Unlock a user account:**
- sudo usermod -U username

Deleting a User

To remove a user account, use:

sudo userdel -r username

- -r ⟶ Deletes the home directory along with the user account.

Example

A company needs to onboard a new employee, **John Doe**, and assign him a home directory with a default shell:

sudo useradd -m -s /bin/bash johndoe

sudo passwd johndoe

Exercise

1. Create a new user and set a password for them.
2. Change the user's default shell to /bin/zsh and verify the change.

Case Study: Automating User Account Management in Cloud Servers

Cloud service providers manage thousands of users. Automating **user creation, modification, and deletion** using scripts prevents manual errors and saves time. In this case study, we explore how cloud administrators automate user management using useradd, usermod, and userdel.

CHAPTER 3: MANAGING GROUPS IN UNIX

Creating and Managing Groups

A **group** in UNIX is a collection of users with shared access to files and resources. Groups help in managing **team collaborations, access control, and security policies**.

Creating a Group

To create a new group, use:

sudo groupadd developers

Adding Users to a Group

A user can be added to a group using usermod:

sudo usermod -aG developers username

- -aG → Appends the user to the group without removing them from other groups.

Viewing Group Membership

To check which groups a user belongs to:

groups username

Or list all groups in the system:

cat /etc/group

Removing a User from a Group

To remove a user from a group:

sudo deluser username groupname

Deleting a Group

To remove a group:

sudo groupdel developers

Example

A project team working on software development creates a group called developers and adds all developers to it:

sudo groupadd developers

sudo usermod -aG developers johndoe

Exercise

1. Create a group called admin_team and add multiple users to it.
2. Remove a user from a group and verify their permissions.

Case Study: Managing Access Control in Multi-User UNIX Environments

A university uses UNIX servers for research and coursework. Professors need access to **grading files**, while students should have access only to **course materials**. By using **user groups**, system administrators enforce role-based access control, improving security and organization.

CHAPTER 4: FILE PERMISSIONS AND OWNERSHIP IN USER AND GROUP MANAGEMENT

Understanding File Ownership

Each file in UNIX has:

1. **Owner** – The user who created the file.
2. **Group** – A group of users who share permissions.
3. **Others** – Any other user on the system.

View file ownership with:

ls -l filename

Example output:

-rw-r--r-- 1 johndoe developers 2048 Feb 24 12:30 report.txt

- **johndoe** → Owner.
- **developers** → Group.
- **Permissions** (rw-r--r--) define who can **read (r), write (w), and execute (x)** the file.

Changing File Ownership and Group

To change a file's owner:

sudo chown newuser filename

To change a file's group:

sudo chgrp newgroup filename

Modifying File Permissions

Use chmod to modify permissions:

chmod 750 filename

- 7 (Owner) = **Read, Write, Execute**
- 5 (Group) = **Read, Execute**
- 0 (Others) = **No access**

Example

A system administrator sets permissions for a confidential document:

sudo chown manager report.txt

sudo chgrp finance report.txt

chmod 640 report.txt

This ensures that **only the manager and finance team can access the file**.

Exercise

1. Create a file and change its owner and group.

2. Modify file permissions to allow only the owner to edit and the group to read.

Case Study: Implementing Secure File Access in a Financial Organization

A financial company stores sensitive payroll data on UNIX servers. Using **strict file permissions (chmod 600) and restricted group access,** administrators prevent unauthorized employees from accessing salary records.

CONCLUSION

Understanding **user and group management in UNIX** is essential for **secure and organized system administration**. Key takeaways include:

✅ Creating, modifying, and deleting **users and groups**.

✅ Assigning **permissions and ownership** to protect files.

✅ Enforcing **access control policies** using group-based permissions.

System Boot Process and Run Levels in UNIX/Linux

Chapter 1: Introduction to the System Boot Process

What is the System Boot Process?

The **system boot process** is the sequence of steps a UNIX/Linux system follows to start up after being powered on. This process involves initializing hardware components, loading the operating system, and preparing the system for user interaction. A smooth and efficient boot process is essential for **system stability, performance, and security**.

The boot process consists of several key stages:

1. **BIOS/UEFI Initialization** – The system firmware performs hardware checks and loads the bootloader.
2. **Bootloader Execution (GRUB/LILO)** – Loads the kernel into memory and hands over control to the OS.
3. **Kernel Initialization** – The Linux kernel initializes hardware and system processes.
4. **Init/Systemd Process Execution** – The first user-space process that manages system initialization.
5. **Runlevel/Target Execution** – Determines which services and processes should start.

Understanding the boot process is essential for **troubleshooting system startup issues**, optimizing performance, and enhancing security.

Example: Diagnosing a System Boot Failure

If a system fails to boot and shows a **kernel panic error,** it may indicate a corrupted kernel or missing system files. Booting into **recovery mode** and checking logs (dmesg) can help identify the issue.

Exercise

1. Restart your system and enter the **GRUB menu**. Identify the available boot options.
2. View the system logs (dmesg or journalctl -b) to analyze the last boot process.

Case Study: Recovering from a Failed Boot in a Production Server

A company's production server crashes and fails to boot due to a corrupted **GRUB bootloader**. By booting from a **live Linux USB,** system administrators restore the bootloader and recover the system without losing data.

CHAPTER 2: DETAILED STEPS OF THE BOOT PROCESS

Step 1: BIOS/UEFI Initialization

When a system is powered on, the **BIOS (Basic Input/Output System)** or **UEFI (Unified Extensible Firmware Interface)** initializes the hardware.

- **BIOS/UEFI Responsibilities:**
  - Performs **Power-On Self Test (POST)** to check CPU, RAM, and disks.
  - Identifies available **boot devices** (HDD, SSD, USB).
  - Loads the **bootloader** from the primary boot device.

If the BIOS/UEFI cannot find a valid boot device, it displays an error like:

No bootable device found.

Example: Changing the Boot Order in BIOS

To boot from a USB device, enter BIOS (F2, F12, Del, or Esc during startup) and change the **boot order** to prioritize USB.

Exercise

1. Access your system's BIOS/UEFI settings and locate the **boot device priority** menu.
2. Change the boot order and boot the system from a USB drive.

Case Study: Recovering a System by Changing Boot Order

A company's laptop fails to start because the boot order was accidentally changed to **Network Boot** instead of **HDD**. By restoring the correct boot order in BIOS, the issue is resolved quickly.

CHAPTER 3: BOOTLOADER EXECUTION (GRUB/LILO)

What is a Bootloader?

A **bootloader** is a small program responsible for **loading the operating system kernel** into memory. The most common bootloaders in UNIX/Linux are:

- **GRUB (GRand Unified Bootloader)** – Used in most Linux distributions.
- **LILO (Linux Loader)** – An older bootloader, now rarely used.

Functions of a Bootloader:

1. **Loads the Kernel** – Locates and loads the kernel file (vmlinuz).
2. **Provides Boot Options** – Allows users to select different operating systems or kernels.
3. **Passes Boot Parameters** – Sends configuration options to the kernel.

Example: Editing GRUB Boot Parameters

If the system is stuck in a boot loop, you can edit the GRUB menu:

1. Press Esc or Shift during boot to open the GRUB menu.
2. Select a boot entry and press e to edit.
3. Modify the linux line (e.g., add single for single-user mode).
4. Press Ctrl + X to boot with the modified settings.

Exercise

1. Open the GRUB bootloader menu and view available boot entries.
2. Modify a GRUB entry to boot into **single-user mode**.

Case Study: Fixing a Corrupted GRUB Bootloader

A system upgrade corrupts the GRUB bootloader, causing the system to display a **grub rescue prompt**. By using a **Live CD** and running grub-install, administrators restore the bootloader and successfully boot the system.

CHAPTER 4: KERNEL INITIALIZATION

Role of the Kernel in the Boot Process

The **kernel** is the core component of the operating system, responsible for:

- **Initializing hardware drivers** (CPU, RAM, storage).
- **Mounting the root filesystem** (/).
- **Starting system processes (init/systemd)**.

If the kernel fails to load, the system may display a **kernel panic** error, requiring manual intervention.

Example: Checking Kernel Messages After Boot

Use the following command to view kernel logs:

dmesg | less

This helps diagnose **hardware failures or missing drivers**.

Exercise

1. Check the currently running kernel version using:
2. uname -r
3. List all available kernels using:
4. ls /boot/vmlinuz*

Case Study: Resolving a Kernel Panic on a Server

A new system update causes a **kernel panic**, preventing the server from booting. The administrator boots into an **older working kernel from GRUB**, restores the broken kernel, and ensures system stability.

Chapter 5: Init/Systemd Process Execution

What is Init/Systemd?

Once the kernel loads, it starts the first user-space process:

- **SysV Init (init)** – The traditional init system, used in older UNIX/Linux versions.
- **Systemd (systemctl)** – The modern init system, offering faster boot times and improved process management.

To check the system's init process:

ps -p 1

Output for Systemd:

PID TTY TIME CMD

1 ? 00:01:23 systemd

Managing System Services with Systemd

Use systemctl to manage services:

- **Check running services:**
- systemctl list-units --type=service
- **Start/Stop a service:**
- systemctl start apache2
- systemctl stop apache2
- **Enable a service at boot:**
- systemctl enable ssh

Example: Restarting a Service After Boot

If the networking service fails to start after boot, restart it manually:

systemctl restart networking

Exercise

1. Check which services are enabled to start at boot using:
2. systemctl list-unit-files --type=service | grep enabled
3. Disable an unnecessary service and verify its status.

Case Study: Fixing a Slow Boot Issue Caused by a Failing Service

A company experiences slow boot times. Checking the system logs (journalctl -b), administrators find that a **failing service is delaying startup**. By disabling the service, boot time is significantly improved.

CONCLUSION

Understanding the **system boot process and run levels** is essential for:

- **Troubleshooting boot failures** (GRUB errors, kernel panics).
- **Optimizing startup times** by managing services.
- **Enhancing system security** through bootloader configurations.

PACKAGE MANAGEMENT IN UNIX/LINUX (YUM, APT, RPM)

CHAPTER 1: INTRODUCTION TO PACKAGE MANAGEMENT IN UNIX/LINUX

What is Package Management?

Package management is the process of **installing, updating, removing, and managing software packages** on a UNIX/Linux system. Since UNIX/Linux distributions rely heavily on package managers, understanding them is essential for **system administration, software deployment, and maintenance**.

A package manager:

- **Automates software installation and dependency resolution.**
- **Manages package versions and updates securely.**
- **Ensures system stability by preventing broken dependencies.**
- **Verifies package integrity using cryptographic signatures.**

There are two main types of package management systems:

1. **Debian-based (Debian, Ubuntu, etc.)** – Uses dpkg and apt package managers.
2. **Red Hat-based (RHEL, CentOS, Fedora, etc.)** – Uses RPM and YUM/DNF package managers.

These package managers retrieve software from **repositories**, which are official or third-party servers hosting software packages.

Example: Installing Software with a Package Manager

To install the wget utility on an Ubuntu system:

sudo apt install wget

On a Red Hat-based system:

sudo yum install wget

Exercise

1. Find out which package manager your system uses (apt, yum, or dnf).
2. List all installed packages on your system.

Case Study: Automating System Updates in a Large IT Environment

A company with **hundreds of Linux servers** automates software updates using package managers (apt and yum). By scheduling regular updates, the company ensures **system security and stability** while reducing manual workload.

CHAPTER 2: RPM PACKAGE MANAGEMENT (RED HAT-BASED SYSTEMS)

What is RPM?

The **RPM Package Manager (RPM)** is the standard package format for Red Hat-based distributions, including:

- **RHEL (Red Hat Enterprise Linux)**
- **CentOS**

- **Fedora**

RPM packages have a .rpm extension and contain **software binaries, dependencies, and metadata**.

Using RPM Commands

RPM does not resolve dependencies automatically. Instead, it installs **only the specified package**, requiring manual installation of dependencies.

1. Installing an RPM Package

sudo rpm -ivh package.rpm

- -i ⟶ Install package.
- -v ⟶ Verbose mode.
- -h ⟶ Show progress with hash marks.

2. Upgrading an RPM Package

sudo rpm -Uvh package.rpm

This upgrades an existing package or installs a new one if it's not present.

3. Removing an RPM Package

sudo rpm -e package-name

4. Querying Installed RPM Packages

- List all installed packages:
- rpm -qa
- Find information about a specific package:
- rpm -qi package-name

Example: Installing and Querying an RPM Package

1. Download an RPM package:
2. wget http://mirror.centos.org/centos/7/updates/x86_64/Packages/wget-1.14-18.el7.x86_64.rpm
3. Install the package:
4. sudo rpm -ivh wget-1.14-18.el7.x86_64.rpm
5. Verify the installation:
6. rpm -qa | grep wget

Exercise

1. Install an RPM package and verify its details using rpm -qi.
2. Remove an installed package and confirm its removal.

Case Study: Using RPM for Offline Software Installation in Secure Environments

A **high-security government network** requires **offline package installation** due to restricted internet access. Administrators use rpm to manually install and upgrade software without needing external repositories.

CHAPTER 3: YUM PACKAGE MANAGEMENT (YELLOWDOG UPDATER, MODIFIED)

What is YUM?

YUM (Yellowdog Updater, Modified) is a package manager for Red Hat-based systems that **automatically resolves dependencies**, unlike RPM. It retrieves software from **configured repositories**.

Basic YUM Commands

1. Installing a Package

sudo yum install package-name

2. Removing a Package

sudo yum remove package-name

3. Updating All Packages

sudo yum update

4. Searching for a Package

yum search package-name

5. Viewing Package Information

yum info package-name

Example: Installing and Managing a Package with YUM

1. Install the vim package:
2. sudo yum install vim
3. Check the installed version:
4. rpm -q vim
5. Remove the package:
6. sudo yum remove vim

Exercise

1. Search for a package available in YUM repositories and install it.
2. List all installed packages and identify recently installed ones.

Case Study: Automating Patch Management with YUM

A financial institution schedules **automated updates** using yum-cron to ensure all servers receive **security patches and software updates,** reducing vulnerabilities.

CHAPTER 4: APT PACKAGE MANAGEMENT (ADVANCED PACKAGE TOOL)

What is APT?

APT (Advanced Package Tool) is the package manager for **Debian-based systems,** including:

- **Debian**
- **Ubuntu**
- **Linux Mint**

APT simplifies software management by **handling dependencies automatically** and retrieving packages from online repositories.

Basic APT Commands

1. Updating the Package Repository

sudo apt update

2. Installing a Package

sudo apt install package-name

3. Removing a Package

sudo apt remove package-name

4. Upgrading All Installed Packages

sudo apt upgrade

5. Searching for a Package

apt search package-name

Example: Managing Packages with APT

1. Update repositories:
2. sudo apt update
3. Install the curl package:
4. sudo apt install curl
5. Remove the package:
6. sudo apt remove curl

Exercise

1. Search for an available package and install it using apt.
2. Upgrade all installed packages and verify the changes.

Case Study: Managing Large-Scale Ubuntu Deployments with APT

A cloud service provider manages **thousands of Ubuntu servers** using **APT automation tools** to deploy, update, and secure software efficiently across the infrastructure.

CHAPTER 5: COMPARISON OF RPM, YUM, AND APT

Feature	RPM	YUM	APT
Distribution	Red Hat-based (RHEL, CentOS, Fedora)	Red Hat-based (RHEL, CentOS, Fedora)	Debian-based (Ubuntu, Debian, Mint)
Dependency Resolution	No (manual)	Yes (automatic)	Yes (automatic)
Installation Command	rpm -ivh package.rpm	yum install package-name	apt install package-name
Package Removal	rpm -e package-name	yum remove package-name	apt remove package-name
Update All Packages	Not applicable	yum update	apt upgrade

Exercise

1. Compare package managers by installing the same software using **RPM, YUM, and APT**.
2. Write a script to **automate system updates using a package manager**.

Case Study: Choosing the Right Package Manager for Enterprise Systems

A tech company decides between **RPM-based (RHEL) and APT-based (Ubuntu) distributions** for its servers. After evaluating package management features, they choose Ubuntu with APT for **ease of automation and dependency handling**.

CONCLUSION

This guide covered:

- **Using RPM, YUM, and APT for package management.**
- **Installing, updating, and removing software securely.**
- **Choosing the right package manager for different environments.**

DISK MANAGEMENT AND PARTITIONING IN UNIX/LINUX

CHAPTER 1: INTRODUCTION TO DISK MANAGEMENT AND PARTITIONING

What is Disk Management and Partitioning?

Disk management in UNIX/Linux refers to the process of **configuring, maintaining, and optimizing storage devices**. Proper disk management ensures **efficient use of disk space, data security, and system performance**.

Partitioning is the process of **dividing a physical disk into multiple sections (partitions)**, allowing for better organization and separation of system files, user data, and application files. Partitions enable:

- **Efficient disk space utilization**
- **Isolation of system and user data**
- **Multi-boot system configurations**
- **Improved security and backup strategies**

Types of Disk Partitions

1. **Primary Partition** – A bootable partition that holds the operating system.
2. **Extended Partition** – A special partition that acts as a container for logical partitions.
3. **Logical Partition** – A sub-partition inside an extended partition, used for data storage.

Understanding **disk partitioning schemes** such as **MBR (Master Boot Record)** and **GPT (GUID Partition Table)** is essential for proper disk management.

Example: Viewing Disk Partitions in Linux

Use the following command to check disk partitions:

lsblk

It displays:

```
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0  500G  0 disk
 ├──sda1   8:1    0  100G  0 part /
 ├──sda2   8:2    0  200G  0 part /home
 └──sda3   8:3    0  200G  0 part /var
```

Exercise

1. Use lsblk and fdisk -l to list partitions on your system.
2. Identify the partition where your Linux OS is installed.

Case Study: Optimizing Disk Partitioning for a Database Server

A financial company runs large databases. By **separating system, database, and log files into different partitions**, administrators optimize **performance and prevent data corruption** in case of a crash.

CHAPTER 2: PARTITIONING A DISK IN LINUX

Understanding Partitioning Tools in UNIX/Linux

To partition a disk in UNIX/Linux, administrators use tools such as:

- **fdisk** – Command-line tool for MBR partitioning.
- **parted** – Used for GPT and MBR partitioning.
- **gparted** – GUI-based tool for partition management.

Using fdisk to Create a New Partition

1. **List Available Disks**
2. sudo fdisk -l
3. **Select the Disk to Partition**
4. sudo fdisk /dev/sdb
5. **Create a New Partition**
    - Press n ⟶ Create new partition.
    - Choose primary or logical.
    - Specify the partition size.
    - Press w ⟶ Write changes and exit.
6. **Format the New Partition**
7. sudo mkfs.ext4 /dev/sdb1
8. **Mount the Partition**
9. sudo mkdir /mnt/newdisk
10. sudo mount /dev/sdb1 /mnt/newdisk
11. **Make the Mount Permanent**
    Edit /etc/fstab to automatically mount the partition at boot:

12. echo "/dev/sdb1 /mnt/newdisk ext4 defaults 0 0" | sudo tee -a /etc/fstab

Example: Creating and Formatting a Partition on a New Disk

If a new **500GB disk (/dev/sdc)** is added to a server, you can partition and format it as follows:

sudo fdisk /dev/sdc

Create new partition → `n`, select partition type → `p`, save → `w`

sudo mkfs.ext4 /dev/sdc1

sudo mount /dev/sdc1 /mnt/storage

Exercise

1. Create a new partition on an unallocated disk and mount it.
2. Verify partition creation using lsblk and df -h.

Case Study: Partitioning a Disk for Web Hosting Servers

A web hosting provider **separates system files, website data, and logs** into different partitions. This prevents log files from filling up the root partition, ensuring **server uptime and stability**.

CHAPTER 3: FILESYSTEM MANAGEMENT IN UNIX/LINUX

What is a Filesystem?

A filesystem is the **method used to store and organize data on a partition**. Different filesystems provide varying levels of performance, security, and features.

Common Filesystem Types

Filesystem	Description
ext4	Default Linux filesystem with journaling.
XFS	High-performance filesystem, used for large-scale storage.
NTFS	Windows filesystem, supported in Linux via ntfs-3g.
FAT32	Universal filesystem, compatible with all OS.
Btrfs	Advanced Linux filesystem with snapshots and compression.

Managing Filesystems

- **Check Disk Usage**
- df -h
- **Check Filesystem Type**
- sudo blkid /dev/sdb1
- **Convert an Existing Filesystem** (Example: Ext4 to XFS)
- sudo mkfs.xfs /dev/sdb1

Example: Formatting and Mounting a Filesystem

sudo mkfs.ext4 /dev/sdb1

sudo mkdir /mnt/data

sudo mount /dev/sdb1 /mnt/data

Exercise

1. Check the filesystem type of all partitions using lsblk -f.

2. Convert a partition to XFS and mount it.

Case Study: Choosing the Right Filesystem for a Cloud Storage Provider

A cloud storage provider selects **XFS** for large file storage due to its **fast performance and scalability,** ensuring high-speed access to massive datasets.

CHAPTER 4: MANAGING DISK SPACE AND STORAGE OPTIMIZATION

Monitoring Disk Usage

- **Check Free Space:**
- df -h
- **Find Large Files:**
- sudo du -sh /var/log/*
- **Check Inode Usage:**
- df -i

Expanding Storage: Resizing Partitions

Resizing an Ext4 Partition

1. **Unmount the Partition:**
2. sudo umount /dev/sdb1
3. **Resize the Partition:**
4. sudo resize2fs /dev/sdb1

Example: Increasing Storage for a Full Disk Partition

If /var is full, resize its logical volume:

sudo lvextend -l +100%FREE /dev/mapper/centos-var

sudo resize2fs /dev/mapper/centos-var

Exercise

1. Identify which partition is using the most space.
2. Resize an existing partition and verify the change.

Case Study: Preventing Server Downtime Due to Full Disk Issues

A company experiences downtime due to a **full root partition**. By **expanding the partition dynamically,** they prevent system crashes and ensure **continuous uptime**.

CONCLUSION

This guide covered:

- **Creating and managing disk partitions.**
- **Formatting and mounting filesystems.**
- **Monitoring and optimizing disk space.**

MOUNTING FILE SYSTEMS AND NFS IN UNIX/LINUX

CHAPTER 1: INTRODUCTION TO MOUNTING FILE SYSTEMS

What is Mounting in UNIX/Linux?

Mounting is the process of **attaching a storage device or partition to a directory in the UNIX/Linux filesystem** so that users and applications can access it. Unlike Windows, where drives have separate letters (C:, D:), UNIX/Linux integrates all devices into a **single directory structure**.

A mounted filesystem allows:

- **Accessing external storage devices (USB drives, hard disks, network drives)**.
- **Expanding system storage by mounting additional partitions**.
- **Sharing resources between multiple users or systems via NFS (Network File System)**.

Key Mounting Concepts

1. **Mount Point** – A directory where the filesystem is attached (e.g., /mnt, /media).
2. **Filesystem Type** – Determines the structure and rules of data storage (ext4, XFS, NTFS, NFS).
3. **Mounting and Unmounting** – Attaching and detaching a filesystem using mount and umount commands.

Example: Checking Mounted File Systems

To see all mounted filesystems:

mount | column -t

Or use:

df -h

This displays:

Filesystem Size Used Avail Use% Mounted on

/dev/sda1 100G 45G 50G 47% /

/dev/sdb1 500G 120G 350G 25% /mnt/storage

Exercise

1. List all mounted filesystems on your system using df -h.
2. Identify the filesystem type of a mounted partition using blkid.

Case Study: Expanding Storage for a Web Server

A company's web server runs out of space. Instead of reinstalling the system, administrators **attach a new hard disk and mount it as /var/www**, expanding web storage without downtime.

CHAPTER 2: MANUALLY MOUNTING AND UNMOUNTING FILE SYSTEMS

Mounting a File System

To manually mount a filesystem, use the mount command:

sudo mount -t ext4 /dev/sdb1 /mnt/storage

- -t ext4 → Specifies the filesystem type.

- /dev/sdb1 → The partition being mounted.
- /mnt/storage → The mount point.

Making the Mount Permanent

To auto-mount at boot, add an entry to /etc/fstab:

```
/dev/sdb1  /mnt/storage  ext4  defaults  0  2
```

After modifying /etc/fstab, apply changes:

```
sudo mount -a
```

Unmounting a File System

Before removing a device, unmount it:

```
sudo umount /mnt/storage
```

If the device is busy, force unmount:

```
sudo umount -l /mnt/storage
```

Example: Mounting a USB Drive Manually

If a USB drive is detected as /dev/sdc1:

```
sudo mkdir /mnt/usb
sudo mount /dev/sdc1 /mnt/usb
ls /mnt/usb  # Verify access
```

Exercise

1. Mount a newly attached storage device and verify its contents.
2. Unmount the device and check if it's successfully removed.

Case Study: Mounting External Storage for Backup Servers

A backup server needs additional space for archiving files. Administrators **connect an external 1TB disk and mount it as /backup,** ensuring smooth daily backups.

CHAPTER 3: INTRODUCTION TO NETWORK FILE SYSTEM (NFS)

What is NFS?

The **Network File System (NFS)** allows UNIX/Linux systems to **share directories and files over a network**. It enables remote access to files as if they were local, making it useful for:

- **Centralized file storage** – Shared home directories and project files.
- **Efficient data management** – Allowing multiple systems to access shared resources.
- **Cloud and distributed computing** – Mounting remote filesystems in enterprise networks.

How NFS Works

- **NFS Server** – Shares a directory over the network.
- **NFS Client** – Mounts the shared directory and accesses files remotely.
- **NFS Protocol** – Uses TCP/UDP for communication, commonly over port 2049.

Example: Checking NFS Services on a System

Verify if NFS is running:

```
sudo systemctl status nfs-server
```

Check active NFS shares:

showmount -e <server-ip>

Exercise

1. Find out if your system has nfs-utils or nfs-common installed.
2. Identify available NFS shares on a remote server.

Case Study: Using NFS for Multi-User Data Access in a Research Lab

A university research lab needs a **shared storage solution** for all researchers. By setting up **an NFS server,** all lab computers can access research data in real-time without duplication.

CHAPTER 4: SETTING UP AN NFS SERVER AND CLIENT

Configuring an NFS Server

1. **Install NFS Server Packages**
2. sudo apt install nfs-kernel-server # Debian-based
3. sudo yum install nfs-utils # Red Hat-based
4. **Create a Shared Directory**
5. sudo mkdir -p /srv/nfs/shared
6. sudo chown nobody:nogroup /srv/nfs/shared
7. **Edit the NFS Exports File**
   Add the following line to /etc/exports:
8. /srv/nfs/shared 192.168.1.0/24(rw,sync,no_root_squash)

This allows read/write access for clients in the 192.168.1.x network.

9. **Restart NFS Server**
10. sudo systemctl restart nfs-server
11. sudo exportfs -rav

Configuring an NFS Client

1. **Install NFS Client Packages**
2. sudo apt install nfs-common # Debian-based
3. sudo yum install nfs-utils # Red Hat-based
4. **Mount the NFS Share**
5. sudo mount -t nfs 192.168.1.100:/srv/nfs/shared /mnt/nfs
6. **Make the Mount Persistent**
   Add to /etc/fstab:
7. 192.168.1.100:/srv/nfs/shared /mnt/nfs nfs defaults 0 0

Example: Verifying NFS Connectivity

To confirm successful NFS mounting:

ls /mnt/nfs

Exercise

1. Set up a temporary NFS share on your local machine and access it from another system.
2. Configure /etc/fstab to auto-mount an NFS share at boot.

Case Study: Using NFS for Distributed Software Development

A software development company sets up **an NFS server** to store source code. Developers across multiple locations mount the **same shared directory,** enabling real-time collaboration.

Chapter 5: Managing and Troubleshooting Mounting and NFS Issues

Common Mounting Issues and Fixes

1. **"Device is busy" error when unmounting**
    - Find active processes using the mount:
    - sudo lsof /mnt/storage
    - Kill the process and retry:
    - sudo umount -l /mnt/storage
2. **NFS mount hangs or fails**
    - Check if the NFS server is running:
    - sudo systemctl status nfs-server
    - Verify firewall rules:
    - sudo ufw allow from 192.168.1.0/24 to any port 2049

Example: Debugging NFS Mount Failures

To check NFS logs for errors:

journalctl -xe | grep nfs

Exercise

1. Simulate an NFS failure and troubleshoot it.

2. Verify network connectivity between an NFS server and a client.

Case Study: Resolving NFS Access Issues in a Production Environment

A company's shared NFS directory **becomes inaccessible** due to a firewall update. By checking logs and updating firewall rules, administrators restore access without downtime.

CONCLUSION

This guide covered:

- **Mounting local and network file systems.**
- **Setting up and managing NFS shares.**
- **Troubleshooting common mounting and NFS issues.**

PROCESS AND JOB SCHEDULING IN UNIX/LINUX

CHAPTER 1: INTRODUCTION TO PROCESS AND JOB SCHEDULING

What is Process and Job Scheduling?

In UNIX/Linux, **process and job scheduling** refers to the **management and execution of processes and tasks** at different times based on user or system requirements. A **process** is an instance of a running program, while **job scheduling** allows users to execute tasks at a specified time automatically.

Efficient process and job scheduling is crucial for:

- **Managing system performance** by prioritizing tasks.
- **Automating repetitive tasks** like backups, updates, and maintenance.
- **Optimizing CPU usage** by distributing workloads efficiently.
- **Running background processes** without user intervention.

The UNIX/Linux kernel manages processes and job scheduling using process states, job control commands (fg, bg, jobs), and scheduling tools like cron and at.

Example: Listing Running Processes

Use the ps command to list active processes:

ps aux

This displays:

USER PID %CPU %MEM COMMAND

root 1234 0.1 2.5 /usr/bin/apache2

john 5678 1.2 1.0 /usr/bin/firefox

Exercise

1. Use ps -ef and top to analyze active processes.
2. Identify the highest CPU-consuming process using top.

Case Study: Managing System Performance in a High-Traffic Web Server

A company's web server slows down due to **high CPU usage**. By analyzing process scheduling and setting priority levels (nice, renice), administrators optimize performance by reducing CPU allocation for background processes.

CHAPTER 2: UNDERSTANDING PROCESS MANAGEMENT IN UNIX/LINUX

Process Lifecycle and States

Each process in UNIX/Linux has a **lifecycle** and goes through different states:

1. **Created (New)** – The process is created but not yet running.
2. **Ready** – The process is waiting for CPU time.
3. **Running** – The process is executing instructions.
4. **Waiting (Sleeping)** – The process is waiting for an event (e.g., I/O operation).
5. **Terminated (Zombie)** – The process has completed but still occupies memory.

Use the following command to check process states:

ps -eo pid,stat,cmd

- R ⟶ Running
- S ⟶ Sleeping
- Z ⟶ Zombie

Example: Killing a Process Using kill

Find and terminate a process with:

ps aux | grep firefox

kill 5678

Exercise

1. Create a process using sleep 100 & and list it using ps -ef.
2. Terminate the process using kill and verify its status.

Case Study: Handling Unresponsive Applications in Enterprise Systems

A financial institution runs **real-time stock trading applications**. When a process hangs, administrators use kill -9 to force termination and ensure **continuous system availability**.

CHAPTER 3: JOB SCHEDULING AND BACKGROUND PROCESSES

Managing Background and Foreground Jobs

A job can run **in the foreground** (requiring user interaction) or **in the background** (running independently).

Running a Process in the Background

Append & to a command:

sleep 100 &

Check background jobs using:

jobs

Bringing Jobs to Foreground and Background

Move a job from background to foreground:

fg %1

Send a job to the background:

bg %1

Example: Running a Long Task in the Background

If a user is downloading a large file:

wget https://example.com/largefile.iso &

This allows the user to continue working while the download progresses in the background.

Exercise

1. Start a process in the background and bring it to the foreground using fg.
2. List all running background jobs and terminate one.

Case Study: Running Multiple Computational Jobs on a Research Cluster

A research institute runs **machine learning models** that take hours to complete. By running them in the background using nohup and bg, scientists can **log out while the processes continue running**.

CHAPTER 4: SCHEDULING TASKS WITH CRON JOBS

What is cron?

The cron daemon automates **scheduled execution of tasks**. It reads the **crontab (cron table),** which contains scheduled jobs for users and the system.

Creating a Cron Job

1. Open the crontab editor:
2. crontab -e
3. Add a cron job to run a backup script daily at midnight:
4. 0 0 * * * /home/user/backup.sh
5. Save and exit.

Understanding Cron Syntax

```
* * * * * command_to_execute
| | | | |
| | | |  └──── Day of the week (0-7, 0=Sunday)
| | |  └────── Month (1-12)
| |  └──────── Day of the month (1-31)
|  └────────── Hour (0-23)
```

└──────────────────────── Minute (0-59)

Common Cron Job Examples

Task	Cron Syntax
Run every 10 minutes	*/10 * * * * /script.sh
Run every Monday at 5 AM	0 5 * * 1 /script.sh
Run on the 1st of every month	0 0 1 * * /script.sh

Example: Scheduling a Disk Cleanup Task

0 3 * * 0 rm -rf /var/log/*.log

This deletes log files **every Sunday at 3 AM**.

Exercise

1. Schedule a script to run **every 5 minutes**.
2. Modify a cron job to execute a script **at 10 PM daily**.

Case Study: Automating Server Updates with cron

A cloud provider schedules **automated system updates** at midnight using cron jobs, ensuring that all servers receive **critical security patches without manual intervention**.

CHAPTER 5: ONE-TIME JOB SCHEDULING WITH AT AND BATCH

Using at for One-Time Jobs

The at command schedules a **one-time execution** of a command.

Example: Running a Task at a Specific Time

1. Open the at prompt:

2. at 5:30 PM
3. Enter the command:
4. echo "System update" >> /var/log/sysupdate.log
5. Press Ctrl + D to save.

Viewing and Managing Scheduled at Jobs

- List pending jobs:
- atq
- Remove a scheduled job:
- atrm job_id

Using batch for System Load-Based Scheduling

The batch command runs jobs when system load is low.

echo "tar -czf backup.tar.gz /home/user" | batch

Example: Scheduling a Database Backup with at

at midnight

mysqldump -u root -p database > /backup/db.sql

Exercise

1. Schedule a task to **run 30 minutes from now** using at.
2. Use batch to schedule a job when CPU usage is low.

Case Study: Scheduling a One-Time Job for Software Installation

A system administrator uses at to install **security updates outside working hours,** ensuring minimal disruption to services.

CONCLUSION

This guide covered:

- ✅ **Managing processes (ps, kill, jobs).**
- ✅ **Running background tasks (bg, fg, nohup).**
- ✅ **Automating recurring jobs (cron) and one-time tasks (at).**

System Logs and Monitoring in UNIX/Linux

Chapter 1: Introduction to System Logs and Monitoring

What are System Logs and Monitoring?

System logs and monitoring in UNIX/Linux play a crucial role in **troubleshooting, security auditing, and performance optimization**. Logs record system activities, user actions, and hardware or software issues, while monitoring tools help track system health in real time.

System logs and monitoring help in:

- **Detecting system errors and failures**
- **Analyzing security incidents**
- **Optimizing performance and resource utilization**
- **Troubleshooting software and hardware issues**

Log files in UNIX/Linux are stored in /var/log/, and monitoring tools like top, htop, vmstat, and dstat provide **real-time insights** into system performance.

Example: Viewing System Logs

To check system logs, use:

sudo cat /var/log/syslog

Or, to view logs dynamically:

sudo tail -f /var/log/syslog

Exercise

1. Check the latest system logs using tail -f /var/log/syslog.
2. Identify login attempts using grep "sshd" /var/log/auth.log.

Case Study: Investigating System Crashes with Logs

A company's database server crashes unexpectedly. By analyzing /var/log/messages and /var/log/syslog, administrators identify **high memory usage as the root cause,** preventing future crashes.

CHAPTER 2: UNDERSTANDING SYSTEM LOGS IN UNIX/LINUX

Common System Log Files

Log File	Purpose
/var/log/syslog	General system messages (Debian-based)
/var/log/messages	General system messages (RHEL-based)
/var/log/auth.log	User authentication logs
/var/log/secure	Security-related logs
/var/log/dmesg	Kernel messages at boot
/var/log/cron	Cron job execution logs
/var/log/httpd/access.log	Apache web server access logs
/var/log/httpd/error.log	Apache error logs

Checking System Logs

1. View logs using cat, less, or tail:

2. sudo less /var/log/syslog
3. Filter logs for specific events:
4. grep "error" /var/log/syslog
5. View authentication logs:
6. sudo cat /var/log/auth.log

Example: Checking User Login Attempts

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

This helps detect unauthorized login attempts.

Exercise

1. Find the last failed login attempt in /var/log/auth.log.
2. Check kernel messages using dmesg | tail -20.

Case Study: Investigating Unauthorized Login Attempts

A company notices multiple failed SSH login attempts. By analyzing /var/log/auth.log, they identify a brute force attack and secure the system using **fail2ban**.

CHAPTER 3: REAL-TIME SYSTEM MONITORING TOOLS

Monitoring System Performance

Several UNIX/Linux tools help monitor system performance:

- **top** – Displays real-time CPU and memory usage.
- **htop** – Interactive version of top with color-coded output.
- **vmstat** – Monitors CPU, memory, and disk usage.

- **iostat** – Monitors disk I/O statistics.
- **netstat** – Displays network connections.

Using the top Command

The top command provides a **dynamic view** of system resources:

top

- PID – Process ID
- %CPU – CPU usage
- %MEM – Memory usage
- TIME+ – Total CPU time used

To sort processes by memory usage, press M.

Using htop for Interactive Monitoring

Install htop (if not installed):

sudo apt install htop # Debian-based

sudo yum install htop # RHEL-based

Run htop for a user-friendly process monitoring interface.

Example: Monitoring CPU and Memory Usage with vmstat

vmstat 5

This updates CPU, memory, and I/O stats every 5 seconds.

Exercise

1. Run top and identify the process using the most CPU.
2. Use htop to find processes consuming the most memory.

Case Study: Identifying Performance Bottlenecks in a Web Server

A company's web application runs slowly. Using top and htop, administrators find **Apache processes consuming excessive memory**. Restarting the service restores normal performance.

CHAPTER 4: NETWORK AND DISK MONITORING

Monitoring Network Activity

To check active network connections:

netstat -tunlp

To monitor live network traffic:

sudo iftop

Checking Disk Usage with df and du

- **View free disk space**:
- df -h
- **Find the largest files in a directory**:
- sudo du -sh /var/*

Example: Identifying Disk Space Issues

df -h | grep '/dev/sda'

This helps detect **full partitions** that need cleanup.

Exercise

1. List the top 10 largest files in /var/log.
2. Monitor live network connections using netstat.

Case Study: Preventing Disk Overflow in Database Servers

A database server crashes due to **a full /var/log partition**. Using df -h and du -sh, administrators clear old logs and schedule **automatic log rotation**.

CHAPTER 5: LOG MANAGEMENT AND AUTOMATION

Rotating Logs with Logrotate

Log files grow over time and must be managed efficiently. **Logrotate** automatically rotates, compresses, and removes old logs.

Configuring Log Rotation

Check existing log rotation rules:

```
cat /etc/logrotate.conf
```

To rotate logs daily, edit /etc/logrotate.d/custom:

```
/var/log/custom.log {
    daily
    rotate 7
    compress
    missingok
    notifempty
}
```

This rotates logs daily, keeps 7 copies, and compresses old logs.

Automating Monitoring with monit

monit automatically restarts failed services. Install it using:

```
sudo apt install monit  # Debian-based
sudo yum install monit  # RHEL-based
```

Configure monit to monitor Apache:

```
check process apache with pidfile /var/run/apache2.pid
   start program = "/etc/init.d/apache2 start"
   stop program  = "/etc/init.d/apache2 stop"
```

Restart monit:

```
sudo systemctl restart monit
```

Example: Automating Log Cleanup

Schedule log cleanup with cron:

```
0 3 * * * sudo find /var/log -name "*.log" -mtime +7 -delete
```

This deletes logs older than 7 days **every night at 3 AM**.

Exercise

1. Configure log rotation for a custom log file.
2. Set up a cron job to clear logs every week.

Case Study: Automating System Health Monitoring in a Cloud Infrastructure

A cloud provider uses **Logrotate for log management** and monit to **restart failed services automatically,** ensuring **99.99% uptime**.

CONCLUSION

This guide covered:

- ✅ **Understanding system logs (syslog, auth.log, dmesg).**
- ✅ **Real-time monitoring with top, htop, vmstat.**
- ✅ **Network and disk monitoring (netstat, df, du).**
- ✅ **Automating log management (logrotate, monit).**

Assignment Solution: Setting Up and Managing Users with Different Access Levels in UNIX/Linux

Objective

This assignment provides a step-by-step guide to **create, modify, and manage users with different access levels** in UNIX/Linux. You will:

- ✅ Create users and assign them to different groups.
- ✅ Configure file permissions for access control.
- ✅ Set up **sudo privileges** for administrative users.
- ✅ Manage restricted and privileged access levels.

Step 1: Create Users with Different Access Levels

1. Create Standard Users

Standard users **have limited access** and cannot execute administrative tasks.
To create users:

```
sudo useradd -m -s /bin/bash user1
```

```
sudo useradd -m -s /bin/bash user2
```

- -m ⟶ Creates a home directory.
- -s /bin/bash ⟶ Assigns the default shell.

Set passwords for the users:

```
sudo passwd user1
```

sudo passwd user2

2. Create Administrative Users (With Sudo Access)

Admins can execute **root-level** commands.

sudo useradd -m -s /bin/bash adminuser

sudo passwd adminuser

sudo usermod -aG sudo adminuser

- -aG sudo ⟶ Adds adminuser to the **sudo group** for administrative privileges.

3. Verify User Access Levels

Check user groups:

groups user1

groups adminuser

Step 2: Managing User Groups and Access Control

1. Create Custom Groups

Create groups for specific access control:

sudo groupadd developers

sudo groupadd hr

2. Assign Users to Groups

Add user1 to **developers** and user2 to **hr**:

sudo usermod -aG developers user1

sudo usermod -aG hr user2

Verify group assignments:

groups user1

groups user2

Step 3: Set File and Directory Permissions

1. Restrict Access to Sensitive Files

Make a directory accessible **only** to developers:

sudo mkdir /project

sudo chown :developers /project

sudo chmod 770 /project

- chown :developers /project → Assigns group ownership.
- chmod 770 /project → Only **developers** can access it.

2. Protect System Files from Standard Users

sudo chmod 640 /etc/shadow

- This prevents non-root users from reading **password hashes**.

3. Test Access Restrictions

Log in as user1 and attempt to access /project:

cd /project

If user1 is in developers, access is granted. Otherwise, permission is denied.

Step 4: Configure Sudo Privileges for Specific Users

1. Grant Limited Sudo Access

Edit sudoers file using visudo:

sudo visudo

Add this line to allow adminuser to restart the web server but prevent other admin commands:

adminuser ALL=(ALL) NOPASSWD: /bin/systemctl restart apache2

- NOPASSWD: → Allows execution without password entry.
- systemctl restart apache2 → Limits commands to web server restarts.

2. Test Sudo Access

Switch to adminuser and try:

sudo systemctl restart apache2

If configured correctly, the command will execute without a password prompt.

Step 5: Managing User Accounts

1. Lock an Inactive User Account

To temporarily disable user2:

sudo usermod -L user2

To unlock the account:

sudo usermod -U user2

2. Delete a User and Their Home Directory

If an employee leaves the company:

sudo userdel -r user1

- -r ⟶ Deletes user's home directory.

Step 6: Verify and Monitor User Access

1. Check Active Users

who

Lists logged-in users.

2. Check Login Attempts

sudo cat /var/log/auth.log | grep "user1"

Displays login activity for user1.

3. Monitor User Activity

sudo lastlog

Shows last login times for all users.

Conclusion

✅ Created and managed users with **different access levels**.
✅ Configured **file permissions and group access**.
✅ Assigned **sudo privileges** for administrative tasks.
✅ Implemented **account security measures**.

Assignment Solution: Configuring and Managing System Logs for Error Tracking in UNIX/Linux

Objective

This assignment provides a step-by-step guide to configuring and managing system logs for **error tracking, monitoring system health, and troubleshooting issues** in UNIX/Linux. You will:

✅ Configure **log rotation** to manage system logs.
✅ Use **system logging tools** (rsyslog, journalctl) for error tracking.
✅ Implement **custom logging** for applications.
✅ Automate **log monitoring and alerts**.

Step 1: Understanding System Logs

1. Identify Key Log Files

System logs are stored in /var/log/. Some important logs include:

Log File	Purpose
/var/log/syslog	General system events (Debian-based)
/var/log/messages	General system events (RHEL-based)
/var/log/auth.log	User authentication logs
/var/log/secure	Security logs
/var/log/dmesg	Kernel messages at boot
/var/log/cron	Cron job execution logs
/var/log/apache2/error.log	Apache web server errors

2. View System Logs

Check system logs for errors:

sudo cat /var/log/syslog | grep "error"

Monitor logs in real time:

sudo tail -f /var/log/syslog

Exercise

1. Identify the last 10 login attempts using:
2. sudo tail -10 /var/log/auth.log
3. Search for failed SSH login attempts:
4. sudo grep "Failed password" /var/log/auth.log

STEP 2: CONFIGURING LOGGING WITH RSYSLOG

1. Verify and Enable rsyslog

Most modern Linux distributions use rsyslog for logging. Check if rsyslog is running:

sudo systemctl status rsyslog

If not running, start it:

sudo systemctl start rsyslog

sudo systemctl enable rsyslog

2. Configure Custom Log Rules

Edit the configuration file:

sudo nano /etc/rsyslog.conf

Add a rule to log kernel errors to a custom file:

kern.* /var/log/kernel_errors.log

Restart rsyslog to apply changes:

sudo systemctl restart rsyslog

3. Test Custom Logging

Send a test message to the log file:

logger -p kern.err "Test kernel error log"

Verify the log entry:

sudo cat /var/log/kernel_errors.log

Exercise

1. Configure rsyslog to log authentication failures to /var/log/auth_failures.log.
2. Test the logging rule using logger.

STEP 3: LOG ROTATION FOR EFFICIENT LOG MANAGEMENT

1. Configure Log Rotation Using Logrotate

Logrotate prevents logs from **consuming excessive disk space** by rotating and compressing logs.
Check existing log rotation settings:

cat /etc/logrotate.conf

2. Configure Log Rotation for a Custom Log File

Create a new configuration file:

sudo nano /etc/logrotate.d/custom_logs

Add the following configuration:

/var/log/kernel_errors.log {

```
    weekly
    rotate 4
    compress
    missingok
    notifempty
}
```

- weekly → Rotates logs weekly.
- rotate 4 → Keeps 4 backups.
- compress → Compresses old logs.

3. Test Log Rotation

Manually run log rotation:

```
sudo logrotate -f /etc/logrotate.d/custom_logs
```

Exercise

1. Set up log rotation for /var/log/auth_failures.log to run daily.
2. Verify the rotated log files in /var/log/.

STEP 4: MONITORING LOGS WITH JOURNALCTL

1. View Recent System Logs

```
journalctl -xe
```

2. Filter Logs by Priority

Show only error logs:

```
journalctl -p err
```

3. Display Logs for a Specific Service

Example: View SSH logs

journalctl -u sshd --since "1 hour ago"

Exercise

1. Find system errors that occurred in the last 30 minutes.
2. List log entries related to nginx service.

Step 5: Automating Log Monitoring and Alerts

1. Set Up Logwatch for Log Analysis

Install logwatch (if not installed):

sudo apt install logwatch # Debian-based

sudo yum install logwatch # RHEL-based

Generate a log report:

sudo logwatch --detail high --mailto root --range yesterday

2. Configure Email Alerts for Log Events

Use logwatch to send email alerts for errors: Edit /etc/logwatch/conf/logwatch.conf:

MailTo = admin@example.com

Range = yesterday

Detail = High

3. Set Up a Cron Job for Daily Log Reports

Open crontab:

crontab -e

Add the following entry to send reports daily at midnight:

0 0 * * * /usr/sbin/logwatch --output mail

Exercise

1. Configure logwatch to send error reports to your email.
2. Set up a cron job to automate log analysis.

Step 6: Securing and Managing Logs

1. Protect Log Files from Unauthorized Access

Ensure logs are only accessible to administrators:

sudo chmod 640 /var/log/auth.log

2. Restrict Remote Log Access

To disable remote logging, edit /etc/rsyslog.conf:

Remove or comment the following line

$ModLoad imudp

$UDPServerRun 514

Restart the logging service:

sudo systemctl restart rsyslog

Exercise

1. Modify permissions for /var/log/syslog to restrict access.
2. Disable remote logging in rsyslog and restart the service.

Step 7: Troubleshooting Log Issues

1. Logs Not Updating? Check Disk Space

If logs are missing or not updating, check available disk space:

df -h

If /var is full, free space by clearing old logs:

sudo rm -rf /var/log/*.gz

2. rsyslog Not Running? Restart the Service

Check the status:

sudo systemctl status rsyslog

Restart it:

sudo systemctl restart rsyslog

3. Logs Not Rotating? Debug logrotate

Manually force rotation and check errors:

sudo logrotate -d /etc/logrotate.conf

Exercise

1. Check system logs for errors related to rsyslog.
2. Identify the last failed login attempt and its timestamp.

CONCLUSION

✅ Configured system logs for **error tracking**.
✅ Implemented **log rotation** using logrotate.
✅ Automated **log analysis and email alerts** using logwatch.
✅ Secured logs from **unauthorized access**.