

Year: 3 rd	Semester: 6th	Academic Session: 2024-25
	Course Name: Linux Administration	

UNIT 1

Development of LINUX:

The development of Linux is a fascinating journey that began as a personal project and evolved into a cornerstone of modern computing. Here's an overview of its development:

Origins and Early Development

- 1991: Linus Torvalds, a Finnish computer science student, started the Linux project as a hobby. He was dissatisfied with the licensing of MINIX, a UNIX-like system used for education.
- Initial Release: He announced his work on the comp.os.minix newsgroup, sharing a basic kernel version 0.01, which was not very functional but sparked interest.

Linux Kernel

- Open Source: Licensed under the GNU General Public License (GPL), allowing anyone to view, modify, and distribute the code.

- Modular Design: Linux is a monolithic kernel, meaning it runs all OS services in kernel space, but it's modular, so features (e.g., file systems, drivers) can be loaded as modules.
-

GNU/Linux

- The term "GNU/Linux" is used by some (notably the Free Software Foundation) to acknowledge the GNU tools and libraries that work with the Linux kernel to form a complete operating system.
-

Growth and Adoption

- Mid to Late 1990s: Gained traction in academia and with hobbyists. Commercial distributions like Red Hat, Debian, and Slackware appeared.
 - 2000s-Present: Rapid adoption in data centers, cloud computing, and embedded systems. Android (based on Linux) powers billions of mobile devices.
-

Modern Ecosystem

- Distributions: Ranging from beginner-friendly (e.g., Ubuntu, Linux Mint) to advanced/server-grade (e.g., Arch, CentOS, RHEL).
 - Development: Still led by Linus Torvalds, with thousands of contributors worldwide.
 - Applications: Used in servers, supercomputers, IoT, smartphones, desktops, and even space missions.
-

Key Features That Fueled Growth

- Stability and performance
 - Security through open-source transparency
 - Flexibility and modularity
 - Support for a wide range of hardware and architectures
 - Strong community and enterprise support
-

Linux Distribution:

A Linux distribution (or distro) is a complete operating system built around the Linux kernel, combined with software packages, tools, and utilities to provide a functional system. Each distribution is tailored for different use cases, such as desktops, servers, embedded systems, or specialized environments.

◆ Key Components of a Linux Distribution

- Linux Kernel: The core of the OS, managing hardware and system resources.
- GNU Tools: Essential tools and libraries (bash, coreutils, etc.).
- Package Manager: Tools for installing, updating, and managing software (e.g., APT, YUM, Pacman).
- Init System: Manages system boot and services (e.g., systemd, SysVinit).

- Desktop Environment (*optional*): For graphical interface (e.g., GNOME, KDE, XFCE).

Popular Linux Distributions (by Category)

General-Purpose

Distribution Description

Ubuntu	User-friendly, widely used, Debian-based. Ideal for desktops and servers.
Debian	Known for stability, the base for many distros including Ubuntu.
Fedora	Cutting-edge features, sponsored by Red Hat.
Arch Linux	Rolling release, minimal, for advanced users who prefer manual configuration.
openSUSE	Versatile, strong system management tools like YaST.

Enterprise-Grade

Distribution	Description
Red Hat Enterprise Linux (RHEL)	Commercial support, widely used in corporate environments.
CentOS Stream	Upstream for RHEL; ideal for devs targeting RHEL.
SUSE Linux Enterprise Server (SLES)	Enterprise-ready, used in SAP environments.

Lightweight/Embedded

Distribution	Description
--------------	-------------

Alpine Linux	Small, secure, often used in containers.
--------------	--

Puppy Linux	Very lightweight, runs on older hardware.
-------------	---

Raspberry Pi OS	Debian-based distro optimized for Raspberry Pi.
-----------------	---

Security & Privacy Focused

Distribution	Description
--------------	-------------

Kali Linux	Penetration testing and ethical hacking.
------------	--

Tails	Privacy-focused, runs from USB with anonymity tools.
-------	--

Qubes OS	Security through isolation using virtualization.
----------	--

Package Managers by Distribution

- APT: Debian, Ubuntu, Linux Mint
- YUM/DNF: RHEL, CentOS, Fedora
- Pacman: Arch Linux, Manjaro
- Zypper: openSUSE

Structure of Linux Operating System:

The structure of the Linux Operating System is layered and modular, designed for flexibility, stability, and performance. Here's a breakdown of its main components:

1. Hardware Layer

- Includes all physical devices: CPU, memory, disk drives, I/O devices, etc.
 - The Linux kernel directly interacts with this layer to manage hardware operations.
-

2. Kernel Layer

- The core of the OS. It acts as a bridge between hardware and software.
 - Responsibilities:
 - Process Management: Schedules and manages running programs.
 - Memory Management: Allocates RAM to processes.
 - Device Drivers: Interfaces with hardware devices.
 - System Calls and Services: Provides system functions for user-space programs.
 - File System Management: Manages storage and file access.
-

3. System Libraries

- Special functions and tools used by system programs and applications to interact with the kernel (e.g., GNU C Library glibc).
 - Abstract kernel functions so users don't interact with the kernel directly.
-

4. System Utilities

- Basic command-line tools and utilities that perform OS-level tasks like file manipulation, process control, networking, etc.

- Examples: cp, mv, ps, top, netstat, etc.
-

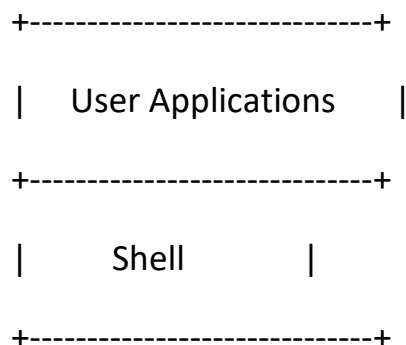
5. User Space (Shell & Applications)

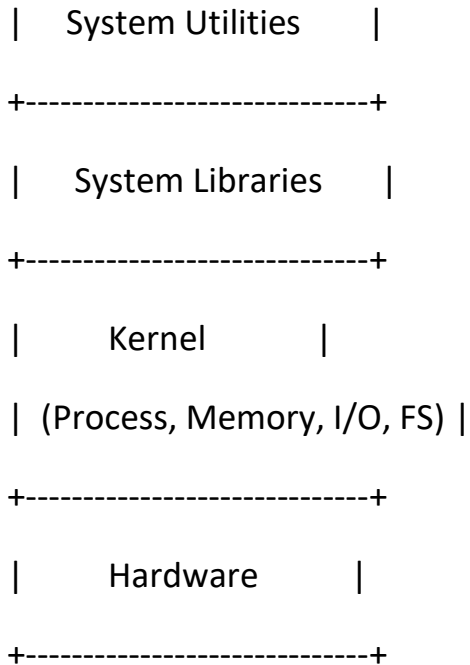
- Shell: The command-line interpreter users interact with (e.g., Bash, Zsh, Fish).
 - User Applications: Everything from text editors and media players to browsers and custom programs.
-

(Optional) 6. Graphical User Interface (GUI)

- Built on top of user space tools.
 - Consists of:
 - X Server or Wayland: Handles display.
 - Window Manager: Manages windows (e.g., i3, Mutter).
 - Desktop Environment: Provides full UI (e.g., GNOME, KDE, XFCE).
-

Diagram Summary





Logging In & General Orientation:

Logging In & General Orientation in Linux

When a user accesses a Linux system—locally or remotely (via SSH)—they are greeted with a login prompt or graphical login screen. After login, they enter the user environment, typically through a shell (e.g., Bash). Here's a general guide:

1. Logging In

Terminal Login (CLI)

- Prompt Example:
- login: user1
- Password: *****

SSH Login (Remote)

- Command:
- `ssh user1@192.168.1.10`
- Requires valid credentials or key pair (for secure login).

GUI Login

- Available in desktop environments (e.g., GNOME, KDE).
 - Enter username and password via graphical display manager (GDM, LightDM, etc.).
-

2. Post-Login Orientation

Working Directory

- On login, you start in your home directory:
- `echo $HOME`
- `pwd`

Common Shell Prompt

- Example:
 - `user1@hostname:~$`
 - Structure:
 - `user1`: Current username
 - `hostname`: System name
 - `~`: Current directory (tilde means home)
-

3. Essential Commands to Get Oriented

Command Description

whoami	Shows your username
pwd	Prints the current directory
ls	Lists files in current directory
cd	Changes directory
hostname	Displays system hostname
uptime	Shows how long the system has been up
top / htop	Shows active processes
df -h	Disk usage summary
free -h	Memory usage summary
id	Shows UID, GID, and groups of the user

4. Filesystem Hierarchy Quick Overview

Directory Purpose

/ Root of the filesystem

Directory Purpose

/home	User home directories
/etc	Configuration files
/var	Variable data (logs, mail, etc.)
/usr	User binaries and libraries
/bin	Essential user commands
/sbin	System commands (admin use)
/dev	Device files
/proc	Kernel & process info (virtual FS)

The X Window System:

The X Window System (often just called X11 or X) is the traditional graphical windowing system used by many Unix and Linux systems. It provides the basic framework for building Graphical User Interfaces (GUIs) and managing graphical display output.

What Is the X Window System?

- A client-server system that separates the display (server) from the applications (clients).
- Allows applications to draw windows and receive input (mouse/keyboard) in a network-transparent way.
- It's not a desktop environment, but the low-level layer upon which desktops like GNOME, KDE, and XFCE are built.

X Window System Architecture

1. X Server

- Handles:
 - Display output (monitor)
 - Keyboard/mouse input
 - Window drawing
- Runs on the local machine or remotely.
- Examples: Xorg, XFree86

2. X Clients

- GUI applications that request drawing services from the X server.
- Communicate with the server over a protocol.

3. Window Manager

- Controls the appearance and behavior of windows (move, resize, title bar).
- Examples:
 - Lightweight: Openbox, Fluxbox
 - Tiling: i3, Xmonad
 - Compositing: Compiz, Mutter

4. Display Manager (*Optional*)

- Login screen and session manager for graphical logins.
- Examples: GDM (GNOME), LightDM, SDDM (KDE)

Common X Window Tools

- startx: Starts an X session from the terminal.
 - xinit: Manually initialize X without a display manager.
 - xrandr: Manage screen resolutions and displays.
 - xset: Configure user preferences (keyboard, screen saver).
 - xeyes, xclock: Example X client apps.
-

Limitations of X

- Complexity and outdated architecture.
 - Lack of modern features like proper GPU acceleration, security isolation.
 - Performance issues in remote/modern setups.
-

Successor: Wayland

- A modern replacement for X, designed to be simpler and more secure.
 - Now used by default in some distributions (e.g., Fedora, Ubuntu GNOME).
 - Compositors like Mutter, KWin, and Sway support Wayland.
-

KDE:

KDE (K Desktop Environment)

KDE is a powerful, modern, and highly customizable desktop environment for Linux and Unix-like systems. It provides a complete graphical interface built on top of the X Window System (or Wayland), offering a user-friendly experience with a focus on aesthetics and functionality.

Key Features of KDE

Feature	Description
Plasma Desktop	KDE's flagship desktop interface, sleek and modern.
Highly Customizable	Panels, widgets, themes, shortcuts—all can be modified.
Qt Framework	Uses Qt libraries, known for performance and beautiful UIs.
KRunner	A versatile command launcher and search tool.
Dolphin	KDE's advanced file manager with split views, previews, etc.
Discover	KDE's GUI software center for installing/updating apps.
System Settings	Centralized control panel with vast customization options.

KDE Technologies

- **Plasma:** The graphical workspace environment.
- **KWin:** KDE's window manager and Wayland compositor.
- **Frameworks 5:** Reusable libraries built on Qt.
- **Applications:** KDE ships with a suite of tools like:
 - **Kate** (text editor)
 - **Konsole** (terminal emulator)
 - **Gwenview** (image viewer)

- **Okular (document viewer)**

KDE-Based Linux Distros

Distribution	Description
Kubuntu	Ubuntu + KDE
Fedora KDE Spin	Fedora with KDE desktop
openSUSE	Excellent KDE support by default
Manjaro KDE	Arch-based distro with polished KDE
KaOS	KDE-focused independent distro

KDE vs. GNOME

Feature	KDE Plasma	GNOME
Toolkit	Qt	GTK
Customization	Extensive	Minimal (by design)
Resource Use	Lighter than GNOME (recent versions)	Heavier
Appearance	Windows-like, more visual options	Clean, modern, minimal

GNOME:

GNOME (GNU Network Object Model Environment)

GNOME is one of the most widely used desktop environments for Linux and Unix-like operating systems. Known for its simplicity, clean design, and usability, it aims to offer a minimalist yet powerful user experience.

Key Features of GNOME

Feature	Description
GNOME Shell	The main interface for launching applications and managing windows.
Activities Overview	Unified view for window switching, application launching, and workspace management.
GTK Toolkit	Built with GTK (GIMP Toolkit), providing a clean, native look.
Extensions Support	Customizable via GNOME Shell Extensions (e.g., Dash to Dock, system monitors).
Nautilus	The default file manager. Simple and integrated.

GNOME Settings Centralized settings with a clean UI.

Wayland by Default	In many distros, GNOME uses Wayland for better security and performance.
---------------------------	---

GNOME Core Applications

- Terminal: GNOME's terminal emulator.**
- Files (Nautilus): File management tool.**
- Text Editor (gedit): Lightweight and efficient text editor.**
- GNOME Software: GUI tool for managing software and updates.**

- **Evince: Document viewer.**
 - **GNOME Calendar, Clocks, Weather, and more.**
-

GNOME-Based Linux Distro

Distribution	Description
Ubuntu (default)	Comes with GNOME since version 17.10.
Fedora Workstation	Premier GNOME experience, upstream-focused.
Debian GNOME	Stable and pure GNOME implementation.
Pop!_OS	GNOME-based but heavily customized.

GNOME vs KDE (Quick Comparison)

Feature	GNOME	KDE Plasma
Toolkit	GTK	Qt
Design Philosophy	Minimalist, clean UI	Highly customizable
Performance	Heavier by default	Lighter (in Plasma 5+)
Configuration	GUI-based, fewer options	Extensive via settings

Extensions and Tweaks

- **Use GNOME Tweaks to:**
 - **Adjust themes, fonts, and UI behaviors.**
 - **Enable extensions.**

- GNOME Extensions Website: extensions.gnome.org

Navigating The File System:

Navigating the file system is a fundamental skill in Linux. Everything in Linux is represented as a file or directory (even devices and processes).

Basic Directory Structure

Directory Purpose

/	Root directory (top of the hierarchy)
/home	Contains personal directories for each user
/etc	System-wide configuration files
/bin	Essential binaries (e.g., ls, cp)
/usr	User applications and files
/var	Variable data (logs, mail, cache)
/tmp	Temporary files
/dev	Device files
/proc	Virtual filesystem for system and process info

Essential Navigation Commands

Command Description

pwd	Print current directory path
ls	List files/directories
cd	Change directory
cd ~	Go to your home directory
cd ..	Move one directory up
cd /	Go to the root directory
ls -l	Long listing format
ls -a	Show hidden files (starting with .)

Examples

```
cd /etc      # Go to system config directory
ls -l /etc   # List contents with permissions
cd ~/Downloads # Navigate to Downloads in your home dir
cd ../../    # Go up two levels
```

File & Directory Handling (Quick Look)

Task	Command
------	---------

Create directory	mkdir myfolder
------------------	-----------------------

Remove directory	rmdir myfolder (empty) or rm -r myfolder
------------------	---

Task	Command
Copy files	<code>cp file1 /path/to/destination/</code>
Move/Rename	<code>mv file1 file2</code>
Delete file	<code>rm file1</code>
View file content	<code>cat, less, or more</code>

Managing Files:

File management in Linux is done primarily through the command line using a variety of powerful tools. Here's a comprehensive guide:

Creating Files

Command	Description
<code>touch filename</code>	Creates an empty file
<code>nano filename</code>	Creates/edits a file with nano editor
<code>vi filename</code>	Opens file in vi or vim
<code>cat > filename</code>	Create a file and type content (Ctrl+D to save)

Creating & Removing Directories

Command	Description
<code>mkdir dirname</code>	Create a new directory

Command	Description
<code>mkdir -p path/dir</code>	Create nested directories
<code>rmdir dirname</code>	Remove an empty directory
<code>rm -r dirname</code>	Remove a directory and its contents recursively

Viewing File Content

Command	Description
<code>cat file</code>	Display content all at once
<code>less file</code>	Scrollable view (use q to quit)
<code>more file</code>	View file page by page
<code>head -n 10 file</code>	View first 10 lines
<code>tail -n 10 file</code>	View last 10 lines
<code>tail -f file</code>	Monitor file in real-time (useful for logs)

Copying, Moving, and Renaming

Command	Description
<code>cp file1 file2</code>	Copy file1 to file2
<code>cp -r dir1 dir2</code>	Copy directory recursively
<code>mv file newname</code>	Rename or move a file
<code>mv file /path/</code>	Move file to a new location

Command	Description
<code>mv file1 dir1/</code>	Move file1 into dir1

Deleting Files and Directories

Command	Description
<code>rm file</code>	Delete a file
<code>rm -i file</code>	Confirm before delete
<code>rm -r folder</code>	Delete folder and contents
<code>rm -rf folder</code>	Force delete (use with caution!)

Searching Files

Command	Description
<code>find /path -name "file"</code>	Find file by name
<code>locate filename</code>	Quickly find files (uses a database)
<code>grep "text" file</code>	Search for text in a file
<code>grep -r "text" dir</code>	Recursive search in directory

What is sudo?

- `sudo` stands for "superuser do".

- It allows a permitted user to execute a command as the superuser (root) or another user, as specified by the sudoers file.
-

Common File Management with sudo

Task	Command
Edit a system config file	<code>sudo nano /etc/fstab</code>
Copy a file to a system directory	<code>sudo cp file.conf /etc/</code>
Change ownership of a file	<code>sudo chown user:group file</code>
Delete protected file	<code>sudo rm /etc/protected_file</code>
Create directory in /usr	<code>sudo mkdir /usr/local/myscripts</code>

Security with sudo

- Only users in the sudo group (or wheel group on some systems) can use sudo.
 - The system logs each sudo command to `/var/log/auth.log` (Debian-based) or `/var/log/secure` (RHEL-based).
 - Example:
 - `sudo usermod -aG sudo yourusername`
-

Elevating to Root Shell Temporarily

`sudo -i` # Opens a root shell (interactive)

`sudo su` # Another method (less preferred)

Be cautious when using elevated privileges—you can easily damage your system if you delete or modify **critical files**.

File Permission & Access:

Linux uses a permission model to control who can read, write, or execute files and directories. This ensures system security and multi-user management.

Linux Permission Structure

Run `ls -l` to view permissions:

`-rwxr-xr-- 1 user group 1234 Mar 23 10:00 example.sh`

Breakdown:

- `-` → File type (`-` for file, `d` for directory)
- `rwx` → Owner permissions
- `r-x` → Group permissions
- `r--` → Others (world) permissions

Symbol Meaning

r Read

w Write

x Execute

- No permission

Ownership

Every file has:

- An owner (usually the creator)
- A group (users with shared access)

Check with:

`ls -l file`

Change ownership:

`sudo chown newuser:newgroup file`

Changing Permissions with chmod

Symbolic Mode:

`chmod u+x script.sh` # Add execute for user (owner)

`chmod g-w file.txt` # Remove write for group

`chmod o+r file.txt` # Add read for others

Numeric (Octal) Mode:

Value Permission

7 rwx

6 rw-

5 r-x

4 r--

0 ---

`chmod 755 script.sh` # rwxr-xr-x

chmod 644 file.txt # rw-r--r--

chmod 700 private.sh # Only owner has access

Changing Group Ownership with chgrp

chgrp staff report.pdf

Special Permission Types

Type	Symbol	Purpose
------	--------	---------

Setuid	s in user exec	Executes as file owner
--------	----------------	------------------------

Setgid	s in group exec	Executes as group
--------	-----------------	-------------------

Sticky	t on dir	Only owner can delete their files (common in /tmp)
--------	----------	--

Example:

chmod +t /sharedfolder # Sticky bit

chmod g+s scripts/ # Setgid

chmod u+s script.sh # Setuid

UNIT 2

Installation Of LINUX On VMWare:

What You Need

1. **VMware Workstation Player** (Windows/Linux) or **VMware Fusion** (Mac)
 2. A **Linux ISO file** (e.g., Ubuntu, Fedora, Debian)
-

Step-by-Step Installation Guide

1. Install VMware Workstation Player

- Download from: [VMware Workstation Player](#)
- Install as per your OS instructions.

2. Download a Linux ISO

- Ubuntu: ubuntu.com/download
- Fedora: getfedora.org
- Debian: debian.org/distrib

3. Create a New Virtual Machine

- Open VMware → Click “**Create a New Virtual Machine**”
- Choose “**Installer disc image file (iso)**” and browse to the Linux ISO.
- Click **Next**.

4. Configure VM Settings

- **Name:** e.g., "Ubuntu_Training"
- **Location:** Choose storage path
- **Disk size:** Recommend at least **20 GB**
- Choose **"Store virtual disk as a single file"** for performance.

5. Customize Hardware

- Click **"Customize Hardware"**:
 - **RAM:** Allocate at least **2–4 GB**
 - **Processors:** Set to 2 (if available)
 - **Network Adapter:** Use **NAT** or **Bridged** based on internet access preference

6. Start the VM

- Click **"Finish"**, then **"Power on"** the VM.
- The Linux installer will boot from the ISO.

7. Install Linux

- Follow the installation prompts:
 - Choose language, keyboard
 - Configure user, password
 - Select **Erase disk and install** (only affects the virtual disk)

8. Post-Installation

- After installation, **remove ISO** from virtual CD drive.

- Reboot the VM into your new Linux system.
-

Optional Enhancements

- **Install VMware Tools** (for shared clipboard, auto-resize display)
 - **Snapshot** the clean state of your system
 - **Enable shared folders** between host and VM
-

1. Install VMware Tools (Open VM Tools)

VMware Tools enhances VM performance and enables features like shared clipboard, drag-and-drop, and screen resizing.

For Ubuntu/Debian-based Distro:

Open a terminal and run:

```
sudo apt update
```

```
sudo apt install open-vm-tools open-vm-tools-desktop -y
```

```
sudo reboot
```

For Red Hat/CentOS/Fedora:

```
sudo dnf install open-vm-tools open-vm-tools-desktop -y
```

```
sudo systemctl enable --now vmtoolsd
```

```
sudo reboot
```

After reboot, features like **auto-resize screen** and **shared clipboard** should work.

2. Take a Snapshot of Clean State

In VMware Workstation Player:

1. Power off the VM.
2. Go to **VM > Snapshot > Take Snapshot**.
3. Name it (e.g., “Fresh Install”) and click **OK**.

Tip: Use snapshots before making major changes or updates.

3. Enable Shared Folders

Step 1: Configure in VMware

1. With the VM powered off, go to **VM Settings > Options > Shared Folders**.
2. Set it to **Always enabled** or **Enabled until next power off**.
3. Click **Add...**, select a host folder, and give it a name (e.g., **SharedDocs**).

Step 2: Access in Linux Guest

Ubuntu/Debian:

Install support if not already:

```
sudo apt install open-vm-tools open-vm-tools-desktop -y
```

```
sudo reboot
```

Check mount location:

```
ls /mnt/hgfs
```

You should see your shared folder ([SharedDocs](#)). You can copy or work with files from there.

4. Networking Setup: NAT vs. Bridged

- **NAT** (default): VM uses the host's IP address to access the internet. Safe and simple.
- **Bridged**: VM gets its own IP on your network (acts like another PC).

To Change:

1. Shut down the VM.
2. Go to **VM Settings > Network Adapter**.
3. Choose either **NAT** or **Bridged** mode.

Power on the VM and verify IP:

```
ip a
```

Configuring Disk Devices in Linux

Disk device configuration in Linux involves identifying, partitioning, formatting, mounting, and managing storage devices. This is a critical skill for system administrators and a common topic in exams and practical scenarios.

1. Identifying Disk Devices

Commands to View Disks:

Command	Purpose
---------	---------

`lsblk` Lists block devices (recommended view)

`fdisk -l` Lists all disks and partitions (detailed)

`blkid` Lists device UUIDs and filesystem types

`df -h` Shows mounted filesystems and usage

`ls` Lists all SCSI/SATA disk devices
`/dev/sd*`

2. Partitioning Disks

Common Tools:

- `fdisk` — for MBR partitioning (older)
- `parted` — for both MBR and GPT
- `gdisk` — for GPT
- `cfdisk` — TUI-based partitioning

Example with `fdisk`:

`sudo fdisk /dev/sdb`

- `n` = new partition

- **p** = print partition table
- **w** = write changes
- **d** = delete partition
- **q** = quit without saving

Use **g** to create GPT and **o** for MBR if needed.

3. Formatting Partitions

After creating partitions, format them with a filesystem:

Filesystem	Command
ext4	<code>sudo mkfs.ext4 /dev/sdb1</code>
xf	<code>sudo mkfs.xfs /dev/sdb1</code>
vf	<code>sudo mkfs.vfat /dev/sdb1</code>
swap	<code>sudo mkswap /dev/sdb2</code>

4. Mounting and Accessing

Mount manually:

```
sudo mount /dev/sdb1 /mnt/mydisk
```

Unmount:

```
sudo umount /mnt/mydisk
```

Auto-mount using */etc/fstab*:

Get UUID:

```
sudo blkid /dev/sdb1
```

Edit fstab:

```
sudo nano /etc/fstab
```

Add line:

```
UUID=xxxx-xxxx /mnt/mydisk ext4 defaults 0 2
```

5. Swap Configuration

Create and enable swap partition or file:

```
sudo mkswap /dev/sdb2
```

```
sudo swapon /dev/sdb2
```

Check swap:

```
swapon --show
```

6. Persistent Mount Permissions

To ensure correct ownership and access:

```
sudo chown -R user:user /mnt/mydisk
```

```
sudo chmod 755 /mnt/mydisk
```

Creating and Managing File Systems in Linux

Creating and managing file systems is a core task in Linux system administration. It includes **formatting partitions, mounting, checking, and maintaining** storage volumes.

1. Preparing a Disk

Before creating a file system:

- Identify the target device: `lsblk`, `fdisk -l`
 - Partition it (if needed) using tools like `fdisk`, `parted`, or `gparted`.
-

2. Creating a File System

◆ Common File System Types and Commands

File System	Command
<code>ext4</code>	<code>sudo mkfs.ext4 /dev/sdb1</code>
<code>xfs</code>	<code>sudo mkfs.xfs /dev/sdb1</code>

```
vfat (FAT32)  sudo mkfs.vfat /dev/sdb1
```

```
ntfs          sudo mkfs.ntfs /dev/sdb1 (install ntfs-  
3g first)
```

```
btrfs         sudo mkfs.btrfs /dev/sdb1
```

Example:

```
sudo mkfs.ext4 -L MyVolume /dev/sdb1
```

3. Mounting the File System

Mount Temporarily:

```
sudo mkdir /mnt/mydata
```

```
sudo mount /dev/sdb1 /mnt/mydata
```

View mounted systems:

```
mount | grep sdb1
```

```
df -h
```

4. Persistent Mount with **/etc/fstab**

Steps:

Get UUID:

```
sudo blkid /dev/sdb1
```

Edit fstab:

```
sudo nano /etc/fstab
```

Add line:

```
UUID=xxxx-xxxx /mnt/mydata ext4 defaults 0 2
```

Test:

```
sudo mount -a
```

5. Checking and Repairing File Systems

Command	Purpose
<code>fsck</code>	Check and fix file system issues
<code>e2fsck</code>	Check ext2/ext3/ext4 filesystems
<code>xfstool</code>	Repair XFS volumes (only offline)

Example:

```
sudo fsck /dev/sdb1
```

Make sure the partition is **unmounted** before running `fsck`.

6. Resizing File Systems

Resize ext4:

```
sudo resize2fs /dev/sdb1
```

- (Make sure to resize the partition first using *parted* or *gparted*.)

Resize XFS:

```
sudo xfs_growfs /mnt/mydata
```

-

7. Setting Permissions on Mount Point

```
sudo chown -R $USER:$USER /mnt/mydata
```

```
sudo chmod 755 /mnt/mydata
```

File System Backup in Linux

Backing up your file system is essential for **data protection**, **disaster recovery**, and **system migration**. Linux offers powerful built-in tools for creating full, incremental, and compressed backups.

1. Backup Types

Type	Description
Full Backup	Complete copy of all selected files/directories

Incremental Backup Only changed files since the last backup

Differential Backup Changes since the last full backup

2. Tools for File System Backup

◆ **tar** – Archive Tool (Most Common)

Full backup of /home

```
sudo tar -cvpzf /backup/home_backup_$(date +%F).tar.gz /home
```

Options:

- **c** = create
- **v** = verbose
- **p** = preserve permissions
- **z** = compress (gzip)
- **f** = filename

To extract:

```
sudo tar -xvpzf home_backup_2025-03-26.tar.gz -C /restore/location
```

◆ **rsync** – Sync & Incremental Backup

```
sudo rsync -aAXv /home/ /backup/home_rsync/
```

Options:

- **-a** = archive (preserves everything)
- **-A** = preserve ACLs
- **-X** = preserve extended attributes
- **-v** = verbose

Can be used with **cron** for scheduled backups.

◆ **dd – Disk Image (Block-Level)**

```
sudo dd if=/dev/sda of=/backup/sda_backup.img bs=64K status=progress
```

- Backs up the entire disk (bootloader included).
- Useful for full disk cloning.

To restore:

```
sudo dd if=/backup/sda_backup.img of=/dev/sda bs=64K status=progress
```

◆ **cp – Simple File Copy (not recommended for complex systems)**

```
cp -a /home /backup/
```

3. Secure Your Backup

- Store in separate **disk**, **USB**, or **remote server**.
- Use **gpg** to **encrypt** sensitive archives:

```
gpg -c backup.tar.gz
```

4. Automate with **cron**

Create a script **daily_backup.sh**:

```
#!/bin/bash
```

```
tar -czf /backup/etc_$(date +%F).tar.gz /etc
```

Make it executable:

```
chmod +x daily_backup.sh
```

Add to cron:

```
crontab -e
```

Example (runs daily at 2am):

```
0 2 * * * /home/user/daily_backup.sh
```

Kickstart Installation in Linux (Automated RHEL/CentOS Installation)

Kickstart is a method used to **automate the installation** of Red Hat-based Linux distributions (e.g., RHEL, CentOS, AlmaLinux, Rocky Linux). It uses a **Kickstart configuration file** that answers all the installer's questions, enabling **unattended installations**.

1. What is a Kickstart File?

- A plain text file containing instructions for:
 - Disk partitioning
 - Language/keyboard settings
 - Network config
 - Package selection
 - Post-installation scripts
 - Typically named `ks.cfg`
-

2. Sample Kickstart File (`ks.cfg`)

System language

lang en_US

Keyboard layouts

keyboard us

System timezone

timezone America/New_York --isUtc

Root password

rootpw --iscrypted \$6\$abc...

Use all disk

autopart

Network settings

network --bootproto=dhcp --onboot=on --hostname=kickstart-machine

SELinux

selinux --enforcing

Firewall

firewall --enabled

System authorization

authconfig --enablesshadow --passalgo=sha512

Reboot after install

reboot

```
# Package selection
```

```
%packages
```

```
@core
```

```
vim
```

```
%end
```

```
# Post-install script
```

```
%post
```

```
echo "Kickstart completed" > /root/kickstart_done.log
```

```
%end
```

3. How to Use Kickstart File

Create ISO with Embedded Kickstart:

Generate the `ks.cfg` file.

Place it in the ISO under `/isolinux/` or `/ks/`.

Modify boot menu (`isolinux.cfg`) to include:

```
append initrd=initrd.img inst.ks=cdrom:/ks/ks.cfg
```

Use it on PXE/HTTP:

Host `ks.cfg` via HTTP server:

`http://192.168.1.100/ks/ks.cfg`

-

Boot with:

`linux ks=http://192.168.1.100/ks/ks.cfg`

4. Tools to Generate Kickstart Files

- **Kickstart Configurator** (GUI): `system-config-kickstart`
 - **Anaconda** log files: After a manual install, check `/root/anaconda-ks.cfg` for a working template.
-

Automated Installation in Ubuntu (Equivalent to Kickstart)

Ubuntu uses:

1. Preseed (Legacy)

- Works with older **Ubuntu Server versions (before 20.04)**.
- Similar to Kickstart—answers installer questions via a plain text file.

2. Autoinstall (Modern)

- Used from **Ubuntu 20.04 onward**.
 - YAML-based configuration, integrated with **cloud-init**.
-

1. Autoinstall (Recommended for 20.04+)

◆ Sample **user-data** YAML (Autoinstall file)

```
#cloud-config

autoinstall:

  version: 1

  identity:

    hostname: ubuntu-autoinstall

    username: admin

    password: "$6$abcd..." # Encrypted password (use mkpasswd -m sha-512)

  keyboard:

    layout: us

  locale: en_US

  timezone: UTC

  storage:

    layout:

      name: lvm

  packages:

    - openssh-server

    - vim

  ssh:

    install-server: true

    authorized-keys:

      - ssh-rsa AAAA...user@example
```

user-data:

disable_root: false

late-commands:

- echo "Autoinstall completed" > /target/root/autoinstall.log

2. Using Autoinstall with Ubuntu ISO

Step-by-step:

1. **Download Ubuntu Server ISO** (20.04+).
2. **Create a custom ISO** or set up a **network boot**:
 - o Place **user-data** and **meta-data** in **/nocloud/** directory.

Boot ISO with kernel parameters:

autoinstall ds=nocloud-net;s=http://your-server/nocloud/

- o
3. **Serve files via HTTP** or embed them in the ISO using **xorriso**.

3. Use Cloud-Init for Cloud Environments

- Ubuntu images in cloud platforms (AWS, Azure, OpenStack) use **cloud-init**.
- You can inject **user-data** during instance creation for automatic setup.
-

Linux Bootloader: Explained

The **bootloader** is a crucial component in the Linux boot process. It **loads the Linux kernel** into memory and transfers control to it after system startup. Without it, your operating system won't load.

1. What is a Bootloader?

- A small program executed by the BIOS/UEFI after power-on.
 - Responsible for:
 - Locating the OS kernel.
 - Loading the kernel into memory.
 - Passing necessary parameters to the kernel.
 - Initiating the OS startup.
-

2. Common Linux Bootloaders

Bootloader	Description
GRUB (GRand Unified Bootloader)	Most widely used in modern distros

LILO (Linux Loader)

Legacy; less flexible than GRUB

SYSlinux/ISOLinux

Lightweight, used for booting from FAT/ISO media

systemd-boot

Simpler UEFI bootloader used with systemd

3. GRUB (Most Common)

GRUB Functions:

- Supports multiboot (choose between OSes).
- Can load kernels from **ext2/ext3/ext4**, **Btrfs**, **XFS**, etc.
- Allows editing boot parameters at runtime.

GRUB Configuration:

Main file:

`/etc/default/grub`

To regenerate GRUB config:

`sudo update-grub`

Install/repair GRUB:

`sudo grub-install /dev/sda`

4. Boot Process Summary

1. **BIOS/UEFI** runs POST (Power-On Self-Test).
2. BIOS/UEFI loads **MBR or EFI bootloader** from disk.
3. **Bootloader (e.g., GRUB)** starts and loads the kernel.
4. **Kernel** initializes hardware and mounts root FS.
5. **Init system** (e.g., systemd) takes over and starts services.

5. GRUB Boot Menu Customization

Edit `/etc/default/grub`:

```
GRUB_TIMEOUT=5
```

```
GRUB_DEFAULT=0
```

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
```

Then run:

```
sudo update-grub
```

6. Rescue and Recovery

If GRUB is corrupted:

1. Boot from a Live USB.
2. Mount the system root:

```
sudo mount /dev/sda1 /mnt
```

```
sudo mount --bind /dev /mnt/dev
```

```
sudo mount --bind /proc /mnt/proc
```

```
sudo mount --bind /sys /mnt/sys
```

```
sudo chroot /mnt
```

```
grub-install /dev/sda
```

```
update-grub
```

```
exit
```

```
reboot
```

Linux Kernel Management

The **Linux kernel** is the heart of the operating system—it controls hardware, manages resources, and provides system services. Managing the kernel involves **updating, compiling, installing modules**, and **configuring** boot parameters.

1. Viewing Kernel Version & Info

```
uname -r      # Kernel version
```

```
uname -a      # All kernel info
```

```
cat /proc/version # Detailed kernel info
```

2. Updating the Kernel

Ubuntu/Debian:

```
sudo apt update
```

```
sudo apt install linux-image-generic
```

RHEL/CentOS/Fedora:

```
sudo dnf update kernel
```

Reboot to load the new kernel:

```
sudo reboot
```

3. Installing Specific Kernel Versions

Ubuntu:

```
sudo apt install linux-image-5.x.x-generic
```

RHEL/CentOS:

Enable ELRepo and run:

```
sudo dnf --enablerepo=elrepo-kernel install kernel-ml
```

4. Kernel Modules Management

View Loaded Modules:

```
lsmod
```

Load a Module:

```
sudo modprobe <module_name>
```

Remove a Module:

```
sudo modprobe -r <module_name>
```

View Module Details:

```
modinfo <module_name>
```

5. Kernel Boot Parameters

Edit GRUB file:

```
sudo nano /etc/default/grub
```

Add or modify:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash nomodeset"
```

Update GRUB:

```
sudo update-grub
```

6. Compiling a Custom Kernel (Advanced)

Install required packages:

```
sudo apt install build-essential libncurses-dev bison flex libssl-dev libelf-dev
```

1.

Download kernel source:

```
wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.7.tar.xz
```

```
tar -xvf linux-6.7.tar.xz
```

```
cd linux-6.7
```

2.

Configure:

```
make menuconfig # TUI configuration
```

3.

Build and install:

```
make -j$(nproc)
```

```
sudo make modules_install
```

```
sudo make install
```

4.

Update GRUB and reboot:

```
sudo update-grub
```

```
sudo reboot
```

Managing User Accounts:

Managing user accounts is one of the most basic and essential tasks in Linux.

1. Viewing User Info

Show current user:

```
whoami
```

List all system users:

```
cat /etc/passwd
```

View user ID and group:

```
id
```

2. Creating a New User

```
sudo adduser john
```

- Prompts for password and user details
- Creates home directory `/home/john`
- Assigns default shell (`/bin/bash`)
- Adds to default group with same name

3. Setting or Changing Password

```
sudo passwd john
```

- Prompts for a new password
 - Can be used for any user (if you're root or using `sudo`)
-

4. Modifying a User Account

Task	Command
Change username	<code>sudo usermod -l newname oldname</code>
Change home directory	<code>sudo usermod -d /new/home john</code>
Change shell	<code>sudo usermod -s /bin/zsh john</code>
Add to group	<code>sudo usermod -aG sudo john</code>

5. Deleting a User

`sudo deluser john`

Remove user **with home directory**:

`sudo deluser --remove-home john`

6. Managing Groups

Create a group:

```
sudo groupadd developers
```

Add user to a group:

```
sudo usermod -aG developers john
```

View user's groups:

```
groups john
```

Remove user from group:

```
sudo gpasswd -d john developers
```

7. Check Login & Account Info

File	Description
<code>/etc/passwd</code>	Basic user details
<code>/etc/shadow</code>	Encrypted passwords
<code>/etc/group</code>	Group memberships
<code>/etc/sudoers</code>	Sudo access (edit using <code>visudo</code>)

8. Give Sudo Privileges

Add user to **sudo** group:

```
sudo usermod -aG sudo john
```

Then test:

```
su - john
```

```
sudo ls /
```

Understanding File Listing in Linux:

File listing in Linux is done primarily with the **ls** command. It's a fundamental skill for navigating and managing the file system. Here's a beginner-friendly explanation of how it works and what each part means.

1. Basic **ls** Usage

ls

Lists files and directories in the current working directory.

2. Common **ls** Options

Command	Description
---------	-------------

- `ls -l` Long listing format (detailed info)
 - `ls -a` Show hidden files (those starting with `.`)
 - `ls -lh` Human-readable sizes
 - `ls -lt` Sort by modification time
 - `ls -R` Recursive listing (into subdirectories)
-

3. Sample Output: `ls -l`

`-rwxr-xr-- 1 user group 1234 Mar 26 10:00 example.sh`

Breakdown:

Field	Meaning
<code>-rwxr-xr--</code>	File type and permissions
<code>1</code>	Number of hard links

user	File owner
group	File's group
1234	File size in bytes
Mar 26 10:00	Last modified date & time
example.s h	File name

4. File Types (First Character in Permissions)

Symbol	Type
-	Regular file
d	Directory
l	Symbolic link
c	Character device

b

Block device

5. Understanding Permissions

- **Owner:** `rwX` = read, write, execute
 - **Group:** `r-X` = read, execute
 - **Others:** `r--` = read only
-

1. File Ownership

Every file in Linux has:

- **Owner** (typically the user who created it)
- **Group** (users who share access)

View ownership:

`ls -l filename`

Example output:

```
-rw-r--r-- 1 alice developers 1024 Mar 26 12:00 report.txt
```

- `alice` is the **owner**
- `developers` is the **group**

2. File Permissions Breakdown

Syntax: `-rwxr-xr--`

Section	Entity	Meaning
<code>rwx</code>	Owner	Read, write, execute
<code>r-x</code>	Group	Read, execute
<code>r--</code>	Others	Read only

First character:

- `-` → regular file
- `d` → directory
- `l` → symbolic link

3. Changing Ownership

chown – Change file owner and group

```
sudo chown user:group file
```

Example:

```
sudo chown bob:staff report.txt
```

chgrp – Change only group ownership

```
sudo chgrp developers project/
```

4. Changing Permissions

Symbolic Mode with chmod

Command	Effect
<code>chmod u+x file</code>	Add execute to owner
<code>chmod g-w file</code>	Remove write from group
<code>chmod o+r file</code>	Add read to others

Numeric (Octal) Mode:

Value	Permissions
7	<code>rwX</code>

6 `rw-`

5 `r-x`

4 `r--`

0 `---`

Example:

```
chmod 755 script.sh # rwxr-xr-x
```

```
chmod 644 notes.txt # rw-r--r--
```

5. Permissions on Directories

- `r`: list contents
- `w`: create/delete files
- `x`: access files inside

```
chmod 700 myfolder
```

Grants full access to the owner only.

6. Special Permission Bits

Bit	Symbol	Purpose
Setuid	s	Executes as file owner
Setgid	s	Executes as group
Sticky	t	Only owner can delete (used in /tmp)

`chmod +t /shared/folder`

`chmod u+s script.sh`

`chmod g+s project/`

Managing Software Using RPM in Linux:

RPM (Red Hat Package Manager) is a package management system used primarily in **Red Hat-based distributions** like **RHEL, CentOS, Fedora, AlmaLinux, and Rocky Linux**.

1. What is an RPM Package?

- An **.rpm** file contains a **precompiled binary**, configuration files, and metadata.
- It is used to install, upgrade, remove, and verify software on RPM-based systems.

2. Installing RPM Packages

```
sudo rpm -ivh package-name.rpm
```

Options:

- **-i**: install
- **-v**: verbose
- **-h**: show progress bar

Use **dnf** or **yum** if dependencies need to be resolved automatically.

3. Upgrading an RPM Package

```
sudo rpm -Uvh package-name.rpm
```

- **-U**: upgrade (installs if not already present)

4. Removing an RPM Package

```
sudo rpm -e package-name
```

You must use the **package name**, not the **.rpm** file name.

5. Querying RPM Packages

Command	Purpose
<code>rpm -q package</code>	Check if installed
<code>rpm -qa</code>	List all installed RPM packages
<code>rpm -qi package</code>	Detailed info
<code>rpm -ql package</code>	List installed files
<code>rpm -qf /path/to/file</code>	Find which package a file belongs to

6. Verifying Packages

`rpm -V package-name`

Checks for:

- Modified or missing files
 - Changes in permissions or ownership
-

Connecting to a Network in Linux:

Connecting to a network in Linux can be done via **command-line** or **GUI**, depending on your environment (server or desktop). Here's how to do it for both **wired (Ethernet)** and **wireless (Wi-Fi)** connections.

1. Check Network Interfaces

```
ip link show
```

or

```
ifconfig # Requires net-tools
```

2. Wired (Ethernet) Network

Most wired connections use DHCP and connect automatically.

To bring up/down interface:

```
sudo ip link set eth0 up
```

```
sudo ip link set eth0 down
```

To request IP via DHCP:

```
sudo dhclient eth0
```

3. Connecting to Wi-Fi (CLI)

Step 1: List available Wi-Fi devices

```
nmcli device status
```

Step 2: List Wi-Fi networks

```
nmcli dev wifi list
```

Step 3: Connect to a network

```
nmcli dev wifi connect "SSID_NAME" password "your_password"
```

4. Static IP Configuration

Edit the Netplan file (Ubuntu 18.04+):

```
sudo nano /etc/netplan/01-netcfg.yaml
```

Example:

```
network:
```

```
  version: 2
```

```
  ethernets:
```

```
    eth0:
```

```
      addresses: [192.168.1.100/24]
```

```
      gateway4: 192.168.1.1
```

nameservers:

addresses: [8.8.8.8, 8.8.4.4]

Apply changes:

`sudo netplan apply`

5. Checking Network Status

Command	Purpose
<code>ip a</code> or <code>ip addr</code>	Check IP addresses
<code>ping 8.8.8.8</code>	Check connectivity
<code>nmcli</code>	Manage NetworkManager connections
<code>ss -tuln</code>	List open ports
<code>curl ifconfig.me</code>	Get public IP

6. GUI Method (Ubuntu Desktop)

1. Click the **Network** icon in the top-right.
2. Select **Wi-Fi or Wired Settings**.
3. Choose a network → Enter password if required.

Linux Network Services:

Linux provides a wide range of **network services** essential for communication, administration, and hosting. These services can run in the background (as **daemons**) and respond to client requests via ports and protocols.

1. Common Linux Network Services

Service	Port	Description
SSH	22	Secure remote login
FTP	21	File Transfer Protocol
HTTP	80	Web server (Apache/Nginx)
HTTPS	443	Secure HTTP
DNS	53	Domain Name System
DHCP	67/68	Dynamic IP address assignment

NFS	2049	Network File System
SMB/Samba	137-139, 445	Windows file sharing
SMTP	25	Outgoing email
POP3/IMAP	110/143	Email retrieval

2. Checking Active Services and Ports

`ss -tuln` # Show listening TCP/UDP ports

`netstat -tuln` # (requires net-tools)

Check if a service is running:

`sudo systemctl status ssh`

3. Managing Services (Systemd)

Task	Command
------	---------

Start a service	<code>sudo systemctl start nginx</code>
Stop a service	<code>sudo systemctl stop ssh</code>
Enable at boot	<code>sudo systemctl enable apache2</code>
Disable	<code>sudo systemctl disable vsftpd</code>
Check status	<code>sudo systemctl status dhcpd</code>

4. Installing Common Network Services

SSH Server:

```
sudo apt install openssh-server # Debian/Ubuntu
```

```
sudo systemctl enable --now ssh
```

-

Web Server (Apache/Nginx):

```
sudo apt install apache2
```

```
sudo systemctl enable --now apache2
```

-

FTP Server (vsftpd):

```
sudo apt install vsftpd
```

-

Samba (SMB):

```
sudo apt install samba
```

-

5. Testing Network Services

Ping a remote host:

```
ping google.com
```

-

Check website availability:

```
curl -I http://localhost
```

-

Check open ports:

```
sudo ss -tln
```

-

6. Securing Network Services

Limit access via **firewalld**, **iptables**, or **ufw**:

```
sudo ufw allow 22/tcp
```

```
sudo ufw enable
```

Disable unused services to reduce attack surface:

```
sudo systemctl disable telnet
```

Regularly update packages:

```
sudo apt update && sudo apt upgrade
```

Setting Up a Printer in Linux:

Linux systems support printing through **CUPS (Common UNIX Printing System)**. It allows you to manage local and network printers via command line or web interface.

1. Install CUPS

On Debian/Ubuntu:

```
sudo apt update
```

```
sudo apt install cups
```

```
sudo systemctl enable --now cups
```

On RHEL/CentOS:

```
sudo dnf install cups
```

```
sudo systemctl enable --now cups
```

2. Access the CUPS Web Interface

Open a browser and go to:

<http://localhost:631>

If access is denied:

```
sudo cupsctl --remote-admin
```

3. Add a Printer (Web UI)

1. Go to <http://localhost:631>
2. Click on **Administration** → **Add Printer**
3. Log in with your username and password
4. Choose printer type:
 - Local (USB)
 - Network (IPP, LPD, SMB, etc.)
5. Select the model and driver (PPD file)
6. Set printer name, location, and description
7. Set sharing options

4. Add Printer (Command Line)

List available printers:

```
lpinfo -v
```

Add a printer:

```
sudo lpadmin -p MyPrinter -E -v usb://Printer/Device -m everywhere
```

Set default printer:

```
lpoptions -d MyPrinter
```

5. Print a Test Page

CLI:

```
lp /etc/hosts
```

GUI:

- Go to **Settings > Printers**
- Right-click your printer → **Properties > Print Test Page**

6. Manage Printers

Task	Command
List printers	<code>lpstat -p</code>
Cancel print jobs	<code>cancel -a</code>
Remove printer	<code>sudo lpadmin -x MyPrinter</code>

UNIT 3

Input and Output Redirection

In Linux, **redirection** is a way to change the standard input/output of commands.

► *Input Redirection (<)*

- Normally, commands take input from the keyboard.
- Input redirection allows you to take input from a file.

Example:

```
wc < myfile.txt
```

- This counts words in `myfile.txt` instead of waiting for keyboard input.

► **Output Redirection (>, >>)**

- Normally, command output shows on the screen.
- **>** redirects output to a file, **overwriting** it.
- **>>** appends output to the file without deleting existing data.

Examples:

```
ls > list.txt
```

```
echo "Hello" >> greetings.txt
```

-

► **Error Redirection (2>, 2>>)**

- Errors can also be redirected.
- **2>** sends errors to a file.
- **2>>** appends errors to a file.

Example:

```
ls nonfile 2> errors.txt
```

-

► **Filters**

- A **filter** processes input and produces output.
- Common filters: **grep**, **sort**, **uniq**, **cut**, **tr**

Example:

```
cat data.txt | grep "apple"
```

-

► Pipes (|)

- Pipes connect the output of one command to the input of another.

Example:

```
ls | sort
```

Great question! Let's break down those **command pipelines** and explain how **uniq**, **cut**, and **tr** work too. These are all examples of **filters**—commands that process text data.

```
cat data.txt | grep "apple"
```

🔍 *What it does:*

- **cat data.txt** shows the content of the file.
- The **|** pipe sends that output to the next command.
- **grep "apple"** searches for lines containing the word **"apple"**.

Example:

Suppose **data.txt** contains:

apple pie

banana smoothie

pineapple tart

grape juice

Command:

```
cat data.txt | grep "apple"
```

Output:

apple pie

pineapple tart

Tip: You can also skip `cat` and just use:

`grep "apple" data.txt`

`ls` | `sort`

What it does:

- `ls` lists files and directories.
- `sort` arranges them alphabetically.

Example:

Suppose your directory has these files:

zebra.txt

apple.txt

orange.txt

Command:

`ls | sort`

Output:

apple.txt

orange.txt

zebra.txt

uniq

What it does:

Removes **duplicate lines** from **sorted** input.

Example:

File `names.txt`:

Alice

Bob

Bob

Charlie

Charlie

Charlie

Command:

```
sort names.txt | uniq
```

Output:

Alice

Bob

Charlie

`uniq` only removes *consecutive* duplicates, so use `sort` before it!

`cut`

What it does:

Extracts specific columns or characters from lines.

Example:

File `data.csv`:

John,25,New York

Alice,30,London

Bob,22,Paris

Command to get just the names:

```
cut -d ',' -f 1 data.csv
```

Output:

John

Alice

Bob

- `-d ','` sets the delimiter as comma.
- `-f 1` selects the first field (column).

tr

🔧 *What it does:*

Translates or deletes characters.

Example 1: Convert lowercase to uppercase

```
echo "hello" | tr 'a-z' 'A-Z'
```

Output:

HELLO

Example 2: Remove digits

```
echo "abc123" | tr -d '0-9'
```

Output:

abc

Networking in Linux

Linux offers several tools for network configuration and diagnostics.

➤ ***Network Connectivity***

Test network connection with:

```
ping google.com
```

-

➤ ***IP Address***

Find your system's IP address:

ip a

-

Or:

ifconfig

-

► **Accessing Remote Systems**

Use **ssh** (Secure Shell):

ssh username@remote_ip

- **ssh**: The command for Secure Shell.
- **username**: The user account on the remote machine.
- **remote_ip**: The IP address (or domain name) of the remote machine.
- **Example** : ssh alice@192.168.1.10

This attempts to log in to the computer at **192.168.1.10** using the username **alice**

► **Transferring Files**

Using **scp** (secure copy):

scp file.txt username@remote_ip:/path/

- **scp**: Secure copy command.
- **file.txt**: The file you want to copy (from local).

- **username@remote_ip**: The user and remote machine you're sending the file to.
- **:/path/**: The destination directory **on the remote system**.
- **Example** : `scp report.pdf john@192.168.1.50:/home/john/Documents/`

This will copy **report.pdf** from your current local directory to the **/home/john/Documents/** folder on the remote system at IP **192.168.1.50** under the user **john**.

► **Internet Configuration**

- Edit network config files (**/etc/network/interfaces** or use **nmcli/nmtui**)

Restart network service after changes:

```
sudo systemctl restart NetworkManager
```

What is a Network Configuration File?

A **network config file** is a **system file** in Linux that stores the settings used to configure your network interface (like your computer's Ethernet or Wi-Fi adapter).

These settings tell the system:

- What **IP address** to use (static or dynamic)
- What **gateway** and **DNS** servers to use
- Whether to **automatically connect** on boot

Common Network Configuration Methods

Depending on your Linux distro, there are a few different ways to configure the network:

1. **/etc/network/interfaces** (used in older Debian/Ubuntu systems)

This file is edited manually.

Example: Static IP Configuration

```
auto eth0
```

```
iface eth0 inet static
```

```
address 192.168.1.100
```

```
netmask 255.255.255.0
```

```
gateway 192.168.1.1
```

```
dns-nameservers 8.8.8.8
```

Example: DHCP (Automatic IP)

```
auto eth0
```

```
iface eth0 inet dhcp
```

After editing:

```
sudo systemctl restart networking
```

Static IP vs DHCP

Feature	Static IP	DHCP (Dynamic Host Configuration Protocol)
IP Assignment	Manually set by the user/admin	Automatically assigned by a DHCP server
Configuration	You must configure IP, gateway, DNS manually	No manual setup needed
IP Changes	IP address stays the same	IP can change every time the system connects
Use Case	Servers, printers, routers (need consistency)	Home PCs, phones, temporary devices
Setup Effort	More time-consuming, prone to mistakes	Quick and easy
Reliability	Always reachable at the same IP	Not guaranteed unless DHCP reservation is used

When to Use Static IP

- For devices that provide **services** like:
 - Web servers
 - Database servers
 - Network printers
 - NAS (file servers)
- Helps in **port forwarding** and **network access control**

Example (Manual Config)

address: 192.168.1.100

netmask: 255.255.255.0

gateway: 192.168.1.1

dns: 8.8.8.8

When to Use DHCP

- For **general-purpose devices**:
 - Laptops, smartphones, guest devices
 - Minimizes configuration work
 - The IP can change, which is okay for temporary devices
-

A Common Practice:

- Use DHCP **for most devices**
 - Use Static IP **for servers and network-critical devices**
-

Gateway (Default Gateway)

What is it?

The gateway is the device that connects your local network to the outside world (like the internet). It usually refers to your router.

Think of it like:

“The exit door from your house to the street.”

In Network Config:

gateway: 192.168.1.1

This tells your system:

“To reach anything outside my local network, send data to 192.168.1.1 (the router).”

DNS (Domain Name System)

What is it?

DNS translates domain names (like **google.com**) into IP addresses (like **142.250.180.78**), because computers communicate using IPs, not names.

Think of it like:

“The contact list on your phone—so you don’t have to remember numbers.”

In Network Config:

dns-nameservers: 8.8.8.8

This tells your system:

“Ask 8.8.8.8 (Google's DNS server) to resolve domain names for me.

“Ask 8.8.8.8 (Google's DNS server) to resolve domain names for me.”

Translation:

"When I type a website name like **google.com**, I want my system to contact Google's DNS server at **IP address 8.8.8.8** to find out **what IP address that domain points to.**"

Why This Is Necessary:

You type:

ping google.com

-
- Your system needs the **IP address** of **google.com** to send data.
- But you only know the **name**.
- So, it sends a request to the **DNS server** (like **8.8.8.8**) asking:

"What's the IP address of **google.com**?"

The DNS server replies with something like:

142.250.180.78

-

Now your system can reach the server at that IP.

Why 8.8.8.8?

- It's a **public DNS server** run by Google.
- It's **free, fast, and reliable**.
- Anyone can use it.

Other public DNS options include:

- **1.1.1.1** (Cloudflare)
- **9.9.9.9** (Quad9)
- **8.8.4.4** (Google's secondary)

Where You Configure This

You set it in your network settings or in config files like:

/etc/resolv.conf (traditional way)

nameserver 8.8.8.8

-
- Or via NetworkManager tools like **nmcli**.

2. **nmcli** (command-line tool for NetworkManager)

Used on modern systems like Fedora, CentOS, Ubuntu (desktop), etc.

*Example: Set static IP using **nmcli***

nmcli con mod "Wired connection 1" ipv4.addresses 192.168.1.100/24

- This assigns your system the IP `192.168.1.100`.
- `/24` means the **subnet mask** is `255.255.255.0`.

```
nmcli con mod "Wired connection 1" ipv4.gateway 192.168.1.1
```

- This tells the system to use `192.168.1.1` as the **router** (default gateway).

```
nmcli con mod "Wired connection 1" ipv4.dns 8.8.8.8
```

- This sets **Google's DNS server** to resolve domain names.

```
nmcli con mod "Wired connection 1" ipv4.method manual
```

- This tells NetworkManager **not** to use DHCP, but use **manual/static IP** settings.

```
nmcli con mod "Wired connection 1" ipv4.method auto
```

- This tells NetworkManager to use DHCP

```
nmcli con up "Wired connection 1"
```

- This restarts the connection using your new static IP configuration.

3. **nmtui** (text-based user interface for NetworkManager)

An easy-to-use **GUI inside the terminal**.

Run:

```
nmtui
```

You'll get a simple menu to:

- Edit a connection
 - Set static/dynamic IP
 - Enable/disable interfaces
-

Process Control(Still to Study)

Processes are instances of running programs.

► *Identifying Processes*

- Use `ps`, `top`, or `htop` to view running processes.

Example:

```
ps aux
```

-

► *Managing Processes*

Kill a process using PID:

```
kill 1234
```

-

Niceness/priority adjustment:

```
nice -n 10 command
```

```
renice -n 5 -p 1234
```

-

► *Background Processing*

Run a job in the background by appending `&`:

```
command &
```

-

► *Putting Jobs in Background*

Stop a job with `Ctrl+Z`, then:

```
bg
```

-

To see background jobs:

jobs

Offline File Storage

Working with removable media and understanding system boot processes.

► *Storing Files to Media*

- Use USB or external drives.

Mounting example:

```
sudo mount /dev/sdb1 /mnt
```

```
cp file.txt /mnt
```

```
sudo umount /mnt
```

-

► *Booting Process and User*

- Linux Boot Sequence:
 1. BIOS/UEFI starts the system.
 2. Bootloader (GRUB) loads kernel.
 3. Kernel initializes hardware.
 4. `init` or `systemd` starts services.
 5. Login prompt appears.

- User logs in and starts using the system.
-

UNIT 4

1. Introduction to Networking

Networking is the practice of connecting computers and other devices to share resources and communicate.

- **Types of Networks:**
 - **LAN (Local Area Network)** – Small network, usually within a building.
 - **WAN (Wide Area Network)** – Large network, often spanning cities or countries.
 - **IP Addressing:** Unique identifier for each device on a network (e.g., 192.168.1.10).
 - **Example:** Two Linux PCs connected over a switch in a lab, communicating via IP addresses.
-

2. Networking Concepts

- **Hostname:** Human-readable name of a computer in a network.
 - View with `hostname` command.
- **Ports:** Logical endpoints for communication (e.g., HTTP uses port 80).
- **Protocols:**
 - **TCP (Transmission Control Protocol)** – Reliable communication.

- **UDP (User Datagram Protocol)** – Faster, less reliable.
 - **Tools:**
 - `ping`: Test connectivity.
 - `ifconfig` or `ip addr`: View IP configuration.
 - `netstat -tuln`: View open ports and services.
-

3. Internet Network Services

- **SSH (Secure Shell)**: Secure remote login. `ssh user@host`
 - **FTP/SFTP**: File transfers.
 - **HTTP/HTTPS**: Web access.
 - **DNS**: Resolves domain names to IP addresses.
-

4. Dynamic DNS (DDNS)

- **Dynamic DNS** updates DNS records automatically when IP addresses change.
 - Useful for home or small business networks with dynamic IPs.
 - **Example**: `ddclient` – an open-source DDNS client.
-

5. Electronic Messaging

- **Email System Components:**

- **MTA (Mail Transfer Agent)** – e.g., `Postfix`, `Sendmail`
 - **MUA (Mail User Agent)** – e.g., `Thunderbird`
 - **IMAP/POP3**: Protocols to retrieve emails.
 - **Basic Setup Example**: Configure Postfix to send email from a Linux server.
-

6. Apache Web Server

- Open-source HTTP server for hosting websites.
 - **Install**: `sudo apt install apache2` (Debian/Ubuntu)
 - **Default Web Page**: `/var/www/html/index.html`
 - **Start Service**: `sudo systemctl start apache2`
 - Access via `http://localhost` or server IP.
-

7. NIS and Network File Sharing

a) NIS (Network Information Service)

- Centralized user management across Linux systems.
- Allows shared user credentials across machines.
- **Example**: NIS master server distributes `/etc/passwd` entries.

b) Network File Sharing

- Sharing files across a network using different protocols:
 - **NFS (Network File System)** – Unix/Linux file sharing.
 - **Install:** `sudo apt install nfs-kernel-server`
 - **Export a directory:** Add to `/etc/exports`, restart NFS service.
 - **Samba** – Share files between Linux and Windows.
 - **Install:** `sudo apt install samba`
 - **Configure:** Edit `/etc/samba/smb.conf` to share directories.
-

8. Security

a) Defining System Security Policies

- Set rules for system access and protection.
- Example policies: Password complexity, login restrictions, firewall usage.

b) System Authentication Services

- Use of services like **LDAP**, **Kerberos**, and **PAM** for login and user management.

c) Securing Services

- Disable unused services.
- Use **firewalls** (`ufw`, `iptables`) to block unwanted traffic.
- Use **SELinux** or **AppArmor** for enforcing security policies.

d) Securing Data and Communication

- Use **encryption**: [ssh](#), [https](#), [gpg](#)
 - **TLS/SSL** for secure web communication.
 - Regularly update packages to patch vulnerabilities.
-