From OS Concepts to Programming

Unix/Linux-Based Operating System Concepts & Programming

Written by: Md. Tariqul Islam

Title: Unix/Linux-Based Operating System Concepts & Programming

Written By:

Md. Tariqul Islam

Publisher: Self-Published

Edition: 1st Edition

About Author

MD. TARIQUL ISLAM

B.Sc.Eng. in CSE, NWU, M.Sc.Eng. in ICT, KUET, Ph.D. Fellow in ICT, BUET
Lecturer & Coordinator, Department of Computer Science and Engineering
University of Global Village (UGV), Barishal 8200, Bangladesh.
Mobile: +880-1842733104
Email: tariq.ugv@gmail.com
1024314003@iict.buet.ac.bd
Official Site: www.faculty.ugv.edu.bd/tariqul
Personal Site: www.sites.google.com/view/tariq-ugv
Github: www.github.com/tariqulnwu
IEEE Xplore: https://ieeexplore.ieee.org/author/370861755138
Google Scholar: https://scholar.google.com/citations?user=bo1z-SUAAAAJ

Linux cd Command — Full Lecture with Examples & Flags

● cd stands for Change Directory.
● It is a built-in command in the shell (like Bash).
● It is used to navigate between directories in the Linux file system.

cd [directory_path]


```
cd /home/user/Documents  # Go to an absolute path
cd Projects/Website      # Go to a relative path
cd                       # Go to the home directory
```


Command	Description
cd	Goes to the home directory (/home/user)
cd ~	Also goes to the home directory
cd ..	Goes one level up (parent directory)
cd ../..	Go up two levels
cd -	Goes to the previous working directory
cd /	Goes to the root directory (/)
cd ~username	Goes to another user's home (if permitted)
cd ./folder	Goes into a subfolder in the current directory

Symbol	Meaning
.	Current directory
..	Parent directory (one level up)

~
Home directory
-
Last visited directory
🔹 5. Flags / Options
While cd has no traditional flags like -l, -a, etc., some special usage cases exist:
Usage
Effect
cd -
Switches to the previous directory and prints it
cd ~
Goes to the current user's home directory
cd ~username
Goes to a different user's home directory (if accessible)
✅ Note: cd is a shell builtin, so man cd may not work. Instead, use:
bash
CopyEdit
help cd
🔹 6. Checking Current Directory
Use pwd (print working directory):
bash
CopyEdit
pwd
It will show the absolute path of your current location.
🔹 7. Practical Examples
cd ~ # Go to home
cd /var/log # Go to system logs
cd ../ # Go one level up
cd - # Toggle between current and last dir
cd ~/Downloads # Go to Downloads inside home
🔹 8. Common Errors & Fixes
Error
Cause
Fix
cd: No such file or directory
Wrong path or typo
Double-check the path
cd: Permission denied
Trying to enter protected folder
Use sudo or check permissions
cd: Not a directory
Trying to cd into a file
Only use directories with cd
🔹 9. Real-Life Use Cases

What is Shell Scripting?
Shell scripting is the process of writing a sequence of commands in a file to automate tasks in Unix-like systems. These scripts are commonly used for system administration, backups, automation, and more.

What is a Shell?
The shell is a command-line interface that allows users to interact with the operating system. Common shell types include sh, bash, and zsh.

What is a Shebang (#!)?
The first line of a shell script usually starts with a special sequence called a shebang: #!/bin/sh
This tells the system which shell to use when executing the script. If this line is missing, some execution methods (like ./filename.sh) may not work correctly.

Lab Section: Step-by-Step Instructions

Step 1: Open the Terminal and go to the Desktop
Type the following command:
cd Desktop

Step 2: Create a New Shell Script File
To create a new file using Nano editor, type:
nano new.sh
This command opens the Nano text editor and creates a file named new.sh.

Step 3: Write a Simple Script
Inside Nano, type the following lines:
#!/bin/sh
echo "Hello World"
echo "This is my first shell script!"

Step 4: Save and Exit Nano

- Press Ctrl + O to save
- Press Enter to confirm the file name
- Press Ctrl + X to exit the editor

Step 5: Check the File in Desktop
Type:
ls
If the file name appears in white color, it means it does not have execution permission.

Step 6: Check File Permissions
Type:
ls -la
This will display the permission status of all files. You may see that your script has -rw-r--r-- meaning it is readable and writable, but not executable.

Step 7: Make the Script Executable
To give execution permission, type:
sudo chmod +x new.sh
It will ask for the root password. Type your password and press Enter.

Step 8: Verify Execution Permission
Now type:
ls -la
You should now see -rwxr-xr-x for your script file, which means it has execute permission.

Step 9: Run the Script
You can run the script using either of the following two methods:

1. bash new.sh
2. ./new.sh

Note: The second method (./new.sh) will only work if the shebang line #!/bin/sh is present at the top of

Introduction to UNIX/Linux

Reference Book:

Reference Site: https://www.tutorialspoint.com/unix/index.htm
Source Website: https://www.hpc.iastate.edu/guides/unix-introduction

1. Architecture of Unix/Linux-Based OS (UNIX/Linux) Site Link
2. Types of Unix/Linux-Based OS (UNIX/Linux) Slide Link

3. Process Control Block (PCB) in Unix/Linux
   In the Linux operating system, a Process Control Block (PCB) is a data structure used by the kernel to store all the information necessary to manage and track a process. In Linux, this is primarily represented by the `task_struct` data structure.

The `task_struct` contains a comprehensive set of attributes and information about a process, including:

- **Process State:** Indicates the current state of the process (e.g., running, ready, waiting, stopped, zombie).

- **Process ID (PID):** A unique identifier assigned to each process.
- **Parent Process ID (PPID):** The PID of the process that created it.
- **Program Counter:** Stores the address of the next instruction to be executed.
- **CPU Registers:** Contains the values of the CPU registers when the process was last running, crucial for context switching.
- **Memory Management Information:** Details about the process's memory space, including pointers to page tables and allocated memory regions.
- **Scheduling Information:** Data used by the scheduler, such as process priority, CPU time consumed, and scheduling policy.
- **I/O Status Information:** Information about open files, I/O devices in use, and pending I/O requests.
- **Signals:** Information regarding signals received and signal handlers.
- **User and Group IDs:** Identifiers for the user and group that own the process, determining its permissions.

The PCB is essential for the operating system to perform multitasking and manage resources efficiently. When the operating system switches between processes (context switching), it saves the state of the currently running process into its PCB and loads the state of the next process from its respective PCB, allowing for seamless transitions.

4.

QN: Why is a Command-Line Interface (CLI)-based operating system generally faster and more efficient than a Graphical User Interface (GUI)-based system? What are the main benefits of using a CLI-based OS for remote access and automation?

A Command-Line Interface (CLI)-based operating system is generally faster and more efficient than a Graphical User Interface (GUI)-based system due to several key reasons:

1. Lower Resource Consumption

- CLI-based systems do not require a graphical environment, reducing the need for RAM, CPU power, and storage.

- GUIs demand additional processes for rendering windows, icons, animations, and other visual elements, which increase system overhead.

2. Faster Execution

- Commands in CLI are executed directly without needing multiple clicks or navigating through menus.
- A well-optimized CLI workflow can automate repetitive tasks efficiently using scripts.

3. Greater Stability and Reliability

- Since CLIs do not rely on a GUI layer, they are less prone to crashes and system slowdowns.
- CLI-based OSs are widely used in servers and embedded systems where uptime and stability are critical.

4. Efficient System Management

- CLI provides fine-grained control over the system, making tasks like network configuration, user management, and file manipulation more efficient.
- Remote administration is faster as it does not require graphical rendering.

Main Benefits of Using a CLI-Based OS for Remote Access and Automation

1. Lightweight Remote Access

- CLI allows remote access using SSH (Secure Shell), which consumes minimal bandwidth compared to GUI-based remote desktop solutions.
- Users can perform system management and troubleshooting from virtually anywhere without needing a graphical interface.

2. Powerful Automation

- CLI supports scripting (e.g., Bash, PowerShell, Python) to automate tasks like software updates, log analysis, backups, and deployments.
- Automation minimizes human errors and increases productivity.

3. Security Advantages

- CLI-based access is generally more secure as it reduces the attack surface compared to GUI-based tools that may require open ports for remote desktops (e.g., RDP or VNC).
- Scripts and commands can be executed with privileged access control without exposing unnecessary graphical components.

4. Scalability for System Administration

- CLI allows managing multiple remote servers simultaneously using tools like Ansible, SSH scripts, and cron jobs.
- System administrators can deploy updates, modify configurations, or troubleshoot issues across hundreds of machines efficiently.

Types of UNIX or Linux-based Operating Systems:

1. Traditional UNIX Systems

These are based on the original UNIX source code.

- AIX (Advanced Interactive eXecutive) – Developed by IBM
- HP-UX (Hewlett Packard UNIX) – Developed by Hewlett-Packard
- Solaris (now OpenIndiana) – Originally from Sun Microsystems

---

🔹 **2. Linux Distributions (Linux Distros)**

These are open-source UNIX-like systems built around the Linux kernel.

Desktop/General-Purpose Distros

- Ubuntu – User-friendly, great for beginners
- Linux Mint – Based on Ubuntu; lightweight and beginner-friendly

- Fedora – Cutting-edge technologies, backed by Red Hat
- Debian – Known for stability and free software
- Arch Linux – Lightweight and highly customizable
- Zorin OS – Designed for Windows users switching to Linux

Enterprise/Server Distros

- Red Hat Enterprise Linux (RHEL) – Commercial, used in enterprise servers
- CentOS / CentOS Stream – Previously a free version of RHEL (now replaced by CentOS Stream)
- Rocky Linux / AlmaLinux – RHEL-compatible replacements for CentOS
- SUSE Linux Enterprise Server (SLES) – Used in business environments

Penetration Testing / Hacking / Security Distros

- Kali Linux – Security and penetration testing
- Parrot OS – Forensics, penetration testing, and privacy

🔹 **3. Lightweight Linux Distros**

Best for old or low-resource hardware.

- Puppy Linux
- Lubuntu

- Tiny Core Linux
- antiX

🔹 **4. Special Purpose Distros**

Designed for specific uses.

- TrueNAS CORE – NAS (Network Attached Storage) server OS (based on FreeBSD, a UNIX-like OS)
- Raspberry Pi OS – For Raspberry Pi devices
- Tails – Focuses on anonymity and privacy
- Ubuntu Server – Lightweight version of Ubuntu for server use

---

🔹 **5. BSD Systems (UNIX-like)**

Although not Linux, they are UNIX or UNIX-like.

- FreeBSD
- OpenBSD
- NetBSD
- DragonFly BSD

Why We Use Kali Linux in Our Lab

1. Pre-installed Security Tools

Kali Linux includes a wide range of pre-installed cybersecurity tools used for:

- Penetration testing
- Vulnerability assessment
- Digital forensics
- Wireless attacks
- Password cracking
- Network sniffing

Some commonly used tools in Kali are Nmap, Wireshark, Metasploit, Burp Suite, and John the Ripper. This makes it an ideal platform for hands-on security practice.

2. Open Source and Free

Kali Linux is completely free and open source. This makes it cost-effective for educational institutions and students.

3. Widely Used in the Cybersecurity Industry

Kali is one of the most commonly used operating systems among ethical hackers, penetration testers, and cybersecurity professionals. Using it in the lab prepares students with industry-relevant skills.

4. Customizable and Lightweight

Kali Linux is lightweight and can be run on physical machines, virtual machines (VMs), or through live USBs. This flexibility makes it easy to deploy in a controlled lab environment.

5. Safe Environment for Ethical Hacking

Kali provides a controlled and isolated environment, allowing students to practice ethical hacking techniques without posing a risk to real-world systems.

6. Regular Updates and Community Support

Maintained by Offensive Security, Kali Linux receives frequent updates and has strong community support, ensuring students have access to the latest tools and features.

How to Install Kali Linux (Step-by-Step Guide)

Prerequisites

Before you begin, ensure the following:

- A computer with at least **2 GB RAM** and **20 GB free disk space**
- **Kali Linux ISO** (download from https://www.kali.org/get-kali/)
- A **bootable USB drive** (minimum 8 GB) for installation
- Optional: **VirtualBox/VMware** if installing virtually
- **Back up your important data** if you are installing on a physical system

---

Installation Methods Overview

1. Installation on Physical System via Bootable USB
2. Installation in a Virtual Machine
3. **Dual Boot Installation with Windows**

---

1. Installation Using Bootable USB (Physical System)

Step 1: Download Kali ISO

- Visit: https://www.kali.org/get-kali/

- Download the installer image (64-bit recommended)

Step 2: Create Bootable USB

Use **Rufus** (Windows tool):

- Download from https://rufus.ie
- Select the USB drive
- Select the Kali ISO file
- File system: FAT32
- Click **Start** and wait for the process to complete

Step 3: Boot from USB

- Restart your PC
- Enter the BIOS/Boot Menu (usually by pressing **F2**, **F12**, **ESC**, or **DEL**)
- Select the USB drive as the boot device

Step 4: Start Installation

- Choose **Graphical Install**
- Select language, location, and keyboard layout

Step 5: Configure System

- Set hostname (e.g., kali)

- Set domain name (optional)
- Create a new user and password

Step 6: Partition Disk

- Use "**Guided – Use entire disk**" (for full installation)
- Write changes to disk and continue

Note: This will erase the entire disk. For dual booting, refer to section 3.

Step 7: Install System and GRUB

- Choose to install the **GRUB boot loader**
- Select the appropriate disk (e.g., /dev/sda)

Step 8: Reboot

- Remove USB when prompted
- Kali Linux is now installed

---

2. Installation Using Virtual Machine (e.g., VirtualBox)

Step 1: Install VirtualBox

- Download from https://www.virtualbox.org

Step 2: Create New VM

- Name: Kali Linux
- Type: Linux → Debian (64-bit)
- RAM: At least 2048 MB
- Create a new virtual hard disk (20 GB+)

Step 3: Load ISO

- Go to **Settings → Storage** → Mount the Kali ISO

Step 4: Start VM and Install

- Follow the same steps as the physical installation

---

3. Dual Boot Installation with Windows

Dual booting allows you to run **both Windows and Kali Linux** on the same system, choosing between them at startup.

Step 1: Prepare Windows System

- Log in to Windows
- **Create free space** from your hard drive:
    - Open Disk Management
    - Shrink volume to free up **at least 20–30 GB**
    - Leave the space **unallocated**

Step 2: Disable Fast Startup and Secure Boot

- Go to Control Panel → Power Options → Choose what the power buttons do → Disable **Fast Startup**
- Reboot into BIOS and disable **Secure Boot** (if required)

Step 3: Boot with Kali Linux USB

- Restart and boot into Kali using the USB
- Select **Graphical Install**

Step 4: Follow Standard Installation

- Set language, region, user credentials, etc.

Step 5: Select "Manual Partitioning"

- Choose the **unallocated space** created earlier
- Create the following partitions:
    - **/ (root)**: 20 GB or more
    - (Optional) **swap**: Same as RAM
    - (Optional) **/home**: For user files

Step 6: Install GRUB Bootloader

- Install the bootloader on the **primary disk** (e.g., /dev/sda)
- GRUB will detect Windows and add it to the boot menu

Step 7: Reboot

- On startup, the **GRUB menu** will appear, allowing you to select between **Kali Linux** and **Windows**

How to Practice Linux Commands Online

1. Webminal – The GNU/Linux Terminal

Website: https://webminal.org

Features:

- **Browser-based Linux terminal**
- **Practice basic and advanced commands**
- **Write shell scripts and run them**
- **Includes tutorials and assignments**

How to Use:

1. **Visit the website and create a free account**
2. **Access the terminal directly from the browser**
3. **Start typing commands like `ls`, `cd`, `mkdir`, `echo`, etc.**

---

2. Jupyter Terminal (via TutorialsPoint)

Website: https://www.tutorialspoint.com/unix_terminal_online.php

Features:

- **Interactive online Linux terminal**
- **Ideal for Unix/Linux command practice**
- **No sign-up required**

How to Use:

1. **Visit the page**
2. **Wait for the terminal to load**
3. **Start typing and testing commands immediately**

---

3. CoCalc

Website: https://cocalc.com

Features:

- **Full Linux terminal environment**
- **Collaborative coding and scripting**
- **Supports bash, Python, LaTeX, etc.**

How to Use:

1. **Sign up for a free account**
2. **Create a new project**
3. **Launch the terminal and begin practicing**

---

4. JS/Linux (Copy.sh)

Website: https://copy.sh/v86/

Features:

- **Emulates Linux in-browser**
- **Fast and lightweight**
- **Ideal for basic command-line experience**

How to Use:

1. **Go to the website**
2. **Select a Linux distribution (e.g., Buildroot or BusyBox)**
3. **Use the in-browser terminal for command testing**

---

5. LinuxZoo

Website: https://linuxzoo.net

Features:

- **Remote access to real Linux systems**
- **Supports shell, networking, and scripting practice**
- **Useful for learning system administration**

How to Use:

1. **Register for a free account**
2. **Launch your virtual machine**
3. **Access terminal and practice real Linux scenarios**

Day 01 – Useful Keyboard Shortcuts in Kali Linux Terminal (Simple Format)

In this lab session, students will explore key terminal shortcuts and understand the Linux command prompt. These keyboard shortcuts are useful for enhancing command-line efficiency during real-time tasks in Kali Linux or any Unix-like environment.

Terminal Shortcuts and Their Uses

1. **Ctrl + Alt + T**
   → Open a new terminal window.

2. **Ctrl + Shift + N**
   → Open another new terminal window (alternative method).

3. **Ctrl + Shift + T**
   → Open a new tab inside the same terminal window.

4. **Ctrl + Shift + W**
   → Close the current terminal window or tab.

5. **Ctrl + L**
   → Clear the terminal screen.

6. **Ctrl + Shift + +**
   → Zoom in (increase the terminal font size).

7. **Ctrl + -**
   → Zoom out (decrease the terminal font size).

8. **Ctrl + C**
   → **Terminate or exit a running process in the terminal.**
   Example: Run `ping google.com`, then press `Ctrl + C` to stop it.

9. **Tab**
   → **Auto-complete file or folder names and commands.**
   Example: Type `cd Dow` and press `Tab` — it will complete to `Downloads` (if available).

10. **Super Key (Windows Key)**
    → **Opens the application menu or search bar in Linux.**

11. **Super Key + Left Arrow**
    → **Move the current terminal window to the left half of the screen.**

Terminal Prompt Explanation

When you open a terminal, you may see something like this:

```
tariq@kali:~$
```

Explanation:

- `tariq` → **Username (current user)**
- `kali` → **Hostname (your computer name)**
- `:` → **Separator**
- `~` → **Home directory**

- `$` → Regular user prompt
- `#` → Root user or sudo user prompt

Practice Tasks

1. Open a terminal using `Ctrl + Alt + T`.
2. Open a new tab using `Ctrl + Shift + T`.
3. Run `ls` to list files, then clear the screen using `Ctrl + L`.
4. Run `ping google.com`, stop it with `Ctrl + C`.
5. Try auto-completion: Type `cd Dow` and press `Tab`.
6. Run `whoami` to see the current user.
7. If allowed, run `sudo -i` to switch to root and observe the change in prompt (`$` changes to `#`).

Day 02 – Managing Files & Directories in Kali Linux

In this section, we will learn basic commands to manage files and directories within Kali Linux.

---

1. Navigating Directories

- `pwd`
  Displays the current working directory (path).

- `cd [directory_name]`
  Changes the directory.

- `cd ..`
  Goes one step back in the directory structure.

- `cd ~`
  Takes you to your home directory.

---

2. Creating Files

- `cd Desktop`
  Navigate to the Desktop directory.

- `cat > test.text`
- `Press Ctrl + D to save and exit.Press Ctrl + D to save and exit.`
  Creates a new text file named `test.text` and allows you to add content.

- `touch test.txt`

  Creates a new empty file named `test.txt`. Useful when you just want to create a blank file quickly.

- `echo "Hello Kali" > test.txt`

  Creates a new file named `test.txt` (or overwrites it if it exists) and writes the text `"Hello Kali"` into it.

- **`cat test.text`**
  Displays the content of the `test.text` file.

THE `ls` COMMAND IN LINUX

🔷 1. INTRODUCTION

The `ls` command is one of the most frequently used commands in Linux. It is used to list the contents of a directory, showing files, folders, and providing detailed information about each item if required.

🔷 2. DEFINITION

✅ `ls` Command:
Meaning: *"list"*
Function: Displays the names of files and directories within the file system.

🔷 3. PURPOSE OF `ls`

- **To view files and folders in a directory**
- **To check permissions, ownership, and sizes**

- 🔹 To sort files based on time, size, etc.
- 🔹 To list hidden files
- 🔹 To navigate efficiently within the terminal

🔷 4. SYNTAX

```
ls [options] [directory]
```

➡️ Options are flags that modify the behavior of `ls`.
➡️ directory is the path you want to list. If omitted, it lists the current directory.

🔷 5. WHY USE OPTIONS OR FLAGS?

Flags (options) are used to:

✔️ View more details (permissions, size)
✔️ Sort outputs (by time, size)
✔️ Show hidden files
✔️ Display outputs in readable formats

Without options, `ls` shows only file and folder names in a minimal view.

🔷 6. IMPORTANT OPTIONS WITH DETAILS & EXAMPLES

✅ (i) ls

- Description: Lists files and directories in the current path.
- Example:

```
ls
```

- **Output: Shows names only, arranged alphabetically by default.**

✅ **(ii) ls -l**

- **Description: Long listing format showing detailed information.**
- **Example:**

```
ls -l
```

- **Output Explanation:**

diff

CopyEdit

```
drwxr-xr-x 2 root root 4096 Jul 25 20:35 Documents
-rw-r--r-- 1 user user  567 Jul 20 15:10 notes.txt
```

- **Columns breakdown:**

Column	Description
drwxr-xr-x	File type & permissions
2	Number of hard links
root	Owner
root	Group
4096	Size in bytes
Jul 25 20:35	Modification date & time
Documents	File or directory name

- **d at start means directory.**
- **- at start means file.**

✅ **(iii) ls -a**

- **Description: Shows all files, including hidden files (those starting with .).**

- **Example:**

bash

CopyEdit

```
ls -a
```

- **Output: Includes `.` (current directory) and `..` (parent directory) plus hidden files like `.bashrc`.**

✅ **(iv) ls -la or ls -al**

- **Description: Combines `-l` and `-a` to show detailed list including hidden files.**
- **Example:**

```
ls -la
```

✅ (v) ls -lh

- **Description: Shows file sizes in human-readable format (KB, MB).**
- **Example:**

```
ls -lh
```

- **Output: Instead of bytes (e.g. 4096), shows as 4.0K, making it easier to read.**

✅ **(vi) ls -R**

- **Description: Lists all directories and subdirectories recursively.**
- **Example:**

```
ls -R
```

- **Output: Shows entire directory tree from the current folder.**

✅ **(vii) ls -t**

- **Description: Sorts files by modification time, newest first.**
- **Example:**

```
ls -lt
```

✅ **(viii) ls -S**

- **Description: Sorts files by size, largest first.**
- **Example:**

```
ls -lS
```

✅ **(ix) ls -r**

- **Description: Reverses the order of listing.**
- **Example:**

```
ls -lr
```

✅ ***(x) ls -d /***

- **Description: Shows only directories.**
- **Example:**

```
ls -d */
```

✅ **(xi) ls -i**

- **Description:** **Displays inode numbers of files.**
- **Example:**

```
ls -i
```

- **Output: Shows each file's inode, a unique identifier in the filesystem.**

✅ **(xii) ls -F**

- **Description:** **Appends symbols to indicate file types:**
  - **`/` for directories**
  - **`*` for executables**
- **Example:**

```
ls -F
```

🔷 **7. COMBINING OPTIONS**

You can combine multiple options together for powerful outputs.

- **Example:**

```
ls -lhS
```

- **Meaning: Long listing + human-readable sizes + sorted by size.**

🔷 8. PRACTICAL EXERCISES

1. **List all files including hidden files.**
2. **Display files with details sorted by size.**
3. **Show directories only.**
4. **List all files in home directory recursively.**
5. **Explain each column of `ls -l` output.**

🔷 9. SUMMARY

✅ **`ls` is the primary command to view directory contents.**
✅ **Options modify how information is displayed.**
✅ **Combining options increases command efficiency.**
✅ **Always use `man ls` to explore full command capabilities.**

Linux `cd` Command — Full Lecture with Examples & Flags

🔹 1. What is `cd`?

- `cd` stands for Change Directory.
- It is a built-in command in the shell (like Bash).
- It is used to navigate between directories in the Linux file system.

🔹 2. Basic Syntax:

```
cd [directory_path]
```

🔸 **Examples:**

```
cd /home/user/Documents  # Go to an absolute
  path

cd Projects/Website      # Go to a relative path
```

```
cd                    # Go to the home directory
```

🔹 3. Common Uses of `cd`

Command	Description
`cd`	Goes to the home directory (`/home/user`)
`cd ~`	Also goes to the home directory
`cd ..`	Goes one level up (parent directory)
`cd ../..`	Go up two levels
`cd -`	Goes to the previous working directory
`cd /`	Goes to the root directory (`/`)
`cd ~username`	Goes to another user's home (if permitted)
`cd ./folder`	Goes into a subfolder in the current directory

🔹 4. Special Symbols Used in Paths

Symbol	Meaning
`.`	Current directory
`..`	Parent directory (one level up)
`~`	Home directory
`-`	Last visited directory

🔹 5. Flags / Options

While `cd` has no traditional flags like `-l`, `-a`, etc., some special usage cases exist:

Usage	Effect
`cd -`	Switches to the previous directory and prints it
`cd ~`	Goes to the current user's home directory

`cd ~username`	Goes to a different user's home directory (if accessible)

✅ Note: `cd` is a shell builtin, so `man cd` may not work. Instead, use:

bash

CopyEdit

```
help cd
```

🔹 6. Checking Current Directory

Use `pwd` (print working directory):

bash

CopyEdit

```
pwd
```

It will show the absolute path of your current location.

🔹 7. Practical Examples

```
cd ~                 # Go to home

cd /var/log      # Go to system logs

cd ../           # Go one level up

cd -       # Toggle between current and last dir

cd ~/Downloads    # Go to Downloads inside home
```

🔹 8. Common Errors & Fixes

Error	Cause	Fix
`cd: No such file or directory`	Wrong path or typo	Double-check the path
`cd: Permission denied`	Trying to enter protected folder	Use `sudo` or check permissions
`cd: Not a directory`	Trying to cd into a file	Only use directories with `cd`

🔹 9. Real-Life Use Cases

- Navigating to project folders
- Switching to configuration directories (`/etc/`, `~/.config/`)
- Quickly switching between recently used folders using `cd -`
- Writing scripts to operate in certain directories

🔹 10. Summary Table

Command	Description
`cd`	Home directory
`cd /path/`	Go to specific path
`cd ..`	Go to parent directory
`cd -`	Go to previous directory
`cd ~/folder`	Folder inside home directory

3. Creating Directories

- `mkdir [directory_name]`
  Creates a new directory or folder.

- `ls`
  Lists all files and directories in the current location.

---

4. Copying Files

- `cp [source_file] [destination]`
  Copies a file to another location or directory.

- `cd [directory_name]`
  Navigate to the specified directory.

- `ls`
  Check the contents of the directory.

---

5. Moving Files

- `mv [source_file] [destination]`
  Moves a file to another location or renames it.

- `ls`
  Check the contents of the directory to verify the moved file.

Practice Tasks

1. **Use `pwd` to check your current directory.**

2. **Navigate through directories using `cd` commands. Try `cd /tmp`, `cd ..`, and `cd ~`.**

3. **Create a text file named `test.text` with the `cat > test.text` command.**

4. **Display the content of `test.text` using `cat test.text`.**

5. **Create a new directory named `Tariq` using `mkdir Tariq`.**

6. **Copy `test.text` to the `Tariq` directory using the `cp` command.**

7. **Move the file `test.text` to a new name `new` using `mv`.**

Day 03 – Managing Files & Directories in Kali Linux

In this section, we will learn how to delete files, folders, and use additional commands to manage files effectively.

---

1. Deleting Files

- `cd new`
  Navigate to the `new` directory.

- `ls`
  List the files and directories to verify the file exists.

- `rm test.text`
  Deletes the `test.text` file.
- `rm -r folder name`

  Deletes the `folder` recursively.

- `ls`
  List the files again to verify that the file is deleted.

- `cd ..`
  Go back one directory level.

- `ls`
  List the files in the parent directory.

- `cd ..`
  Go back another level.

- `ls`
  List the files again to confirm.

---

2. Deleting Directories

- `rm -r Tariq`
  Delete the `Tariq` directory and all of its contents recursively.

---

3. Additional File Management Commands

- `ls -R`
  Lists files and directories recursively, showing all subdirectories and their contents.

- `ls -l`
  Lists files in long format, showing detailed information such as file permissions, owner, size, and modification time.

- `rm -f`
  Forcefully deletes a file without asking for confirmation (use with caution).

- `rmdir [directory_name]`
  Deletes an empty directory. Only works if the directory is empty.

`find [path] -name [filename]`

Finds a file or directory by name within a specific path. Example:

```
$ find /home/tariq -name test.text
```

Practice Tasks

1. **Navigate to the `new` directory and verify its contents using the `ls` command.**

2. **Delete the `test.text` file using `rm test.text` and confirm the deletion.**

3. **Move back one directory level with `cd ..` and verify the contents.**

4. **Use the `rm -r Tariq` command to delete the `Tariq` directory and all of its contents.**

5. **List the files recursively in your current directory using `ls -R`.**

6. **List the files in long format using `ls -l`.**

Lab Lecture: User Management and File Permissions in Kali Linux

Objective:

In this lab, students will learn how to manage users and groups, and how to configure file permissions using Linux command-line tools in a Kali Linux environment.

---

1. Understanding Users in Linux

In Linux, there are two types of users:

- **Standard User: A regular user with limited system privileges.**
- **Root User (Administrator / Super User): A user with complete system access, able to perform any operation.**

To switch to root user mode, we use the command:

```
sudo su
```

(Super User Do - grants root privileges)

---

2. User Management

Creating a New User (Standard User Mode)

To create a new user:

```
sudo useradd test
```

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

```
sudo passwd test
```

This command adds a user named `test` to the system.

Checking User List

To view all system users:

```
sudo cat /etc/passwd
```

Setting a Password for a User

To set or change the password of a user:

```
sudo passwd test
```

---

3. Group Management

Creating a New Group

To create a new group:

```
sudo groupadd boys
```

Adding a User to a Group

To add an existing user to a group:

```
sudo usermod -a -G boys test
```

Deleting a User

To delete a user account:

```
sudo userdel test
```

Deleting a Group

To delete a group:

```
sudo groupdel boys
```

---

4. Understanding File Permissions

Linux assigns permissions to files and directories to control who can access and modify them.
There are three types of permissions:

Permission	Symbol	Description
Read	r	View or read file contents
Write	w	Modify or edit file contents
Execute	x	Run a file (for scripts, etc.)

Permissions are assigned separately to:

- **Owner (User)**
- **Group**
- **Others**

Each permission type is represented numerically:

- **Read = 4**
- **Write = 2**

- **Execute = 1**

The sum of these numbers defines the access level.

---

5. File Permission Table

Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read + Write	rw-
7	Read + Write + Execute	rwx

---

6. Changing File Permissions

To change the permissions of a file, we use the `chmod` command.
Example:

```
sudo chmod 764 test.txt
```

Here:

- **7 (Owner) → Read (4) + Write (2) + Execute (1) = 7**
- **6 (Group) → Read (4) + Write (2) = 6**
- **4 (Others) → Read (4) = 4**

Thus, the file `test.txt` will have:

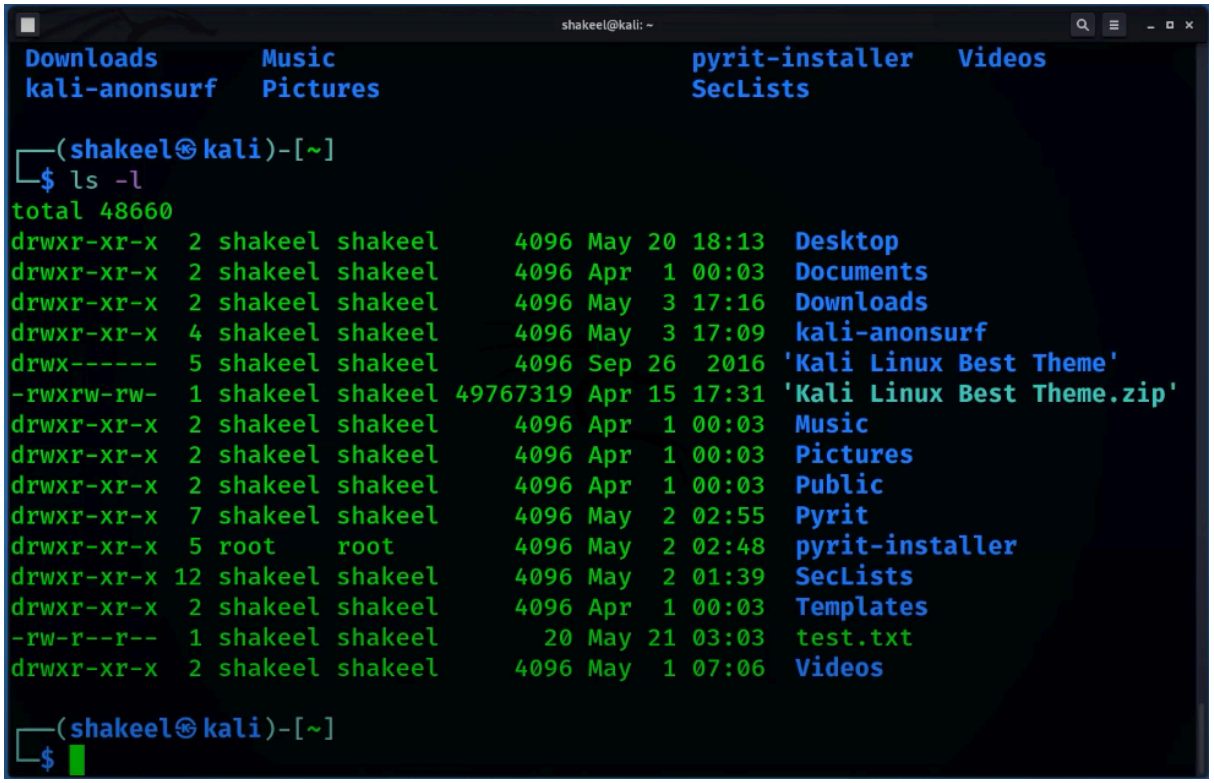
- **Owner: Full access (rwx)**
- **Group: Read and write access (rw-)**
- **Others: Read-only access (r--)**

Conclusion:

This lab covers basic user and group management, along with file permission handling in Kali Linux.
Mastering these fundamentals ensures secure system administration and helps maintain the integrity of the Linux operating environment.

File Permission Mode:

1. Read
2. Write
3. Execute



Number	Permission Type	Symbol
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx

File Permissions and `chmod` in Linux

In Linux, every file and directory has permissions that control who can read, write, or execute them. These permissions are divided into three groups:

1. **User (u) → the owner of the file**
2. **Group (g) → members of the file's group**
3. **Others (o) → everyone else**

🔹 **Understanding Permissions**

- **Read (r): allows viewing the file contents or listing a directory.**
- **Write (w): allows modifying a file or creating/deleting items inside a directory.**
- **Execute (x): allows running a file as a program or entering a directory.**

🔹 **The `chmod` Command**

`chmod` (change mode) is used to modify permissions.

There are two main ways to set permissions:

1. Numeric (Octal) Method

Permissions are represented by numbers:

- **`4 = read`, `2 = write`, `1 = execute`**
- **Add them up for each category.**

Example:

```
chmod 764 test.txt
```

- `7 (rwx)` → user (owner)
- `6 (rw-)` → group
- `4 (r--)` → others

---

2. Symbolic Method

Instead of numbers, we use letters:

- `u` = user, `g` = group, `o` = others, `a` = all

Example:

```
chmod u=rwx,g=rw,o=r test.txt
```

This is the same as `764`.

You can also add or remove specific permissions:

```
chmod g+w test.txt   # give write permission to group
chmod o-r test.txt   # remove read permission from others
```

🔹 **Why `sudo` is Sometimes Needed**

If the file belongs to you, you can change its permissions directly.
If the file belongs to another user or is in a system directory, you need admin privileges:

```
sudo chmod 764 test.txt
```

Alternatively, you can change the file's ownership:

```
sudo chown yourusername test.txt
chmod 764 test.txt
```

✅ **Summary**

1. **`chmod` changes permissions.**
2. **Use numeric (764) or symbolic (u=rwx,g=rw,o=r) methods.**
3. **`sudo` is only needed if you don't own the file.**

4. **Sudu chmod 764 test.txt**

Finale Exam Topic:

Page Number: 59 - 82, 91-112

Using grep, pipe, sorting & tee of Kali Linux

Lecture: Using `grep`, Pipe (`|`), and `sort` in Linux (Kali Linux)

---

1. Introduction

Linux provides powerful command-line tools for text processing and automation.
Three essential tools are:

- **Pipe (`|`) → pass output of one command as input to another.**
- **`grep` → search for text or patterns.**
- **`sort` → arrange lines of text in a specific order.**

These tools are widely used in system administration, programming, and cybersecurity.

---

2. Definitions

Pipe (`|`)

- **A pipe is a symbol `|` used to connect multiple commands, allowing the output of one command to become the input of the next.**

grep

- **`grep` stands for Global Regular Expression Print.**
- **It is used to search for specific words or patterns in text files or command outputs.**

sort

- **`sort` arranges lines of text in alphabetical or numerical order.**
- **It can also sort in reverse order using the `-r` option.**

---

3. Commands and Examples

Creating and Viewing a File

```
cat > test.txt
Tariqul
Hasan
Rahim
Tariqul Islam
Shawon
```

(Press `CTRL+D` to save)

View file contents:

```
cat test.txt
```

Pipe (|) Example

```
cat test.txt | grep Tariqul
```

- Output of `cat test.txt` is passed to `grep`.
- Only lines containing `"Tariqul"` are displayed.

---

grep Example

```
grep "Islam" test.txt
```

- Directly searches for `"Islam"` in `test.txt`.

---

sort Example

```
sort test.txt        # ascending order
sort -r test.txt     # descending order
```

Combining Commands

```
cat test.txt | grep Tariqul | sort -r
```

- **Show contents → filter `"Tariqul"` → sort in reverse order**

Appending Output

```
echo "New Entry" >> test.txt
cat test.txt
```

- **Adds `"New Entry"` at the end without overwriting existing content.**

Advanced Example

```
cat test.txt | grep Tariqul | sort -r | tee output.txt
```

- **Filter `"Tariqul"` → reverse sort → display AND save to `output.txt`**

5. Advantages

Command	Advantages
**Pipe (`	`)**
grep	Fast, precise search; supports regular expressions; can filter command outputs

sort	**Organizes data alphabetically/numerically; works with large files; combinable with grep/pipe**

---

6. Disadvantages

Command	Disadvantages
**Pipe (`	`)**
grep	**Requires exact patterns unless using regex; complex patterns may confuse beginners**
sort	**Sorts entire lines by default; may need options for numerical or case-insensitive sorting**

---

7. Summary

- **Pipe (`|`) → pass output between commands.**
- **grep → search for words or patterns.**
- **sort → arrange text lines.**
- **These commands can be combined to filter, sort, and process text efficiently in Linux/Kali Linux.**
- **Proper use of permissions (`chmod`) may be necessary to modify files.**

Set Up Users, Group & Permissions of Kali Linux

User: tariq@kali-[~]

Two Types User:

1. **Standard User**
2. **Root User or Admin User or Super User**

- **sudo su (Super User Do) – For Root Login**

1. Using `su` command

If you know the username of the normal user (for example, `tariq`), run:

```
su - tariq
```

- **`su` means "switch user".**
- **`-` ensures you also load the target user's environment (like PATH, HOME, etc.).**
- **You'll be asked for that user's password.**

To confirm you switched successfully:

```
whoami
```

Output will show:

```
tariq
```

New User Create: (In stander User Mude)

- **sudo useradd test**

Check User:

- **Sudo cat /etc/passwd**

User Account Password Set:

- **sudu passwd test**

New Group Create: (In stander User Mude)

- **sudo groupadd boys**

Add User to Group

- **sudo usermod -a -G boys test**

Delete User

- **sudo userdel test**

Delete Group

- **Sudu groupdel boys**
-

2. User Accounts in Linux

Each user in Linux has:

- **A username (unique identifier)**
- **A User ID (UID)**

- **A primary group**
- **A home directory**
- **A default shell**

User account details are stored in:

```
/etc/passwd
```

View All Users:

```
cat /etc/passwd
```

Show Only Usernames:

```
cut -d: -f1 /etc/passwd
```

Show Sorted List of Users:

```
cut -d: -f1 /etc/passwd | sort
```

---

3. Group Accounts in Linux

Groups allow multiple users to share access to files and resources.
Each group has a Group Name and a Group ID (GID).

Group information is stored in:

```
/etc/group
```

View All Groups:

```
cat /etc/group
```

Show Only Group Names:

```
cut -d: -f1 /etc/group
```

---

4. Viewing User and Group Information

Check User Details:

```
id username
```

- **Shows UID, GID, and groups of the user.**

Show Groups of Current User:

```
groups
```

Show Groups of a Specific User:

```
groups username
```

---

5. Currently Logged-in Users

List Active Users:

```
who
```

Detailed Information of Logged-in Users:

```
w
```

Show Current User:

```
whoami
```

---

6. Adding and Managing Users

Add a New User:

```
sudo adduser username
```

Delete a User:

```
sudo deluser username
```

Change a User Password:

```
sudo passwd username
```

---

7. Adding and Managing Groups

Add a New Group:

```
sudo groupadd groupname
```

Rename a group

```
sudo groupmod -n newgroup oldgroup
```

Delete a Group:

```
sudo groupdel groupname
```

Add User to Group:

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

---

8. System vs. Normal Users

- **System users: Created automatically for system processes (e.g., `root`, `daemon`, `mail`)**
- **Normal users: Created for human users (UID usually starts from 1000)**

Show Only Normal Users:

```
awk -F: '$3 >= 1000 {print $1}' /etc/passwd
```

---

9. Summary Table

Purpose	Command
List all users	`cat /etc/passwd`
List all groups	`cat /etc/group`
Check current user	`whoami`
Show logged-in users	`who` or `w`
Show user's groups	`id username`
Add new user	`sudo adduser username`
Add new group	`sudo groupadd groupname`

---

10. Conclusion

Linux provides flexible and secure ways to manage users and groups. By understanding these commands, administrators can control access, assign permissions, and maintain system security effectively.

Chapter: Networking Troubleshooting Commands in Kali Linux

1. Introduction

Networking troubleshooting is a critical skill for system administrators, cybersecurity professionals, and ethical hackers. In Kali Linux, several built-in commands and tools help diagnose, analyze, and fix network-related issues.

This chapter provides a complete overview of the most useful networking troubleshooting commands, their syntax, and practical examples.

---

2. Basic Network Configuration Commands

2.1 `ifconfig` (Interface Configuration)

- **Purpose: Display or configure network interfaces.**

Syntax:

```
ifconfig
```

```
ifconfig eth0 up
ifconfig eth0 192.168.1.10 netmask 255.255.255.0
```

-

Example Output:

```
tariqul@Latitude:~$ ifconfig
enp0s31f6: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether d4:81:d7:5a:de:a5  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xe1200000-e1220000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 451  bytes 48455 (48.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 451  bytes 48455 (48.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.101  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::e50b:5789:d764:a004  prefixlen 64  scopeid 0x20<link>
        ether 34:f3:9a:c6:52:00  txqueuelen 1000  (Ethernet)
        RX packets 15826  bytes 17540168 (17.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8771  bytes 2330810 (2.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Note: In newer Kali versions, `ifconfig` is replaced by `ip` command.

Overview

Overview

Each block represents a network interface on your system.
Here you have:

- `enp0s31f6` → your wired Ethernet port (not connected)
- `lo` → the loopback (internal) interface
- `wlp1s0` → your Wi-Fi adapter (connected)

1️⃣ `enp0s31f6` — Wired Ethernet Interface

```
enp0s31f6: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
```

🔹 `flags`

- **UP → The interface is enabled (not administratively down)**
- **BROADCAST → Supports sending broadcast packets**
- **MULTICAST → Supports multicast traffic (used in modern networking)**
- ***(Not RUNNING → means no cable plugged in / no link)***

🔹 `mtu 1500`

- **MTU (Maximum Transmission Unit) = largest packet size (in bytes) the interface can send — 1500 is standard for Ethernet.**

```
ether d4:81:d7:5a:de:a5  txqueuelen 1000  (Ethernet)
```

- **ether → The hardware (MAC) address: `d4:81:d7:5a:de:a5`**
- **txqueuelen 1000 → Size of the transmit queue (number of packets that can wait to be sent)**

```
RX packets 0  bytes 0 (0.0 B)
TX packets 0  bytes 0 (0.0 B)
```

- **RX (Received) — how many packets and bytes have been received.**
- **TX (Transmitted) — how many packets and bytes have been sent.**
- **Zero → interface is up but not in use (no cable or network link).**

```
RX errors 0  dropped 0  overruns 0  frame 0
TX errors 0  dropped 0 overruns 0  carrier 0
collisions 0
```

These are error counters:

Field	Meaning
errors	Total errors in packets
dropped	Packets dropped due to buffer or driver issues

overruns	**Data overflowed buffers**
frame	**Frame alignment errors**
carrier	**Physical layer problems (like cable issue)**
collisions	**Ethernet collisions (on old shared networks)**

```
device interrupt 16  memory 0xe1200000-e1220000
```

- **interrupt → Hardware interrupt line number**
- **memory → Memory-mapped I/O range for the device (used by driver)**

2️⃣ `lo` — Loopback Interface

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
```

- **LOOPBACK → Used internally by your system (localhost)**
- **RUNNING → Interface is active**
- **MTU 65536 → Can send large packets internally**

```
inet 127.0.0.1  netmask 255.0.0.0
inet6 ::1
```

- **IPv4 loopback address = 127.0.0.1**
- **IPv6 loopback address = ::1**

This interface is used for communication within your own computer (e.g., when you ping `localhost`).

3️⃣ `wlp1s0` — Wireless Wi-Fi Interface

```
wlp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
mtu 1500
```

- **UP → Enabled**
- **RUNNING → Actively connected**
- **BROADCAST, MULTICAST → Supported features**

```
inet 192.168.1.101  netmask 255.255.255.0  broadcast
192.168.1.255
```

- **inet → IPv4 address (your Wi-Fi IP)**
  - **IP = `192.168.1.101` (local network address)**
  - **netmask → `255.255.255.0` (subnet)**

- broadcast → `192.168.1.255` (used to message all devices on the network)

```
inet6 fe80::e50b:5789:d764:a004
```

- IPv6 link-local address — used for communication inside your local Wi-Fi network.

```
ether 34:f3:9a:c6:52:00  txqueuelen 1000
```

- MAC address of the Wi-Fi adapter.

```
RX packets 15826  bytes 17540168 (17.5 MB)
TX packets 8771  bytes 2330810 (2.3 MB)
```

- Packets and bytes received (RX) and sent (TX).

```
RX/TX errors ... collisions 0
```

- Show network health — all zero means your connection is clean and stable.

Summary Table

Term	Meaning
`flags`	Interface status and capabilities
`mtu`	Max Transmission Unit size
`ether`	MAC (hardware) address
`inet`	IPv4 address
`inet6`	IPv6 address
`RX` / `TX` packets	Packets received/sent
`errors`, `dropped`, etc.	Network error stats
`txqueuelen`	Length of transmit queue
`lo`	Loopback (internal network)
`device interrupt`	Hardware interrupt number

2.2 `ip` Command (Modern replacement of ifconfig)

- **Purpose: Show and manage IP addresses, routes, and interfaces.**

Common Uses:

```
ip addr show
ip link show
ip route show
ip addr add 192.168.1.10/24 dev eth0
ip link set eth0 up
```

-

Example:

```
ip a
```

-

---

3. Connectivity Testing Commands

3.1 `ping`

- **Purpose: Test connectivity between your system and another host.**

Syntax:

```
ping <IP or domain>
```

-

Example:

```
ping google.com
```

```
tariqul@Latitude:~$ ping google.com
PING google.com (142.251.220.110) 56(84) bytes of data.
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=1 ttl=117 time=30.2 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=2 ttl=117 time=30.4 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=3 ttl=117 time=30.0 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=4 ttl=117 time=31.4 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=5 ttl=117 time=30.9 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=6 ttl=117 time=33.6 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=7 ttl=117 time=29.3 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=8 ttl=117 time=33.2 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=9 ttl=117 time=31.8 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=10 ttl=117 time=30.6 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=11 ttl=117 time=28.0 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=12 ttl=117 time=29.4 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=13 ttl=117 time=31.3 ms
64 bytes from pnmaaa-be-in-f14.1e100.net (142.251.220.110): icmp_seq=14 ttl=117 time=40.7 ms
```

- **Key Points:**
  - **Verifies if the destination host is reachable.**
  - **Measures packet loss and round-trip time (RTT).**

Breakdown

Part	Meaning
64 bytes	**Size of the reply packet (the ICMP echo reply). Usually 64 bytes by default.**
from pnmaaa-be-in-f14.1e100.net	**The hostname of the server that replied (in this case, a Google server — all `*.1e100.net` are Google-owned).**

(142.251.220.110)	**The IP address of that host.**
icmp_seq=11	**The sequence number of this ping packet — it was the 11th one sent. Helps track lost packets.**
ttl=117	**Time To Live — a counter that decreases each time the packet passes through a router. Shows how many hops remain before it's dropped. (117 is normal for internet hosts.)**
time=28.0 ms	**Round-trip time — how long it took for your packet to go to that host and back, measured in milliseconds. Lower = faster connection.**

- ○

3.2 `traceroute`

- **Purpose: Trace the route packets take to reach a destination.**

Syntax:

```
traceroute <domain or IP>
```

-

Example:

```
traceroute www.google.com
```

```
tariqul@Latitude:~$ traceroute google.com
traceroute to google.com (142.250.183.174), 64 hops max
  1   192.168.1.1  1583.789ms  0.897ms  0.790ms
  2   10.213.155.1  12.834ms  3.247ms  1.069ms
  3   10.101.100.65  7.635ms  8.297ms  38.742ms
  4   10.77.252.25  516.203ms  17.852ms  6.744ms
  5   10.160.60.21  13.985ms  25.426ms  35.122ms
  6   103.75.239.6  110.092ms  60.261ms  48.400ms
  7   *  *  *
  8   142.251.55.70  69.297ms  67.586ms  78.890ms
  9   142.251.230.52  67.726ms  69.242ms  75.174ms
 10   142.250.183.174  49.405ms  51.236ms  47.973ms
```

- **Use Case: Helps identify where network delays or failures occur.**

3.3 `tracepath`

- **Purpose: Similar to `traceroute`, but doesn't require superuser permissions.**

Example:

```
tracepath www.google.com
```

-

---

4. Name Resolution and DNS Testing

4.1 `nslookup`

- **Purpose: Query DNS records (A, MX, CNAME, etc.).**

Example:

`nslookup www.google.com`

```
tariqul@Latitude:~$ nslookup www.google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.251.221.196
Name:   www.google.com
Address: 2404:6800:4007:831::2004
```

Closed for final

4.2 `dig` (Domain Information Groper)

- **Purpose: Advanced DNS query tool.**

Syntax:

```
dig <domain>
dig <domain> ANY
dig @8.8.8.8 www.google.com
```

-

Example Output:

```
;; ANSWER SECTION:
google.com.  21599 IN A 142.250.183.206
```

-

---

5. Route and Gateway Checking

5.1 `route`

- **Purpose: Display or manipulate the IP routing table.**

Example:

```
route -n
route add default gw 192.168.1.1
```

-

5.2 `ip route`

Modern Alternative:

```
ip route show
ip route add default via 192.168.1.1
```

-

---

6. Port and Service Troubleshooting

6.1 `netstat`

- **Purpose: Display active connections, listening ports, and routing tables.**

Examples:

```
netstat -tuln      # Show TCP/UDP listening ports
netstat -anp       # Show all connections with process names
```

-
- **Modern Replacement: `ss`**

6.2 `ss`

Summary:

Examples:

```
ss -tuln
ss -s
```

- **`ss -tuln` → Shows all listening TCP/UDP ports on your system without resolving names. Useful to see which services are waiting for connections.**

- **`ss -s` → Shows a summary of all sockets, including total TCP, UDP, and RAW connections, along with states like established or time-wait.**

  Purpose: Modern, faster alternative to `netstat`.

---

7. Network Packet Analysis

7.1 `tcpdump`

- **Purpose: Capture and analyze network packets in real time.**

Examples:

```
tcpdump -i eth0
tcpdump -i wlan0 port 80
tcpdump -n host 192.168.1.10
```

-
- **Usage: Great for detecting suspicious traffic or debugging network apps.**

---

8. Host and ARP Commands

8.1 `hostname`

Purpose: Display or change the system's hostname.

```
hostname
```

```
hostname new-host
```

-

8.2 `arp`

Purpose: View and modify the ARP (Address Resolution Protocol) table.

```
arp -a
```

```
arp -d <IP>
```

-

---

9. Advanced Diagnostic Tools

9.1 `ethtool`

Purpose: Display and change Ethernet device settings.

```
ethtool eth0
```

-

9.2 `mtr` (My Traceroute)

Purpose: Combines `ping` and `traceroute` in real time.

```
mtr google.com
```

-

9.3 `nmap`

Purpose: Network exploration and security auditing.

```
nmap -sP 192.168.1.0/24     # Ping scan
nmap -p 80,443 192.168.1.5  # Scan specific ports
```

-

---

10. Common Troubleshooting Scenarios

Problem	Possible Command(s)	Purpose
No internet connection	`ping`, `ifconfig`, `ip route`	Check interface and gateway
DNS not resolving	`nslookup`, `dig`, `cat /etc/resolv.conf`	Verify DNS configuration
Slow network	`traceroute`, `mtr`, `ping`	Identify network delay
Unknown open ports	`netstat`, `ss`, `nmap`	Check active connections
Packet loss	`ping`, `tcpdump`	Detect network instability

---

11. Summary

Kali Linux provides a comprehensive set of commands for network troubleshooting. From basic connectivity (`ping`, `ifconfig`) to advanced

packet analysis (`tcpdump`, `nmap`), mastering these tools helps in quickly diagnosing and resolving network issues.

---

12. Practice Exercises

1. **Use `ping` to test connectivity with your router.**
2. **Display your IP address using both `ifconfig` and `ip` commands.**
3. **Find out which DNS server your system is using.**
4. **Use `traceroute` to find the path packets take to `google.com`.**
5. **Capture all HTTP packets using `tcpdump`.**
6. **Identify open ports on your system using `ss` or `nmap`.**

Managing Processes &Task of Kali Linux

1. **Processes: Series of action or steps to archive a perticular end or output.**
2. **ping (Command)**
3. **Poss a Command:** ctrl shift z (Poss a command)
4. **fg ping (Continuer a process)**
5. **ctrl c** (close a process fully)
6. **top** (Command) - Current Running Process
7. **ctrl c** (close a process fully)
8. **sudo kill** (Process ID) – Stop a process

9. **ps us** (Current User process)

10. **How To Use SSH, FTP, TELNET on Kali Linux - Kali Linux**
11. **Ssh -h (Details)**
12. **Ssh ursername@192.168.0.0**
13. **Sudu rm -r folder name (to delete a folder)**
14. **ftp -h (Details)**
15. **ftp ip-192.168.0.0**
16. **ftp ? (for all option)**
17. **or ftp help (help Commant)**
18. **telnet (Plaintext)**
19. **telnet -h (Details)**
20. **telnet ip-192.168.0.0**

Set IP Address & Internet in Kali Linux

1. **ifconfig (Connected Device Details)**
2. **sudo ifconfig eth0 down (Down a port)**
3. **ifconfig (for check)**
4. **sudo ifconfig eth0 up (up a port)**
5. **sudo ifconfig wlan 192.160.0.104 (Manual IP Assign)**
6. **sudo ifconfig wlan netmask 255.255.255.0 (Subnet Musk Assign)**
7. **sudo ifconfig wlan broadcast 192.160.0.104 (Broadcast IP Assign)**
8. **ping command**

Enumerate Users & System Info of Kali Linux

1. **whoami (Linux User Name)**
2. **id (User ID, Group ID Details)**
3. **hostname (For Host Name)**
4. **sudo nano /etc/hostname (Host Name Change)**

5. **uname -a (Linux Operating Current Version)**
6. **hostnamectl (Kali Linux Details Information)**

How To Install Software & Learn Kali Linux Tools

1. **sudo apt install vlc (Software Install Command)**
2. **sudo apt search vlc (Available Tool Names)**
3. **nmap –help (Available Option of this/a tool)**
4. **man nmap (Manual of a tool)**
5. **q (Logout from details)**
6. **man ping (Manual of a tool)**
7. **whatis nmap (Summery of a tool)**

Introduction to Bash Shell in UNIX/Linux

Lecture Title:

Shell Scripting in Unix/Linux – Practical Approach Using Kali Linux

Understanding Shells

A shell is a text-based interface that lets you talk to your computer.

There are different types of shells, but Bash (Bourne Again SHell) is the most popular because it's powerful and easy to use.

Types of Shells:

- **Bourne Shell (sh): The original Unix shell, developed by Stephen Bourne.**
- **C Shell (csh): Known for its C-like syntax, popular for interactive use.**
- **Korn Shell (ksh): Combines features of sh and csh, offering advanced scripting capabilities.**
- **Bash (Bourne Again SHell): An improved version of sh, with additional features like command history and tab completion.**

Why Use Bash?

- **It is widely available on Unix/Linux systems, making scripts portable.**
- **Supports powerful scripting features, including loops, conditionals, and functions.**
- **Provides command history and tab completion for ease of use.**
- **Can be integrated with other Unix/Linux tools for automation.**
-

What is Bash?

Bash is used to run commands and write scripts on Unix/Linux systems.

Bash is the default shell on most Linux and macOS systems.

Why Learn Bash?

Bash is a powerful tool for developers and system administrators.

Understanding and mastering Bash is important for working in Unix/Linux.

Shell vs. Bash Shell

A "Shell" is any command-line tool. "Bash shell" is specifically the Bourne Again SHell.

Because of the popularity, Bash is often what people think of when they say shell

This tutorial will teach you how to use Bash.

History of Bash

Bash was developed in 1989 by Brian Fox.

Bash was created as a free replacement for the Bourne shell. It quickly became the default for GNU and many Linux systems.

Over time, Bash added features from other shells, such as the Korn shell (ksh) and C shell (csh), and it became a very versatile tool for command-line users.

Practical Uses of Bash

System administrators use Bash to:

- **Automate tasks**
- **System operations**
- **Process data**

Developers use Bash for:

- **Build automation**
- **Testing**
- **Deployment**
- **Data processing and manipulation**
-

Setting Up Bash

Most Unix/Linux systems come with Bash pre-installed.

To check if Bash is installed, open a terminal and type:

```
bash --version
```

If Bash isn't installed, you can install it using your system's package manager.

For example, on Ubuntu/Debian, type:

```
sudo apt-get install bash
```

On macOS, you can install Bash via Homebrew:

```
brew install bash
```

On Windows, you can use WSL (Windows Subsystem for Linux) to run Linux or just use Git Bash for Windows.

Common commands:

- `ls` - List directory contents
- `cd` - Change the current directory
- `pwd` - Print the current working directory
- `echo` - Display a line of text
- `cat` - Concatenate and display files
- `cp` - Copy files and directories
- `mv` - Move or rename files
- `rm` - Delete files or folders
- `touch` - Create an empty file or update its time
- `mkdir` - Create a new folder
-

Bash Syntax for Scripting

Bash scripts are sequences of commands executed by the Bash shell. They automate tasks and can be used to perform complex operations. Understanding Bash syntax is crucial for writing effective scripts.

Basic Syntax

Here are some basic rules for using Bash in scripts:

- **Comments: Comments start with a `#` and Bash ignores them.**
- **Command Order: Commands run one after the other, from top to bottom.**
- **Semicolons: Use `;` to run multiple commands on the same line.**

Let's go through them one by one with examples.

Comments

Comments start with a `#` and Bash ignores them. They explain what your code does, making it easier to understand.

Example: Using Comments

```
# This script prints a greeting message

echo "Hello, World!"
```

Command Execution Order

Commands are run (or executed) in sequence from top to bottom

This order affects the logic and functionality of the script.

Example: Command Execution Order

```
  echo "First command"

echo "Second command"
```

Semicolons

Semicolons `;` can be used to separate multiple commands on the same line, which is useful for writing concise scripts.

Example: Using Semicolons

```
echo "This is a test"; echo "This is another test"
```

Best Practices for Writing Scripts

Here are some tips for writing clean and efficient scripts:

- **Use comments to explain your code.**
- **Choose meaningful variable names.**
- **Test your scripts thoroughly before using them in production.**

Introduction to Bash Scripting

Bash scripts are files containing commands that you run in the terminal. They automate tasks and make your work more efficient.

Creating a Bash Script

To create a script, start with the shebang `#!` followed by the path to Bash, usually `/bin/bash`. Make sure your script has execute permissions.

Example: Simple Bash Script

```
#!/bin/bash

# This script prints a greeting message

echo "Hello, World!"
```

Using Variables in Scripts

Variables store data that your script can use. Assign values using the `=` sign without spaces.

Example: Using Variables

```
#!/bin/bash

# Assign a value to a variable

name="World"

echo "Hello, $name!"
```

Lab Section: Step-by-Step Instructions

Step 1: Open the Terminal and go to the Desktop

Type the following command:

```
cd Desktop
```

Step 2: Create a New Shell Script File

To create a new file using Nano editor, type:

```
nano new.sh
```

This command opens the Nano text editor and creates a file named `new.sh`.

Step 3: Write a Simple Script

Inside Nano, type the following lines:

```
#!/bin/sh
echo "Hello World"
echo "This is my first shell script!"
```

Step 4: Save and Exit Nano

- **Press `Ctrl + O` to save**
- **Press `Enter` to confirm the file name**
- **Press `Ctrl + X` to exit the editor**

Step 5: Check the File in Desktop

Type:

```
ls
```

If the file name appears in white color, it means it does not have execution permission.

Step 6: Check File Permissions
Type:
`ls -la`
This will display the permission status of all files. You may see that your script has `-rw-r--r--` meaning it is readable and writable, but not executable.

Step 7: Make the Script Executable
To give execution permission, type:
`sudo chmod +x new.sh`
It will ask for the root password. Type your password and press Enter.

Step 8: Verify Execution Permission
Now type:
`ls -la`
You should now see `-rwxr-xr-x` for your script file, which means it has execute permission.

Step 9: Run the Script
You can run the script using either of the following two methods:

1. `bash new.sh`

2. `./new.sh`

Note: The second method (`./new.sh`) will only work if the shebang line `#!/bin/sh` is present at the top of the script.

Summary / Key Points:

- **Shell scripts are used to automate command-line tasks.**
- **Always begin scripts with a shebang line to specify the interpreter.**
- **Use Nano or other text editors to create `.sh` files.**
- **Use `chmod +x filename.sh` to make a script executable.**
- **Scripts can be run with `bash filename.sh` or `./filename.sh`.**

Homework / Practice Task:

Create a shell script named `info.sh` that does the following:

1. **Prints your full name**
2. **Prints the current date and time**
3. **Lists all files and directories in the current location**

Save and execute the script following the steps learned in this session.

Shell Script Banner Design

Lab Lesson: `echo` Command, Variable Declaration & Banner Design in Linux

---

🎯 Objective:
By the end of this lesson, students will be able to:

- **Use the `echo` command to print text**
- **Declare and print string and integer variables**
- **Design a basic text banner using shell scripting**

🛠️ 1. Using the `echo` Command

The `echo` command is used to display text in the terminal.

bash

CopyEdit

```
echo hello world
```

```
echo "hello world"
```

```
echo 'hello world'
```

Note: All three commands work the same. Using double quotes `"` or single quotes `'` helps preserve spaces and special characters.

🛠️ 2. Printing Variables

String Variable:

bash

CopyEdit

```
name="Nujao Mia"
echo "My name is $name"
```

Integer Variable:

bash

CopyEdit

```
age=25
echo "I am $age years old"
```

Tip: There should be no space around the `=` sign when declaring variables.

🛠️ 3. Designing a Sample Banner

Use `echo` with symbols to make a simple banner.

Example 1 (Single Line Banner):

bash

CopyEdit

```
banner_text="Welcome to Linux Lab"

echo "============================="
echo "     $banner_text     "
echo "============================="
```

Example 2 (Multiline Banner):

markdown

CopyEdit

```
echo "
*********************************
*                               *
*     Welcome to Linux Lab      *
*                               *
*********************************
"
```

Optional Task: Let students design their own name banners using variables.

✅ **Practice Tasks for Students:**

1. **Write a shell script that prints:**
    - **Your full name**
    - **Your age**
    - **Your department name**
2. **Add a banner at the top of the output**
3. **Save the file as `intro.sh`**

```
echo "enter your name"
read
echo  "YOur Name IS" $REPLY
```

echo "enter your name"

read name

echo "YOur Name IS" $name

read - p "echo enter your name" fullname

echo "YOur Name IS" $fullname

Lab: User Input & Basic Scripting in Bash

Objective:
Learn to use the `echo` and `read` commands in Linux shell scripting.
Understand different ways to take input and display output using variables.
Design basic banners and menus in scripts.

🛠️ 1. Using the `echo` Command

The `echo` command is used to display text.

bash

CopyEdit

```
echo hello world
```

```
echo "hello world"
```

```
echo 'hello world'
```

Note: Quotes help preserve spacing and special characters.

🛠️ 2. Printing Variables

String Variable:

bash

CopyEdit

```
name="Nujao Mia"
echo "My name is $name"
```

Integer Variable:

bash

CopyEdit

```
age=25
echo "I am $age years old"
```

🛠️ 3. User Input with `read`

Using default `$REPLY`:

bash

CopyEdit

```
echo "Enter your name"
```

```
read
echo "Your Name Is $REPLY"
```

Using a custom variable:

bash

CopyEdit

```
echo "Enter your name"
read name
echo "Your Name Is $name"
```

Using `-p` to prompt inline:

bash

CopyEdit

```
read -p "Enter your name: " fullname
echo "Your Name Is $fullname"
```

🛠️ 4. Designing a Banner

Single-line banner using a variable:

bash

CopyEdit

```
banner_text="Welcome to Linux Lab"


echo "============================="
echo "     $banner_text     "
echo "============================="
```

Multiline banner:

markdown

CopyEdit

```
echo "
*********************************
*                               *
*     Welcome to Linux Lab      *
*                               *
*********************************
"
```

✅ Practice Task

Create a script called `userinfo.sh` that:

1. **Displays a custom banner**
2. **Asks for name, age, and favorite subject**
3. **Displays the result using `echo`**

Run your script using:

nginx

CopyEdit

```
bash userinfo.sh
```

🖼️ **Final Example – Code from Image (Menu-Based Script)**

bash

CopyEdit

```
#!/bin/bash

echo "########################################"
echo "#                                      #"
echo "#         MY FIRST SCRIPT              #"
echo "#                                      #"
echo "########################################"
```

```
echo "Enter 1 To Show IP"

echo "Enter 2 To Show MAC"


read -p "Select Your Option: " option

echo "You Select Option $option"
```

“Write a Bash Script Using if-else to Display Options for Viewing IP and MAC Address”

```
 GNU nano 7.2                ifelse.sh
#!/bin/bash

echo "Select an option:"
echo "Press 1 to Show IP address"
echo "Press 2 for Show MAC address"
read -p "Enter your choice: " choice




if [ "$choice" == 1 ];
then
   echo "Your IP address is:" $(hostname -I)
elif [ "$choice" == 2 ];
```

```
then
   echo "Your MAC address is:"

   ip link show | grep "link/ether" | awk '{print $2}'
else
   echo "Invalid option. Please press 1 or 2."
fi
```

Meaning:

ip link show | grep "link/ether" | awk '{print $2}'

1. ip link show

Shows all network interfaces on the system.

Displays details like interface names, MAC addresses, MTU, state (UP/DOWN), etc.

Example output snippet:

bash

Copy code

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> …

link/ether 12:34:56:78:9a:bc brd ff:ff:ff:ff:ff:ff

2. | (pipe)

Sends the output of the previous command to the next command.

3. grep "link/ether"

Filters only the lines containing the word link/ether.

These lines are where the MAC addresses appear.

Example matched line:

bash
Copy code
link/ether 12:34:56:78:9a:bc brd ff:ff:ff:ff:ff:ff
4. awk '{print $2}'
awk splits each line into fields separated by spaces.

$2 means "print the second field".

From this line:

bash
Copy code
link/ether 12:34:56:78:9a:bc brd ff:ff:ff:ff:ff:ff
Field breakdown:

$1 = link/ether

$2 = 12:34:56:78:9a:bc ← the MAC address

$3 = brd

$4 = ff:ff:ff:ff:ff:ff

So awk '{print $2}' prints only the MAC address.