Introduction to Linux

Lazhar

November 29, 2025

Course Outline

Overview

What is LInux?

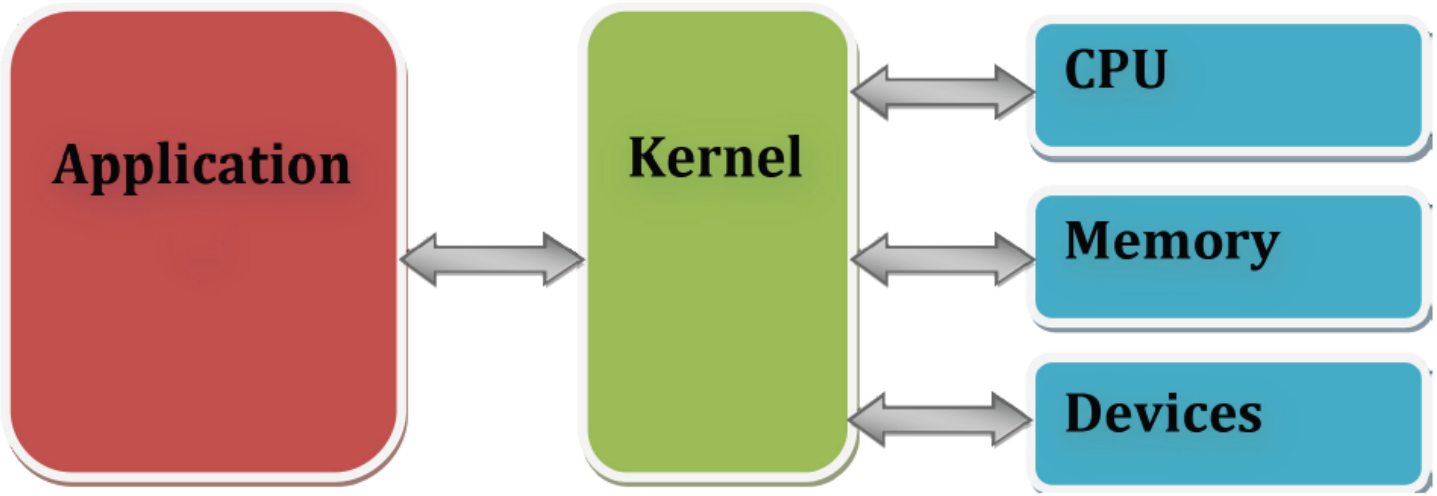
Linux is an operating system, much like Microsoft Windows or Apple Mac OS. Unlike other mainstream Operating Systems, Linux is made freely available and is Open Source.

Why Linux?

- Free and open-source
- Secure and customizable
- Ideal for development and education

A brief History of Linux

Back in August of 1991, a student from Finland began a post to the comp.os.minix newsgroup with the words:
Hello everybody out there using minix - I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones.
The student was Linus Torvalds, and the "hobby" he spoke of eventually became what we know today as Linux.
A full-featured UNIX-like operating system, Linux has been developed not just by Linus, but by hundreds of programmers around the world.



Different Linux Desktop Interfaces

One of the strengths of Linux is that its desktop interface can be fully customized to match your workflow and preferences. Unlike fixed operating systems, Linux allows you to choose from different desktop environments, each offering unique features and aesthetics.

- GNOME and KDE are the two main desktop interfaces in Linux
- They are called Window Managers — they define the look and feel of your desktop
- Just like choosing a Linux distribution, your desktop interface depends on personal preference and intended use

Different Linux Desktop Interfaces

GNOME Desktop Environment

- GNOME is the default desktop for Redhat and Ubuntu
- Aims to provide a complete, user-friendly desktop built entirely on free software
- It has grown rapidly and is now widely adopted across Linux platforms

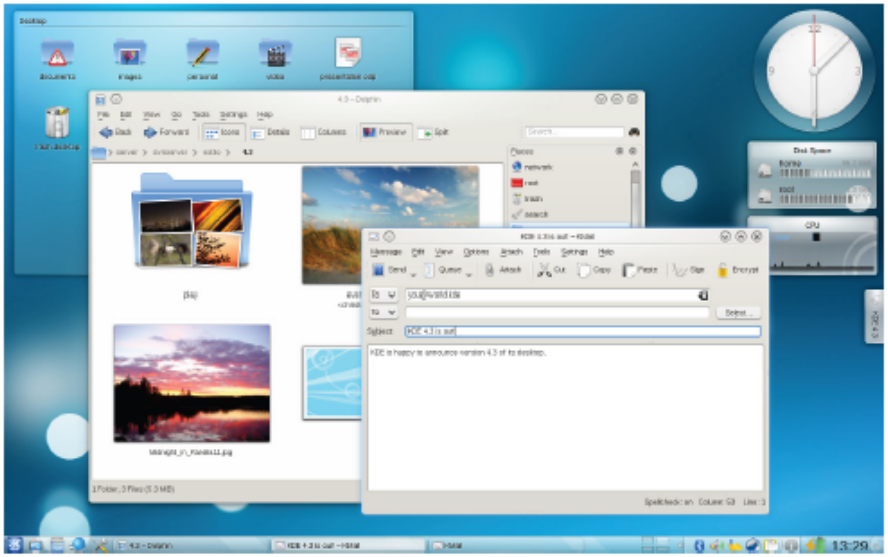
Main Components of GNOME

Component	Description
Top Bar	Displays time, system status (battery, network), and user menu
Activities Overview	Workspace switcher and app launcher, activated via top-left or Super key
Dash (Left Dock)	Vertical bar with favorite and running applications
Search Field	Allows searching for apps, files, and settings
Workspace Switcher	Enables switching between multiple virtual desktops

Different Linux Desktop Interfaces

KDE Desktop Environment

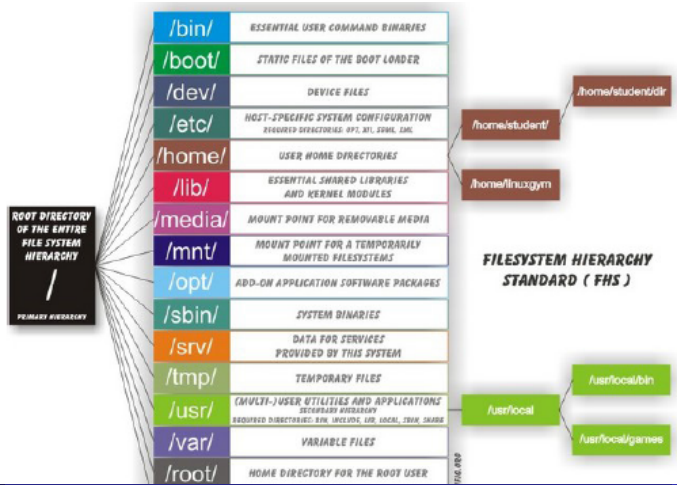
- Modern, network-transparent desktop environment
- Built on the **Qt** cross-platform GUI toolkit
- Includes core applications (Window manager (`kwm`), Web browser, Instant messenger, etc)
- Provides a consistent look and feel
- Translations available in 50+ languages



Linux File System

Linux File System Overview

- Unlike Windows that represents storage drives like c: d: etc., Linux does not represent drives separately
- The entire system is organized under a single **root directory** /
- Files, folders, and drives all exist within this unified hierarchy
- Standard core directories under / include:
  - `/home` – User files
  - `/etc` – Configuration files
  - `/var` – Logs and variable data
  - `/bin`, `/usr`, `/lib` – System programs and libraries
- The folder you operate in **95% of the time** is ~ (usually `/home/<username>`, e.g. `/home/lazhar/`)



Devices and Mounting

- In Linux, even devices exist as files under `/dev/`
- Example: inserting a CD creates a device file (e.g. `/dev/sdb1`)
- Drives are mounted to directories, e.g. `/media/cdrom`
- Mounting can be automatic or manual:
  - `mount /dev/sdb1 /media/cdrom`
  - Access via `/media/cdrom/`
  - Unmount with `umount /media/cdrom`
  - Some systems support `eject` (unmount + eject in one step)
- USB sticks, memory cards, and hard disks mount the same way
- Mount points can be anywhere, e.g. `/home/<username>/Storage` or `/mnt`
- Flexible and scalable design makes Linux device handling powerful

Command line interface and command prompt

Introduction to Linux Interfaces I

Linux provides two main types of interfaces for interacting with the system:

- **Graphical User Interface (GUI)** – Uses windows, icons, and menus. Examples include GNOME, KDE, XFCE.
- **Command Line Interface (CLI)** – Text-based interface where users type commands directly.

The **Command Line Interface (CLI)** is the most important interface to learn for Linux. Linux users will often be asked to perform commands through the CLI, and not through a GUI.

- After login, you can run the **CLI** through a tool like Terminal in Ubuntu, also called the **Command Prompt**.
- The prompt format is:

<username>@<computer>:<path>$

Introduction to Linux Interfaces II

- Example:

```
lazhar@lab-tl243-008:/home/lazhar$
```

- Anatomy of the CLI prompt:
  - `<username>` – your login name
  - `<computer>` – the hostname of the machine
  - `<path>` – current working directory
  - `$` – indicates a standard user prompt

Tip

IF YOU GET LOST: The tilde (~) represents your home directory. To return home, type:

cd ~

Why the command prompt? I

- Provides greater **control and flexibility** than the GUI
- Ensures **accuracy** — scripts can repeat tasks exactly
- More **secure** — avoids hidden actions by graphical tools
- CLI is **stable** over time, while GUIs change frequently
- Essential for **automation** and long-term tasks
- Learning CLI reduces confusion and builds **direct system interaction**

Handling Files and Directories in Linux

Navigating the File System I

- After login and running **Terminal**, you start in your **home directory**
- Each user has their own local home directory, independent from others
- Example: `/home/lazhar` (normally `/home/<username>`)
- Your home directory is private and safe to explore

The Concept of Access Paths

An **access path** is the route or address used to locate a file or directory in the file system hierarchy. It specifies the sequence of directories to traverse from a starting point to reach the target location.

Think of it like giving directions: you can either give the complete address from a known landmark (absolute), or give directions relative to where someone currently stands (relative).

Navigating the File System II

Types of Access Paths

1. **Absolute Path**
   - Starts from the root directory `/` (the top of the file system)
   - Always provides the complete location regardless of current position
   - Format: `/dir1/dir2/.../target`

Examples:

- `/home/lazhar/Desktop` — path to Desktop folder
- `/home/lazhar/Documents/report.pdf` — path to a specific file

2. **Relative Path**
   - Based on your **current working directory**
   - Shorter and more convenient when working within a project
   - Does *not* start with `/`

Navigating the File System III

Relative Path Examples Assume you are currently in `/home/lazhar`:

- `Desktop` — refers to `/home/lazhar/Desktop`
- `Documents/report.pdf` — refers to `/home/lazhar/Documents/report.pdf`
- `./Downloads` — same as `Downloads` (current directory)
- `../alice` — go up one level to `/home`, then into `alice`
- `../../etc` — go up two levels to root, then into `etc`

Special Path Symbols:

- `.` (dot) — represents the current directory
- `..` (dot dot) — represents the parent directory
- `~` (tilde) — shortcut for your home directory

Using the tilde (`~`):

- `~/Desktop` — expands to `/home/lazhar/Desktop`
- `~/Documents/notes.txt` — expands to `/home/lazhar/Documents/notes.txt`

Listing Files and Directories I

Explanation: The `ls` command is used to list the contents of a directory. By default, it shows visible files and folders, but with options you can reveal hidden files, display detailed information, or sort results.

Syntax:

General Form

ls [OPTION]... [Access path]...

Examples with Descriptions:

- `ls` – lists visible files and directories in the current directory
- `ls -a` – lists all files, including hidden ones (those starting with `.`)
- `ls -l` – long listing format: shows details such as size, and modification date
- `ls -lt` – long listing sorted by modification time (newest files first)

Listing Files and Directories II

- `ls -lrt` – long listing sorted by modification time (oldest files first, reverse order)
- `ls -R` – recursively lists all files and directories, including subdirectories

Making Directories

Explanation: The `mkdir` command creates new directories. It's useful for organizing files into separate folders. You can create single or nested directories, and then verify their existence with `ls`.

Syntax:

General Form

mkdir [OPTION]... directoryname

Examples:

- `mkdir unixstuff` – creates a new directory called `unixstuff` in the current location
- `mkdir -p unixstuff/projects/reports` – creates nested directories in one command (using the `-p` option)

Changing Directories

Explanation: The `cd` command changes your current working directory. It lets you move around the file system tree.

- `cd directory` – move into directory
- `cd ..` – go up one level
- `cd` – return to home directory
- `pwd` – show current path

Example

```
cd unixstuff
pwd
cd ..
cd
```

Special Directories

Explanation: In Linux, certain symbols represent special directories that make navigation easier. They act as shortcuts to refer to your current location, the parent directory, or your home directory without typing the full path.

- . – current directory
- .. – parent directory
- ~ – home directory

Example

```
ls ~
# list contents of the home directory
ls ..
# list contents of the parent of the current directory
```

Copying Files

Explanation: The `cp` command copies files or directories. With the `-R` option, it can copy entire directories recursively.

- `cp file1 file2` – copy file1 to file2
- `cp -R dir1 dir2` – copy directories recursively
- `cp file1 dir1/` – copy file1 to directory dir1

Moving and Renaming Files

Explanation: The `mv` command is used to move files from one location to another or to rename them. It can also rename directories.

- `mv file1 dir1/` – move file
- `mv file1 file2` – rename file
- `mv dir1/ dir2/` – rename directory

Example

```
mv science.bak backups/
```

Removing Files and Directories

Explanation: The `rm` and `rmdir` commands are used to delete files and directories. Be careful: deletions are permanent.

- `rm file` – remove file
- `rmdir dir` – remove empty directory
- `rm -r dir` – remove directory recursively

Example

```
rm tempfile.txt
rmdir tempstuff
rm -r backups
```

Viewing File Contents

Explanation: Linux provides several commands to view file contents. Use `cat` for short files, `less` for paging, and `head`/`tail` for partial views.

- `cat file` – display file
- `less file` – view file page by page
- `head file` – first 10 lines
- `tail file` – last 10 lines

Example

cat science.txt :displays the whole file
less science.txt :opens file in pager,scrollable
head -5 science.txt :shows first 5 lines of the file
tail -15 science.txt :shows last 15 lines of the file

Searching Files

Explanation: The `grep` command searches files for specific words or patterns. It supports options for case-insensitivity, line numbers, and counts.

- `grep keyword file` – search for keyword
- Options:
  - `-i` ignore case
  - `-v` invert match
  - `-n` show line numbers
  - `-c` count matches

Example

```
grep -i science science.txt
grep -ivc science science.txt
```

Word Count

Explanation: The `wc` (word count) command counts words, lines, and characters in a file. It's useful for quick statistics.

- `wc -w file` – count words
- `wc -l file` – count lines
- `wc -m file` – count characters

Example

```
wc –w science.txt
wc –l science.txt
wc –m science.txt
```

Installing Packages with APT

APT Package Manager

Explanation: APT (Advanced Package Tool) is the package manager for Debian/Ubuntu systems. It simplifies installing, updating, and removing software.

- APT is used in Debian/Ubuntu systems.
- It allows installing, updating, and removing software packages.
- Requires superuser privileges (use `sudo`).

Updating Package Lists

Explanation: Before installing software, update the package list to ensure you get the latest versions.

Command

```
sudo apt update
```

- Refreshes the list of available packages.
- Always run before installing new software.

Installing Packages

Explanation: Use `apt install` to download and install new software packages from repositories.

Command

```
sudo apt install package-name
```

- Installs the specified package.
- Example: `sudo apt install nano`

Removing Packages

Explanation: You can remove installed packages with `apt remove`. Use `apt purge` to also delete configuration files.

Command

```
sudo apt remove package-name
```

- Removes the package but keeps configuration files.
- Use `sudo apt purge package-name` to remove configs too.

Upgrading System

Explanation: APT can upgrade installed packages to newer versions. `full-upgrade` may install/remove packages to complete upgrades.

Commands

```
sudo apt upgrade
sudo apt full-upgrade
```

- `upgrade`: updates installed packages.
- `full-upgrade`: may remove/install packages to complete upgrade.

Searching Packages

Explanation: You can search for packages by keyword and view details before installing them.

Command

```
apt search keyword
apt show package-name
```

- `apt search`: find packages by keyword.
- `apt show`: display details about a package.