Introduction to Linux & Installation

Topics Covered

✓ Why Linux for DevOps & Cloud

✓ History, Open-source model, Distributions

✓ Architecture: Kernel, Shell, User space

✓ VirtualBox/VMWare & Ubuntu Installation

✓ Basic terminal usage

What is Linux?

🖥️ Operating System Basics

- **Operating System (OS)** - Software that manages computer hardware and software resources
- **Examples** - Windows, macOS, Linux, Android, iOS
- **Linux** - Free, open-source Unix-like operating system

🔄 How OS Works

- Hardware Management - Controls CPU, memory, storage, network
- Process Management - Runs multiple programs simultaneously
- File System - Organizes and stores data
- User Interface - GUI (graphical) or CLI (command-line)

🆚 Linux vs Windows/macOS

- **Cost** - Linux: Free | Windows/macOS: Paid licenses
- **Customization** - Linux: Highly customizable | Others: Limited
- **Security** - Linux: More secure by design | Others: More targeted by malware
- **Learning Curve** - Linux: Steeper initially | Others: More user-friendly

Understanding Open Source

🔓 What is Open Source?

- **Source Code** - The human-readable instructions that make software work
- **Open Source** - Source code is freely available to view, modify, and distribute
- **Proprietary** - Source code is hidden and owned by a company

🏗️ How Open Source Works

- Community Development - Thousands of developers worldwide contribute

- Peer Review - Code is reviewed by many experts
- Rapid Bug Fixes - Issues are found and fixed quickly
- No Vendor Lock-in - You're not dependent on one company

💡 Real-world Examples

- **Web Browsers** - Firefox, Chromium (base of Chrome)
- **Programming** - Python, JavaScript, Git
- **Databases** - MySQL, PostgreSQL, MongoDB
- **Cloud** - Docker, Kubernetes, Apache, Nginx

Why Linux for DevOps & Cloud?

🚀 DevOps Advantages

- Automation-friendly - Powerful scripting capabilities
- Container-native - Docker, Kubernetes run natively
- CI/CD Integration - Jenkins, GitLab CI, GitHub Actions
- Configuration Management - Ansible, Puppet, Chef support

☁️ Cloud Benefits

- Cost-effective - No licensing fees
- Scalability - Lightweight, efficient resource usage
- Security - Granular permissions, regular updates

- Cloud-native - AWS, Azure, GCP primarily Linux-based

History & Open-source Model

📅 Timeline

- **1991** - Linus Torvalds creates Linux kernel
- **1992** - Linux becomes GPL licensed
- **1993** - First distributions emerge (Slackware, Debian)
- **2000s** - Enterprise adoption (Red Hat, SUSE)
- **2010s** - Cloud & container revolution

🔓 Open-source Model

- Free to use - No licensing costs
- Source code available - Transparency & customization
- Community-driven - Global collaboration
- Rapid innovation - Continuous improvements

Linux Distributions

🏢 Enterprise

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

- **SUSE Linux Enterprise** - European enterprise focus
- **Ubuntu LTS** - Long-term support versions

🖥️ Desktop/Development

- **Ubuntu** - User-friendly, great for beginners
- **Fedora** - Cutting-edge features
- **Linux Mint** - Windows-like experience

⚡ Lightweight/Specialized

- **Alpine Linux** - Container-optimized
- **CentOS/Rocky Linux** - RHEL-compatible
- **Arch Linux** - Minimalist, rolling release

Understanding the Kernel

🧠 What is a Kernel?

- **Core of OS** - The most important part of the operating system
- **Bridge** - Connects software applications to hardware
- **Always Running** - Loaded into memory when computer starts

⚙️ Kernel Responsibilities

- Memory Management - Allocates RAM to programs
- Process Scheduling - Decides which program runs when

- Device Drivers - Communicates with hardware (keyboard, mouse, etc.)
- File System - Manages how data is stored and retrieved
- Network Stack - Handles internet and network connections

🔧 Types of Kernels

- **Monolithic** - Linux (everything in one piece, faster)
- **Microkernel** - Minimal core, services separate
- **Hybrid** - Windows, macOS (combination approach)

Linux File System Basics

📁 Everything is a File

- **Philosophy** - In Linux, everything is treated as a file
- **Regular Files** - Documents, images, videos
- **Directories** - Folders that contain other files
- **Device Files** - Hardware devices (hard disk, USB)

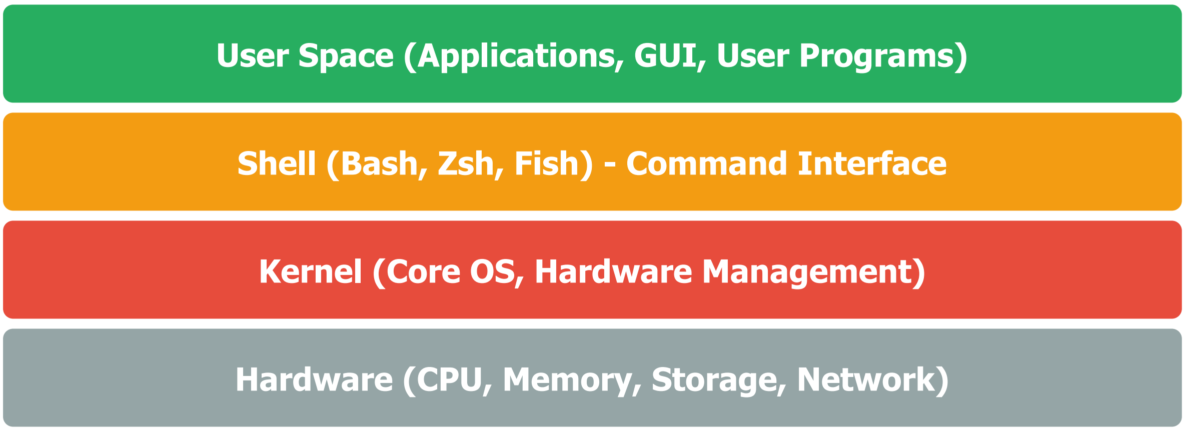
🌳 Directory Structure

```
/ # Root directory (top level)
├── home # User home directories
├── etc # Configuration files
├── var # Variable data (logs, databases)
├── usr # User programs and libraries
├── bin # Essential system programs
├── tmp # Temporary files
└── dev # Device files
```

🔐 File Permissions

- **Read (r)** - Can view file content
- **Write (w)** - Can modify file
- **Execute (x)** - Can run file as program
- **Users** - Owner, Group, Others

Linux Architecture



User Space (Applications, GUI, User Programs)

Shell (Bash, Zsh, Fish) - Command Interface

Kernel (Core OS, Hardware Management)

Hardware (CPU, Memory, Storage, Network)

🔧 Components Explained

- **Kernel** - Core OS, manages hardware resources
- **Shell** - Command-line interface, script execution
- **User Space** - Applications, services, user programs
- **System Calls** - Interface between user space and kernel

Git Bash - Linux Commands on Windows

🪟 For Windows Users

- **Problem** - Most participants have Windows, not Linux
- **Solution** - Git Bash provides Linux-like terminal on Windows
- **Bonus** - Also installs Git version control system

📥 Installing Git Bash

1. Go to **git-scm.com**
2. Download Git for Windows
3. Run installer with default settings
4. Right-click anywhere → "Git Bash Here"

✅ What Works in Git Bash

- File Operations - ls, cd, pwd, mkdir, cp, mv
- Text Processing - cat, grep, head, tail
- Basic Commands - whoami, which, echo
- Git Commands - git clone, git status, git commit

⚠️ Limitations

- **System Commands** - Some Linux system commands won't work
- **Package Management** - No apt, yum, or package managers
- **Services** - Can't manage Linux services

- **File Permissions** - Windows file system differences

Practice Options for Beginners

🎯 Choose Your Path

1. 🪟 Git Bash (Easiest Start)

- **Pros** - Quick setup, no VM needed, familiar Windows environment
- **Cons** - Limited Linux features, not real Linux experience
- **Best for** - Learning basic commands, file operations

2. 🖥️ Virtual Machine (Recommended)

- **Pros** - Full Linux experience, safe environment, can break and reset
- **Cons** - Requires more setup, uses system resources
- **Best for** - Complete Linux learning, system administration

3. ☁️ Online Linux Terminals

- **Options** - repl.it, codeanywhere.com, katacoda.com
- **Pros** - No installation, real Linux, accessible anywhere
- **Cons** - Internet required, limited time/features
- **Best for** - Quick practice, testing commands

VirtualBox/VMWare & Ubuntu Installation

🖥️ Virtualization Setup

- **VirtualBox** - Free, cross-platform
- **VMWare Workstation** - Professional features
- **System Requirements** - 4GB RAM, 25GB storage minimum

💿 Ubuntu Installation Steps

1. Download Ubuntu ISO from ubuntu.com
2. Create new VM (2GB RAM, 25GB disk)
3. Boot from ISO, select "Install Ubuntu"
4. Choose language, keyboard layout
5. Select installation type (erase disk for VM)
6. Create user account and password
7. Complete installation and reboot

Command Line Basics

💻 Why Learn Command Line?

- **Efficiency** - Faster than clicking through menus
- **Automation** - Can write scripts to repeat tasks
- **Remote Access** - Manage servers over internet
- **Precision** - Exact control over what happens

📝 Command Structure

```
command [options] [arguments]

Examples:
ls -l /home # command: ls, option: -l, argument: /home
cp file1.txt file2.txt # command: cp, arguments: file1.txt file2.txt
mkdir -p folder/subfolder # command: mkdir, option: -p, argument: folder/subfolder
```

🔤 Understanding Paths

- **Absolute Path** - Starts from root: `/home/user/documents`
- **Relative Path** - From current location: `documents/file.txt`
- **Home Directory** - ~ represents your home folder
- **Current Directory** - `.` represents where you are now
- **Parent Directory** - `..` represents one level up

Basic Terminal Usage

📁 File & Directory Operations

```
ls # List files and directories
cd /path # Change directory
pwd # Print working directory
mkdir dir # Create directory
rmdir dir # Remove empty directory
cp file1 file2 # Copy file
mv file1 file2 # Move/rename file
rm file # Remove file
```

📄 File Content Operations

```
cat file # Display file content
less file # View file page by page
head file # Show first 10 lines
tail file # Show last 10 lines
grep "text" file # Search for text in file
```

🔧 System Information

```
whoami # Current username
uname -a # System information
df -h # Disk usage
free -h # Memory usage
ps aux # Running processes
```

Hands-on Practice

💡 These exercises work in Git Bash, VM, or online terminals!

🎯 Exercise 1: Navigation

```
# Try these commands step by step:
pwd # See where you are
ls # List files in current directory
cd / # Go to root directory (C:\ in Git Bash)
ls # See what's in root
cd ~ # Go back to home directory
pwd # Confirm you're home
```

🎯 Exercise 2: File Operations

```
# Create and manage files:
mkdir practice # Create a directory
cd practice # Enter the directory
touch myfile.txt # Create an empty file
echo "Hello Linux" > myfile.txt # Add content to file
cat myfile.txt # Display file content
```

```
cp myfile.txt backup.txt # Copy the file
ls -l # List files with details
```

🎯 Exercise 3: Getting Help

```
man ls # Read manual for 'ls' command (may not work in Git
Bash)
ls --help # Quick help for 'ls' (works everywhere)
which ls # Find where 'ls' program is located
```

Common Beginner Mistakes & Tips

❌ Common Mistakes

- **Case Sensitivity** - `File.txt` and `file.txt` are different
- **Spaces in Names** - Use quotes: `"my file.txt"` or escape: `my\ file.txt`
- **Wrong Directory** - Always check where you are with `pwd`
- **Permissions** - Can't modify files you don't own

💡 Helpful Tips

- Tab Completion - Press Tab to auto-complete commands/files
- Command History - Use ↑ arrow to see previous commands
- Ctrl+C - Stop a running command
- Clear Screen - Type `clear` or press Ctrl+L

🆘 When Things Go Wrong

- **Command Not Found** - Check spelling, use `which command`

- **Permission Denied** - Use `sudo` for admin tasks
- **File Not Found** - Check path with `ls` and `pwd`

Summary

🎯 Key Takeaways

- Linux is essential for DevOps and Cloud computing
- Open-source model provides flexibility and cost savings
- Multiple distributions cater to different needs
- Layered architecture separates concerns effectively
- Terminal skills are fundamental for Linux mastery

🚀 Next Steps

- Practice terminal commands daily
- Explore different Linux distributions
- Learn shell scripting (Bash)
- Understand file permissions and users
- Study system administration basics