Linux Based Networking - Class 01: Linux Fundamentals & Networking Basics

Prepared for Premier University Computer Club

Spring 2025

Updated: September 10, 2025

This document serves as a detailed guide to complement the class slides (slide_class1.pdf). It provides in-depth explanations, examples, analogies, and practical tips for each topic. Use this to prepare for teaching, with references to slide pages where relevant. Stats have been updated based on the latest available data as of September 2025.

1. History of Linux & Open-Source Philosophy

Linux is a free and open-source operating system kernel created by Linus Torvalds in 1991 while he was a student at the University of Helsinki. Inspired by the UNIX operating system, Torvalds developed Linux as a hobby project and released it under the GNU General Public License (GPL), which allows anyone to view, modify, and distribute the source code freely. This licensing choice was key to Linux's growth, as it aligned with the open-source philosophy pioneered by Richard Stallman and the Free Software Foundation (FSF) through the GNU Project.

The open-source philosophy emphasizes collaboration, transparency, and community-driven development. Key principles include:

- **Freedom to Use, Modify, and Distribute**: Users can run the software for any purpose, study and change the code, and share improvements.

- **Community Collaboration**: Projects like Linux thrive on contributions from developers worldwide via platforms like GitHub.

- **Benefits**: Faster innovation, better security (through peer review), and cost savings (no licensing fees).

Examples of open-source projects (as shown on Slide Page 5):

- Apache HTTP Server: Powers web servers.

- Linux Kernel: The core of Linux OS.

- Git: Version control system.

- Node.js: JavaScript runtime.

- Docker: Containerization tool.

- xAI (Visual Studio Code): Popular code editor.

- TensorFlow: Machine learning framework (screenshot example on slide shows GitHub repo).

Updated Stats (September 2025):

- Apache HTTP Server is used by approximately 25.7% of all websites whose web server is known.

- 100% of the world's top 500 supercomputers run on Linux.

- Linux powers about 78% of web-facing servers worldwide.

Teaching Tips: Start with Linus Torvalds' story (refer to Slide Page 4 image of a person explaining, likely Torvalds). Discuss how open-source contrasts with proprietary software (e.g., Windows). Ask students: "Why might companies like Google use Linux for Android?"

2. Popular Linux Distributions (Debian, Ubuntu, CentOS, RHEL)

Linux distributions (or "distros") are complete operating systems built around the Linux kernel, including software packages, tools, and user interfaces. They vary in focus: some for desktops, others for servers or beginners.

Popular ones (based on Slide Page 9 diagram):

- **Distro Families** (branching from the Linux Kernel):
  - **Debian Family**: Known for stability and free software focus.
    - Debian: Base distro, used in servers and desktops.
    - Ubuntu: User-friendly, based on Debian. Great for beginners with a large community.
    - Linux Mint: Based on Ubuntu, focuses on ease-of-use with a Windows-like interface.
  - **Fedora Family**: Sponsored by Red Hat, emphasizes cutting-edge features.
    - Fedora: Community-driven, often tests new tech.

- RHEL (Red Hat Enterprise Linux): Commercial, stable for enterprises. Holds ~43% of enterprise Linux server market.

- CentOS: Free alternative to RHEL (now CentOS Stream focuses on future RHEL development).

- Oracle Linux: Based on RHEL, optimized for Oracle software.

- **SUSE Family**: Enterprise-focused with strong support.

- SUSE: Commercial version.

- OpenSUSE: Community edition.

- SLES (SUSE Linux Enterprise Server): For servers.

- **Other Distributions**: Arch Linux (customizable), Kali Linux (security testing), etc.

Key Differences:

- **Debian/Ubuntu**: APT package manager, stable releases. Ubuntu has ~34% share in Linux distributions.

- **RHEL/CentOS**: YUM/DNF package manager, enterprise support.

- **Choosing a Distro**: For desktops: Ubuntu/Mint. For servers: RHEL/Ubuntu Server.

Teaching Tips: Use the tree diagram from Slide Page 9 to visualize families. Discuss real-world use: Android is based on Linux, AWS uses Ubuntu/RHEL. Lab idea: Install Ubuntu in a VM.

3. Linux Boot Process Overview

The boot process is how Linux starts from power-on to a usable system. (Refer to Slide Page 10 diagram: BIOS → MBR → GRUB → Kernel → Init → Runlevels.)

Steps:

1. **BIOS/UEFI**: Firmware checks hardware integrity and loads the bootloader from the boot device (e.g., HDD, USB).

2. **MBR (Master Boot Record)**: First sector of the boot disk, executes the bootloader.

3. **GRUB (Grand Unified Bootloader)**: Common bootloader (Slide Page 8 example). Displays a menu to select OS/kernel. Config in /boot/grub/grub.conf.

4. **Kernel**: Loads into memory, mounts root filesystem, uses initrd (initial RAM disk) for drivers, then starts init (PID 1).

5. **Init/Systemd**: Modern init system (Slide Page 8). Starts services based on runlevels/targets:

- 0: Shutdown
- 1: Single-user (rescue)
- 3: Multi-user (CLI)
- 5: Graphical
- 6: Reboot
- Emergency: Fix critical issues.

6. **Runlevels**: Scripts in /etc/rcX.d/ (X = runlevel) start (S) or stop (K) services.

Analogy: Like starting a car – BIOS checks engine, GRUB turns the key, kernel revs up, init turns on AC/radio.

Teaching Tips: Walk through the flowchart on Slide Page 10. Common issues: GRUB errors (fix with rescue mode).

4. Linux File System Layout (/etc, /var, /usr, /home, /dev, /proc)

Linux uses a hierarchical file system starting from / (root). (Refer to Slide Page 11 diagram: Tree from / with branches like /bin, /dev, etc.)

Key Directories:

1. **/**: Root, contains everything.

2. **/bin**: Essential binaries (ls, cp).

3. **/boot**: Boot files (kernel, GRUB).

4. **/dev**: Device files (e.g., /dev/sda for hard drive).

5. **/etc**: Configuration files (e.g., /etc/passwd for users).

6. **/home**: User home directories (e.g., /home/user).

7. **/lib**: Libraries for binaries.

8. **/media**: Mount points for removable media.

9. **/mnt**: Manual mounts.

10. **/opt**: Optional software.

11. **/proc**: Virtual filesystem for process/kernel info.

12. **/root**: Root user's home.

13. **/sbin**: System admin binaries.

14. **/srv**: Service data (e.g., web files).

15. **/tmp**: Temporary files (cleared on reboot).

16. **/usr**: User applications (/usr/bin, /usr/lib).

17. **/var**: Variable data (/var/log for logs).

Analogy (from original): /bin = basic tools, /etc = settings, /home = user stuff, etc.

Teaching Tips: Use the directory tree from Slide Page 11. Command: `tree /` (install tree if needed). Explain why everything is a file in Linux.

5. Essential CLI Commands

Command-Line Interface (CLI) basics for navigation and management.

1. **ls**: List files. `ls -l` for details.

2. **cd**: Change directory. `cd ~` to home.

3. **cp**: Copy. `cp file.txt copy.txt`.

4. **mv**: Move/rename. `mv file.txt dir/`.

5. **rm**: Remove. `rm file.txt` (careful, no recycle bin!).

6. **touch**: Create file. `touch new.txt`.

7. **mkdir**: Make directory. `mkdir folder`.
8. **rmdir**: Remove empty directory.
9. **man**: Manual. `man ls`.
10. **clear**: Clear screen.

Tips: Use tab for auto-complete. `pwd` for current path.

Teaching Tips: Practice in lab (Slide Page 3: Navigate directories and manage files).

6. User & Group Management (useradd, passwd, groupadd, usermod, id)

Manage access with users and groups (Slide Page 12).

1. **useradd**: Add user. `sudo useradd -m alice` (with home).
2. **passwd**: Set password. `sudo passwd alice`.
3. **groupadd**: Add group. `sudo groupadd developers`.
4. **usermod**: Modify. `sudo usermod -aG developers alice`.
5. **id**: Show info. `id alice`.

Difference: Users are individuals; groups share permissions.

Exercise (from original): Create bob, set password, add to testers group, check id.

Teaching Tips: Emphasize sudo. Lab: Create users with privileges (Slide Page 3).

7. File Ownership & Permissions (chmod, chown, umask)

Control who accesses files (Slide Page 12 & 13 table).

Permissions: r (4/read), w (2/write), x (1/execute). Categories: user (u), group (g), others (o).

- Numeric: 755 = rwxr-xr-x.
- Table (from Slide Page 13):
  - 000 (---): No permission.
  - 001 (--x): Execute.
  - 010 (-w-): Write.
  - 011 (-wx): Write + Execute.
  - 100 (r--): Read.
  - 101 (r-x): Read + Execute.
  - 110 (rw-): Read + Write.
  - 111 (rwx): All.

Commands:

1. **chmod**: Change permissions. `chmod 755 file.txt` or `chmod u+x script.sh`.
2. **chown**: Change owner/group. `sudo chown alice:developers file.txt`.
3. **umask**: Default mask. `umask 022` → New files 644 (rw-r--r--).

Exercise: Create file, set 644, change owner, test umask.

Teaching Tips: Use the table from Slide Page 13. Explain security: Why restrict others?

8. File Viewing & Editing (cat, less, nano, vim)

Handle text files.

Viewing:

- **cat**: Full content. `cat file.txt`.
- **less**: Scrollable. `less big.txt` (q to quit).
- **head**: First 10 lines. `head log.txt`.
- **tail**: Last 10. `tail -f log.txt` (real-time).

Editing:
- **nano**: Beginner-friendly. Ctrl+O save, Ctrl+X exit.
- **vim**: Advanced. i insert, Esc command, :wq save/quit.

Tips: Nano for newbies, vim for power users.

Teaching Tips: Demo in terminal.

9. OSI & TCP/IP Models Overview

OSI Model (7 Layers, from original):
1. Application: User services (HTTP, FTP).
2. Presentation: Data format/encryption (SSL, JPEG).
3. Session: Manages connections (NetBIOS).
4. Transport: Reliable delivery (TCP, UDP).
5. Network: Routing (IP).
6. Data Link: Local transfer (Ethernet).
7. Physical: Hardware signals.

TCP/IP Model (4 Layers, practical internet model):
1. Application: Combines OSI 5-7 (HTTP, FTP).
2. Transport: TCP (reliable), UDP (fast).
3. Internet: IP addressing/routing.

4. Link: Combines OSI 1-2 (Ethernet, Wi-Fi).

Comparison: OSI is theoretical; TCP/IP is used in practice. Analogy: OSI like detailed blueprint, TCP/IP like built house.

Teaching Tips: Use analogy from original (sending a package). Draw layers.

10. IPv4 vs IPv6 Addressing

IP addresses identify devices on networks.

IPv4:

- 32-bit: e.g., 192.168.1.1 (4 octets, ~4.3 billion addresses).
- Exhausted; uses NAT for sharing.
- Features: Simple, widespread.

IPv6:

- 128-bit: e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334 (huge address space, 340 undecillion).
- Benefits: No NAT needed, built-in security (IPsec), auto-configuration, better routing.
- Adoption: Growing; ~40% of internet traffic in 2025.

Differences:

- **Address Space**: IPv4 running out; IPv6 abundant.
- **Format**: IPv4 decimal; IPv6 hex with colons.
- **Header**: IPv6 simpler, efficient.
- **Transition**: Dual-stack (both), tunneling.

Teaching Tips: Explain why IPv6 is future-proof. Command: `ip addr` to see addresses. Lab: Test connectivity (Slide Page 3).