Course overview

CSYE 6225: Network Structure & Cloud Computing
Northeastern University

Instructor: Raja Alomari,PhD

Day 1

- Read syllabus: will go over syllabus in class.
- Github: We will use Github classrooms for several assignments. Link to join will be shared in Canvas. Please create a github account.
- Survey: Anonymous background survey link will be shared.
- Topic: Linux commands and File System

Linux commands and File System

Objectives:

- Understand and apply basic Linux commands,
- Navigate the Linux file system,
- Gain insights into the Linux ecosystem compared to other operating systems.

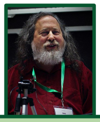
Hands-on:

- Accompanied with hands-on demos.

What is Linux?

- Linux is an open-source operating system kernel initially created by Linus Torvalds in 1991.
- It is widely used in servers, desktops, and embedded systems due to its stability, security, and flexibility.
- Richard Stallman and others wrote the GNU utilities
- Kernel + GNU -> Linux OS as we know it today.

Linus Torvalds


Richard Stallman

Timeline of Linux:

1991

Linus Torvalds releases the first version of the Linux kernel (version 0.01) on September 17, 1991.

The initial release includes basic features like file system support, multitasking, and a terminal.

1992

GNU Utilities Integration

The Linux kernel is combined with GNU utilities to form a complete operating system.

Free and open-source alternative to Unix.

1994

Linux Kernel 1.0

The first official stable release of the Linux kernel (version 1.0) is made available.

This version includes support for networking and the ext2 file system.

1996

Tux, the penguin, is adopted as the official mascot of Linux.

The kernel reaches version 2.0, introducing support for SMP (Symmetric Multiprocessing).

Timeline of Linux:

1998

Major Linux distributions such as Red Hat, Debian, and Slackware become popular.

The open-source community grows, with more developers contributing to Linux.

2003

Linux Kernel 2.6

The Linux kernel is combined with GNU utilities to form a complete operating system.

Free and open-source alternative to Unix.

2009

Linux Kernel 2.6.29

Introduction of the ext4 file system

2011

20th Anniversary

kernel version 3.0

Timeline of Linux:

2015

Linux Kernel 4.0

Introduction of live patching, allowing kernel updates without system reboots.

2020

Linux Kernel 5.0

Linux dominates the server market and powers a majority of the world's supercomputers.

2023

Linux Kernel 6

2024

Linux Kernel 6.9

Linux Commands: Overview

Network	Symbols	File System
Shells	Filters	File Editors
System Information	Hotkeys	Line Editors

Basic Linux Commands: Network

SSH (Secure Shell)

- Description: SSH is a protocol used to securely log into a remote computer and execute commands.
- Syntax: ssh [user@]hostname [command]
- Examples:
  - Connecting to a remote server: ssh user@remote_host
  - Running a command on a remote server: ssh user@remote_host 'ls -l'
- Common Use Cases:
  - Securely accessing remote servers
  - Transferring files securely (using scp)

SCP (Secure Copy)

- Description: SCP is used to securely copy files between hosts on a network.
- Syntax: scp [options] [user@]src_host:]file1 [user@]dest_host:]file2
- Examples:
  - Copying a file to a remote server: scp file.txt user@remote_host:/path/to/destination
  - Copying a file from a remote server: scp user@remote_host:/path/to/source/file.txt /local/path
- Common Use Cases:
  - Securely transferring files between local and remote machines
  - Automating backups

Basic Linux Commands: Network

Ping

- Description: Ping is used to test the reachability of a host on an IP network.
- Syntax: ping [options] destination
- Examples:
  - Pinging a host: ping google.com
  - Specifying the number of packets: ping -c 4 google.com
- Common Use Cases:
  - Checking network connectivity
  - Diagnosing network issues

Telnet

- Description: Telnet is a protocol used to connect to remote computers over a TCP/IP network.
- Syntax: telnet [hostname] [port]
- Examples:
  - Connecting to a remote server: telnet remote_host
  - Specifying a port: telnet remote_host 80
- Common Use Cases:
  - Accessing remote servers (less secure than SSH)
  - Testing and troubleshooting network services

Basic Linux Commands: Network

Nslookup

- Description: Nslookup is used to query Internet domain name servers.
- Syntax: nslookup [options] [domain]
- Examples:
  - Looking up an IP address: nslookup google.com
  - Specifying a DNS server: nslookup google.com 8.8.8.8
- Common Use Cases:
  - Diagnosing DNS issues
  - Gathering domain information

Wget

- Description: Wget is a command-line utility to download files from the web.
- Syntax: wget [options] [URL]
- Examples:
  - Downloading a file: wget http://example.com/file.zip
  - Downloading a file to a specific directory: wget -P /path/to/dir http://example.com/file.zip
- Common Use Cases:
  - Downloading files from the internet
  - Automating file downloads in scripts

Basic Linux Commands: Shell

BASH (Bourne Again Shell)

- Description: BASH is the default shell on many Linux distributions, providing a command-line interface for interacting with the operating system.
- Features:
  - Command history
  - Command-line editing
  - Job control
- Common Commands:
  - ls, cd, pwd, cp, mv, rm
  - Scripting with loops, conditionals, and functions

Clear

- Description: Clear the terminal screen.
- Syntax: clear
- Examples:
  - Simply type clear to clear the screen
- Common Use Cases:
  - Cleaning up the terminal for better readability

Basic Linux Commands: Shell

History

- Description: Displays the command history.
- Syntax: history [options]
- Examples:
  - Viewing history: history
  - Running a command from history: !23
- Common Use Cases:
  - Repeating previous commands
  - Searching for past commands

Echo

- Description: Display a line of text/string.
- Syntax: echo [options] [string]
- Examples:
  - Printing text: echo "Hello, World!"
  - Displaying variables: echo $PATH
- Common Use Cases:
  - Outputting text to the terminal
  - Displaying environment variables

Basic Linux Commands: System Information

w

- Description: Displays information about currently logged-in users and their processes.
- Syntax: w [options]
- Examples:
  - Basic usage: w
  - Display help: w --help

whoami

- Description: Prints the current user's username.
- Syntax: whoami
- Examples:
  - Basic usage: whoami

Basic Linux Commands: System Information

man

- Description: Displays the manual pages for commands.
- Syntax: man [command]
- Examples:
  - Viewing manual for ls: man ls

info

- Description: Displays documentation in info format, usually more detailed than man.
- Syntax: info [command]
- Examples:
  - Viewing info for ls: info ls

Basic Linux Commands: System Information

which

- Description: Locates the executable file associated with a given command.
- Syntax: which [command]
- Examples:
  - Finding path of ls: which ls

free

- Description: Displays the amount of free and used memory in the system.
- Syntax: free [options]
- Examples:
  - Basic usage: free
  - Detailed output: free -h

Basic Linux Commands: System Information

date

- Description: Displays or sets the system date and time.
- Syntax: date [options] [+format]
- Examples:
  - Displaying current date and time: date
  - Custom format: date "+%Y-%m-%d %H:%M:%S"

cal

- Description: Displays a calendar.
- Syntax: cal [options] [[month] year]
- Examples:
  - Displaying current month: cal
  - Displaying a specific month and year: cal 12 2024

Basic Linux Commands: System Information

df

- Description: Displays the amount of disk space available on the file system.
- Syntax: df [options]
- Examples:
  - Basic usage: df
  - Human-readable format: df -h

Basic Linux Commands: Filters

grep

- Description: Searches for patterns within files or input.
- Syntax: grep [options] pattern [file...]
- Examples:
  - Searching for "error" in a file: grep "error" logfile.txt
  - Case-insensitive search: grep -i "error" logfile.txt
- Common Use Cases:
  - Searching logs or text files for specific patterns
  - Filtering output from other commands

egrep

- Description: Extended version of grep that supports extended regular expressions.
- Syntax: egrep [options] pattern [file...]
- Examples:
  - Searching with extended regex: egrep "error|warning" logfile.txt
- Common Use Cases:
  - Advanced pattern matching with extended regular expressions

Basic Linux Commands: Filters

more

- Description: View file content one page at a time.
- Syntax: more [file...]
- Examples:
  - Viewing a file: more logfile.txt
- Common Use Cases:
  - Paging through long files or command outputs

less

- Description: Similar to more, but allows both forward and backward navigation.
- Syntax: less [file...]
- Examples:
  - Viewing a file: less logfile.txt
- Common Use Cases:
  - Interactive paging with advanced navigation features

Basic Linux Commands: Filters

head

- Description: Outputs the first part of files.
- Syntax: head [options] [file...]
- Examples:
  - Displaying the first 10 lines: head logfile.txt
  - Displaying the first 20 lines: head -n 20 logfile.txt
- Common Use Cases:
  - Previewing the beginning of files

tail

- Description: Outputs the last part of files.
- Syntax: tail [options] [file...]
- Examples:
  - Displaying the last 10 lines: tail logfile.txt
  - Displaying the last 20 lines: tail -n 20 logfile.txt
  - Following a file (live updates): tail -f logfile.txt
- Common Use Cases:
  - Monitoring log files in real-time

Basic Linux Commands: Line Editors

awk

- Description: awk is a versatile programming language for pattern scanning and processing.
- Syntax: awk 'pattern { action }' [file...]
- Examples:
  - Print the first column: awk '{print $1}' file.txt
  - Print lines where the second column is greater than 50: awk '$2 > 50' file.txt
  - Sum values in the third column: awk '{sum += $3} END {print sum}' file.txt
- Common Use Cases:
  - Data extraction and reporting
  - Text manipulation and transformation

sed

- Description: sed is a stream editor used for basic text transformations on an input stream.
- Syntax: sed [options] 'command' [file...]
- Examples:
  - Replace 'foo' with 'bar': sed 's/foo/bar/' file.txt
  - Delete lines containing 'error': sed '/error/d' file.txt
  - Print lines 2 to 4: sed -n '2,4p' file.txt
- Common Use Cases:
  - In-place editing of files
  - Batch processing and text manipulation

Basic Linux Commands: File Operations

ls - List directory contents

- ls Command
  - Usage: ls
  - Options: -l (long format), -a (all files)
- Examples
  - ls -l /home/user
  - ls -a /var/log

cp - Copy files and directories

- cp Command
  - Usage: cp source destination
  - Options: -r (recursive)
- Examples
  - cp file.txt /backup/
  - cp -r dir1/ dir2/

mv - Move or rename files and directories

- mv Command
  - Usage: mv source destination
- Examples
  - mv oldname.txt newname.txt
  - mv /home/user/docs /archive/

rm - Remove files or directories

- rm Command
  - Usage: rm file
  - Options: -r (recursive)
- Examples
  - rm temp.txt
  - rm -r /old/

Basic Linux Commands: Directory Navigation

pwd - Print working directory

- pwd Command

cd - Change directory

- cd Command
  - Usage: cd directory
  - Special directories:
    - ~ : home directory,
    - .. :parent directory

Examples

- cd /var/log
- cd ..
- cd ~

Basic Linux Commands: **File Viewing and Editing**

cat - Concatenate and display file content

- cat Command

more and less - View file content page by page

- more and less Commands

Head and tail - view beginning and ending of file contents.

emacs -nw

- Description: emacs -nw runs emacs in the terminal, without a graphical interface.
- Features:
  - Full functionality of emacs in text mode
  - Suitable for remote or low-resource environments
- Common Commands:
  - Save: C-x C-s
  - Quit: C-x C-c
  - Search: C-s
  - Replace: M-%
- Advantages:
  - Runs in terminal environments
  - Allows for extensive text editing features

Basic Linux Commands: **File Viewing and Editing**

gvim

- Description: gvim is the graphical version of vim, providing a graphical user interface.
- Features:
  - All vim functionalities with a GUI
  - Menu bars and toolbars for easier access
  - Mouse support
- Common Commands:
  - Same as vim with additional GUI-based options
- Advantages:
  - Enhanced visual feedback
  - Easier access to menus and options

vim

- Description: vim (Vi IMproved) is a highly configurable text editor built to enable efficient text editing.
- Modes:
  - Normal Mode: Default mode for navigation and commands.
  - Insert Mode: For text entry.
  - Command Mode: For executing commands.
- Common Commands:
  - Enter Insert Mode: i
  - Save and Quit: :wq
  - Quit without Saving: :q!
  - Search: /pattern
  - Replace: :%s/old/new/g
- Features:
  - Syntax highlighting
  - Multi-level undo/redo
  - Customizable with plugins

Basic Linux Commands: **File Viewing and Editing**

nano

- Description: nano is a simple, user-friendly text editor for the terminal.
- Features:
  - Easy to use with on-screen command shortcuts
  - Minimal learning curve compared to other editors
  - Suitable for quick edits and small scripts
- Common Commands:
  - Save: Ctrl + O (then press Enter)
  - Quit: Ctrl + X
  - Search: Ctrl + W
  - Replace: Ctrl + \
  - Cut: Ctrl + K
  - Paste: Ctrl + U
- Advantages:
  - Straightforward and intuitive interface
  - Good for users who need a simple editor without complex features

emacs

- Description: emacs is a highly extensible and customizable text editor.
- Features:
  - Rich set of editing commands
  - Built-in Lisp interpreter for customization
  - Integrated tools (e.g., file manager, debugger)
- Common Commands:
  - Save: C-x C-s
  - Quit: C-x C-c
  - Search: C-s
  - Replace: M-%
- Advantages:
  - Extensive customization options
  - Built-in support for many programming languages and tools

Basic Linux Commands: **File Permissions**

chmod - Change file permissions

- chmod Command
- Syntax: chmod permissions file
- Examples
  - chmod 755 file.txt
  - Description: Sets read, write, and execute permissions for the owner, and read and execute permissions for others.
  - chmod +x script.sh
  - Description: Adds execute permission to script.sh.

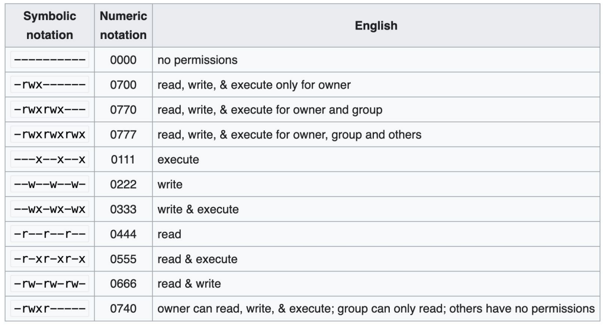
Permission Types:

- "r - Read"
- "w - Write"
- "x - Execute"

chown - Change file owner

- chown Command
- Syntax: chown user:group file
- Examples
  - chown user:group file.txt
  - Description: Changes ownership of file.txt to user and group.

Basic Linux Commands: Owner, group, and others



Symbolic notation	Numeric notation	English
`----------`	0000	no permissions
`-rwx------`	0700	read, write, & execute only for owner
`-rwxrwx---`	0770	read, write, & execute for owner and group
`-rwxrwxrwx`	0777	read, write, & execute for owner, group and others
`---x--x--x`	0111	execute
`--w--w--w-`	0222	write
`--wx-wx-wx`	0333	write & execute
`-r--r--r--`	0444	read
`-r-xr-xr-x`	0555	read & execute
`-rw-rw-rw-`	0666	read & write
`-rwxr-----`	0740	owner can read, write, & execute; group can only read; others have no permissions

Basic Linux Commands: **Special Characters**

| (Pipe)

- Description: Connects the output of one command to the input of another command.
- Syntax: command1 | command2
- Examples:
  - Count lines in a file: cat file.txt | wc -l
  - Search and count occurrences: grep "pattern" file.txt | wc -l

> (Redirect Output)

- Description: Redirects the output of a command to a file, overwriting the file if it exists.
- Syntax: command > file
- Examples:
  - Save command output to a file: ls > filelist.txt
  - Overwrite content: echo "Hello" > greetings.txt

Basic Linux Commands: **Special Characters**

>> (Append Output)

- Description: Appends the output of a command to the end of a file.
- Syntax: command >> file
- Examples:
  - Append command output to a file: echo "New line" >> file.txt
  - Add data without overwriting: date >> logfile.txt

< (Redirect Input)

- Description: Redirects input from a file to a command.
- Syntax: command < file
- Examples:
  - Use file as input for a command: sort < unsorted.txt
  - Provide input from a file: mail -s "Subject" recipient@example.com < message.txt

Basic Linux Commands: **Special Characters**

& (Background Execution)

- Description: Executes a command in the background, allowing the terminal to be used for other commands.
- Syntax: command &
- Examples:
  - Run a command in the background: long-running-task &
  - Start a server: python -m http.server &

>& (Redirect Output and Error)

- Description: Redirects both output and error streams to a file or another command.
- Syntax: command > file 2>&1
- Examples:
  - Redirect output and errors to the same file: command > output.log 2>&1
  - Combine error and standard output: ls non_existent_file > result.log 2>&1

Basic Linux Commands: **Special Characters**

2>&1 (Redirect Error to Output)

Content:

- Description: Redirects standard error (file descriptor 2) to the same location as standard output (file descriptor 1).
- Syntax: command > file 2>&1
- Examples:
  - Combine output and error in a single file: command > output.log 2>&1
  - Display combined output and errors: command 2>&1

; (Command Separator)

- Description: Separates multiple commands to be executed sequentially.
- Syntax: command1 ; command2
- Examples:
  - Run commands in sequence: echo "Start" ; ls ; echo "End"
  - Execute multiple commands: command1 ; command2 ; command3

Basic Linux Commands: **Special Characters**

~ (Home Directory)

- Description: Represents the current user's home directory.
- Syntax: cd ~ or ~/file
- Examples:
  - Change to home directory: cd ~
  - Access a file in home directory: cat ~/file.txt

. (Current Directory)

- Description: Represents the current directory.
- Syntax: ./file
- Examples:
  - Execute a script in the current directory: ./script.sh
  - List contents of the current directory: ls .

Basic Linux Commands: **Special Characters**

.. (Parent Directory)

- Description: Represents the parent directory of the current directory.
- Syntax: cd ..
- Examples:
  - Move up one directory: cd ..
  - List contents of the parent directory: ls ..

$! (Last Background Process ID)

- Description: Returns the process ID (PID) of the most recently executed background command.
- Syntax: $!
- Examples:
  - Get PID of the last background job: echo $!
  - Use PID for monitoring: ps -p $!

Basic Linux Commands: Special Characters

!:<n> (History Expansion)

Content:

- Description: Expands a command from history by its position.
- Syntax: !:<n>
- Examples:
  - Run the nth command from history: !2
  - Re-execute the second-to-last command: !:-2

!<n> (History Expansion by Number)

Content:

- Description: Repeats the command from the history list by its number.
- Syntax: !<n>
- Examples:
  - Execute command number 5 from history: !5
  - View the command: history then use !<n>

Basic Linux Commands: Service Management

systemctl

- Description: Manages systemd services and the system state.
- Syntax: systemctl [command] [service]
- Common Commands:
  - Start a service: systemctl start service-name
  - Stop a service: systemctl stop service-name
  - Restart a service: systemctl restart service-name
  - Enable a service: systemctl enable service-name (starts on boot)
  - Disable a service: systemctl disable service-name (does not start on boot)
  - Check service status: systemctl status service-name
  - View all services: systemctl list-units --type=service
- Examples:
  - Start Apache web server: systemctl start apache2
  - Check status of SSH service: systemctl status ssh

service

- Description: Manages SysVinit services. Works on systems using SysVinit or compatible init systems.
- Syntax: service [service] [command]
- Common Commands:
  - Start a service: service service-name start
  - Stop a service: service service-name stop
  - Restart a service: service service-name restart
  - Check service status: service service-name status

Basic Linux Commands: Service Management

ps

- Description: Displays information about active processes.
- Syntax: ps [options]
- Common Options:
  - List processes: ps aux (all users, detailed)
  - Current shell processes: ps
  - Tree view: ps -ejH (shows process hierarchy)
- Examples:
  - List all processes: ps aux
  - Show process tree: ps -ef --forest

top

- Description: Displays real-time information about system processes and resource usage.
- Syntax: top
- Common Features:
  - Sort by CPU usage: Default view
  - Interactive commands:
    - Kill process: Press k, then enter PID
    - Sort by memory: Press M
    - Change update interval: Press d, then enter seconds

Basic Linux Commands: Service Management

kill

- Description: Sends signals to processes, usually to terminate them.
- Syntax: kill [signal] PID
- Common Signals:
  - Terminate: kill PID (default signal SIGTERM)
  - Forceful kill: kill -9 PID (signal SIGKILL)
  - List signals: kill -l
- Examples:
  - Terminate process by PID: kill 1234
  - Forcefully kill process: kill -9 1234

pkill

- Description: Sends signals to processes based on name or other attributes.
- Syntax: pkill [options] name
- Common Options:
  - Terminate by name: pkill process-name
  - Forceful kill: pkill -9 process-name
  - Search by user: pkill -u username process-name
- Examples:
  - Terminate all instances of firefox: pkill firefox
  - Forcefully kill nginx: pkill -9 nginx

Basic Linux Commands: Service Management

killall

- Description: Kills all processes with a specific name.
- Syntax: killall [options] name
- Common Options:
  - Terminate by name: killall process-name
  - Forceful kill: killall -9 process-name
  - Kill by user: killall -u username process-name
- Examples:
  - Terminate all instances of apache2: killall apache2
  - Forcefully kill java: killall -9 java

top

- Description: Displays real-time information about system processes and resource usage.
- Syntax: top
- Common Features:
  - Sort by CPU usage: Default view
  - Interactive commands:
    - Kill process: Press k, then enter PID
    - Sort by memory: Press M
    - Change update interval: Press d, then enter seconds
- Examples:
  - Monitor system performance: top

Basic Linux Commands: Hot Keys

<Ctrl> + <C>

- Description: Sends an interrupt signal (SIGINT) to the currently running process.
- Function: Terminates the process or command currently being executed in the terminal.
- Common Use Cases:
  - Abort a running command: ping, find, etc.
  - Stop a script or long-running process

<Ctrl> + <D>

Content:

- Description: Sends an end-of-file (EOF) signal to the terminal.
- Function: Logs out of the current shell session or indicates the end of input.
- Common Use Cases:
  - Logout of a terminal session: bash, sh, etc.
  - End input in programs: cat, more, less
- Examples:
  - Logging out of a shell: Type exit and press <Ctrl> + <D>
  - Ending input in cat: Start cat, type content, and press <Ctrl> + <D> to end input

Linux File System Overview: File System Structure

Root Directory: /

- The top-level directory in the Linux file system hierarchy.

Key Directories:

- /home - User home directories
- /etc - Configuration files
- /var - Variable files (logs, databases)
- /usr - User binaries and libraries
- /tmp - Temporary files

Path Types:

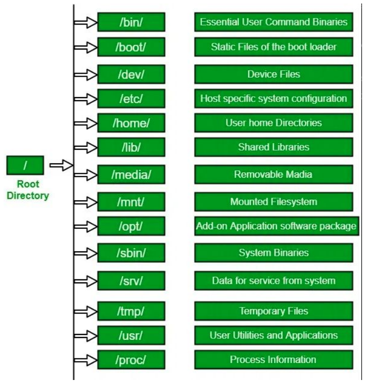
- Absolute Paths: Full path from root,
  - e.g., /home/user/documents
- Relative Paths: Path relative to current directory,
  - e.g., ../documents

Linux File System Overview: Filesystem Hierarchy Standard (FHS)

Filesystem Hierarchy Standard (FHS):

- FHS defines the directory structure and directory contents in Unix-like operating systems.
- Key Directories:

  /bin, /sbin, /lib, /mnt, /media



Popular Linux Distributions

Ubuntu:

- “User-friendly, great for beginners and general use.”
- Example: Desktop and server versions

Fedora:

- “Cutting-edge features, used for testing and development.”
- Example: Desktop and server versions

Debian:

- “Stable and versatile, used for servers and desktops.”
- Example: Stable release cycle

CentOS:

- “Enterprise-focused, based on Red Hat Enterprise Linux.”
- Example: Server environment

ubuntu fedora debian

Package Management

Debian Package Management:

apt Commands:

- "apt-get update - Updates package lists."
- "apt-get install package - Installs a package."

Red Hat Package Management:

Yum commands

- yum and dnf Commands:
  - "yum install package - Installs a package (older systems)."
  - "dnf install package - Installs a package (newer systems)."

Popular File Systems

- FAT: File Allocation Table
- NTFS: New Technology File System
- HFS: Hierarchical File System
- Ext: Extended File System

FAT: (FAT12, FAT16, FAT32).

NTFS

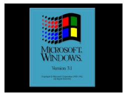
HFS

ext

Popular File Systems

FAT: File Allocation Table

- Windows (1980).
- Uses Tables to allocate files and folders.
- Originally to handle small files systems.
- Variants: FAT12 ('80), FAT16 ('84), FAT32 ('96).



NTFS: New technology File System

- Windows NT (1993).
- No file size limit, no partition limit.
- **Journaling**: record metadata and its changes tin volume or partition.
- **Transactions**: enables recreation, rename, delete of files with no impact on other files.

Popular File Systems

HFS, HFS Plus: Hierarchical File System

APFS('17): Apple File System

- MacOs by Apple ('85).
- Hierarchy of files and folder. Replaced Macintosh FS (MFS),
- Initially for floppy, HDD,m and CD-Rom.
- APFS starting macOS Sierra and later.

EXT: Extended File System

- UNIX and Linux (1992).
- Multiple versions/variants: ext2, ext3, ext4.
- Enhancement of file sizes.
- **Journaling**: record metadata and its changes tin volume or partition.
- **Transactions**: enables recreation, rename, delete of files with no impact on other files.

UNIX vs LINUX

Unix

Origin: Developed in the 1960s at AT&T's Bell Labs.

Ownership: Proprietary, with various commercial versions (e.g., AIX, HP-UX, Solaris).

Licensing: Commercial licenses.

Usage: Primarily used in enterprise and academic environments.

UNIX®
An Open Group Standard

Linux

Origin: Created by Linus Torvalds in 1991 as a free and open-source alternative to Unix.

Ownership: Open-source, with contributions from the global community.

Licensing: GNU General Public License (GPL).

Usage: Widely used in servers, desktops, and embedded systems.

Recommended Certificates

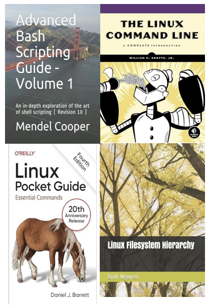
CompTIA Linux+

Red Hat Certified System Administrator

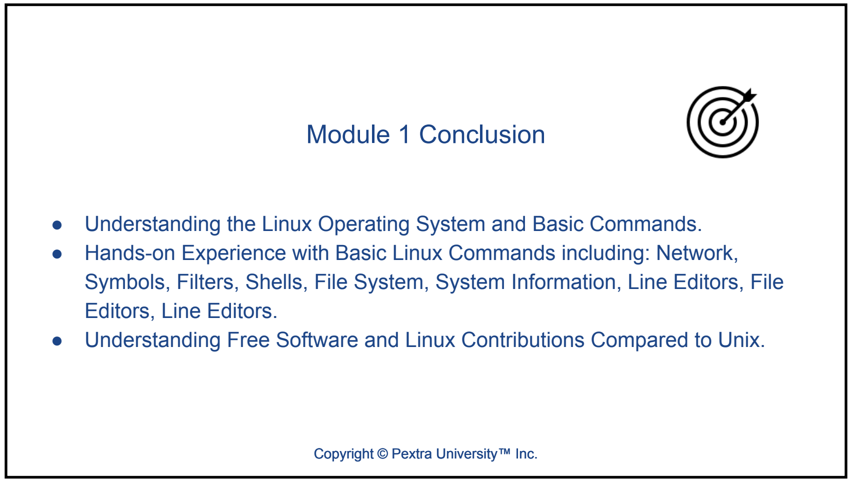
(RHCSA)

Recommended Readings

1. "The Linux Command Line: A Complete Introduction" by William E. Shotts Jr.
2. "Linux Pocket Guide" by Daniel J. Barrett
   - A handy reference for quick look-ups and command usage.
3. "Linux Filesystem Hierarchy" (Online Resource)
   - Official documentation about the filesystem structure.
4. "Advanced Bash-Scripting Guide" by Mendel Cooper
   - For deeper exploration into scripting and automation.


Advanced Bash Scripting Guide - Volume 1
Mendel Cooper
THE LINUX COMMAND LINE
Linux Pocket Guide
Linux Filesystem Hierarchy

Don't Drink & Root



Module 1 Conclusion

- Understanding the Linux Operating System and Basic Commands.
- Hands-on Experience with Basic Linux Commands including: Network, Symbols, Filters, Shells, File System, System Information, Line Editors, File Editors, Line Editors.
- Understanding Free Software and Linux Contributions Compared to Unix.