Unix and Shell Scripting

By Rajath Kumar K S, 40 Hours (4 Hours/Day)

Day 1: Introduction to Unix and Basic Commands

1. Overview of Unix
    - History and Evolution
    - Unix vs. Other Operating Systems
2. Unix Architecture
    - Kernel
    - Shell
    - File System
3. Basic Unix Commands
    - Navigation (pwd, cd, ls)
    - File Operations (cat, cp, mv, rm, touch)
    - Directory Operations (mkdir, rmdir)
4. File Viewing and Manipulation
    - Viewing File Contents (less, more, head, tail)
    - Editing Files (nano, vi, emacs)
5. Mini-Projects
    - **Project 1**: Create a directory structure and navigate through it using basic commands.
    - **Project 2**: Edit a file with vi and nano, making and saving changes.

Day 2: Advanced File and Directory Management

1. File Permissions and Ownership
    - Understanding Permissions
    - Changing Permissions (chmod)
    - Changing Ownership (chown, chgrp)

2. Advanced File Operations
    - Symbolic and Hard Links (ln)
    - Finding Files (find, locate)
    - Comparing Files (diff, cmp)
3. Working with Archives
    - Creating and Extracting Archives (tar, zip, unzip)
4. Hands-on Exercises
    - Managing file permissions, ownership, and archives
5. Mini-Projects
    - **Project 1**: Write a script to set permissions and ownership for a given directory structure.
    - **Project 2**: Create and extract an archive of a directory and verify its contents.

Day 3: Text Processing and Regular Expressions

1. Text Processing Commands
    - Searching with grep, egrep
    - Stream Editor sed
    - Text Processing with awk
2. Regular Expressions
    - Basics of Regular Expressions
    - Using Regular Expressions with grep, sed, and awk
3. Advanced Text Processing
    - Sorting (sort)
    - Text Manipulation (cut, paste, uniq, tr)
4. Hands-on Exercises
    - Complex text processing tasks using grep, sed, and awk
5. Mini-Projects
    - **Project 1**: Write a script to search for a pattern in a set of files and summarize the results.
    - **Project 2**: Create a script to process a log file, extract specific information, and generate a report.

Day 4: Introduction to Shell Scripting

1. Introduction to Shell
    - Types of Shells (Bash, Zsh, etc.)
    - Shell Features and Capabilities
2. Basic Shell Scripting
    - Writing and Executing Shell Scripts
    - Variables and Constants
    - Comments and Documentation
3. Hands-on Exercises
    - Creating and running basic shell scripts
4. Mini-Projects
    - **Project 1**: Write a script to automate file backup with a timestamp.
    - **Project 2**: Create a script to check disk space and send an alert if it exceeds a threshold.

Day 5: Control Structures and Functions in Shell Scripts

1. Control Structures
    - Conditional Statements (if, else, elif)
    - Looping Constructs (for, while, until)
    - Case Statements (case)
2. Functions in Shell Scripts
    - Defining and Calling Functions
    - Scope of Variables in Functions
    - Using Functions for Code Reusability
3. Debugging Shell Scripts
    - Debugging Techniques (set -x, set +x)
    - Common Errors and Troubleshooting
4. Hands-on Exercises
    - Implementing control structures and functions in scripts

5. Mini-Projects
    - **Project 1**: Write a script to calculate factorial of a number using loops and conditionals.
    - **Project 2**: Create a script to sort a list of files by size and move them to respective directories based on size ranges.

Day 6: Working with Processes and Job Control

1. Process Management
    - Understanding Processes
    - Viewing Processes (ps, top, htop)
    - Managing Processes (kill, pkill, xkill)
2. Job Control
    - Background and Foreground Jobs
    - Using bg, fg, jobs
    - Scheduling Jobs (at, batch)
3. Process Communication
    - Inter-process Communication (Pipes, Named Pipes)
4. Hands-on Exercises
    - Process and job management tasks
5. Mini-Projects
    - **Project 1**: Write a script to monitor CPU usage and terminate high-usage processes.
    - **Project 2**: Create a script to schedule a job that backs up a directory at a specified time.

Day 7: Advanced Shell Scripting Techniques

1. Input/Output Redirection
    - Standard Input/Output
    - Redirecting Output (>, >>)
    - Redirecting Input (<)
    - Piping (|)

2. File Descriptors
    - Understanding File Descriptors
    - Redirecting File Descriptors (2>, &>)
3. Using Shell Built-ins
    - Common Shell Built-ins (echo, read, printf)
4. Script Optimization
    - Writing Efficient Scripts
    - Best Practices and Coding Standards
5. Hands-on Exercises
    - Advanced I/O redirection and scripting techniques
6. Mini-Projects
    - **Project 1**: Write a script to redirect error messages to a log file while performing file operations.
    - **Project 2**: Create a script that processes user input and performs calculations based on the input.

Day 8: Working with the Unix File System and Networking

1. Unix File System
    - File System Hierarchy
    - Mounting and Unmounting File Systems
    - File System Types and Attributes
2. Disk Usage and Management
    - Checking Disk Space (df, du)
    - Disk Quotas and Management
    - Disk Utilities (fsck, mkfs, mount, umount)
3. Basic Networking Commands
    - ping, netstat, ifconfig, traceroute
    - Transferring Files (scp, rsync)
    - Network Configuration and Troubleshooting
4. Hands-on Exercises
    - File system navigation, disk space management, and networking tasks

5. Mini-Projects
    - **Project 1**: Write a script to monitor disk usage and clean up unnecessary files automatically.
    - **Project 2**: Create a script to transfer files securely between two systems and verify the integrity of the transferred files.

Day 9: Advanced Networking and Shell Scripting for System Administration

1. Advanced Networking Commands
    - Network Interface Configuration (ifconfig, ip)
    - Network Monitoring and Analysis (netstat, tcpdump)
2. Security and Access Control
    - SSH Configuration and Usage
    - Managing User Accounts and Groups (useradd, usermod, userdel)
    - Configuring Firewalls (iptables, ufw)
3. Automation with Shell Scripts
    - Automated Backups
    - Log Rotation
    - System Monitoring Scripts
4. Advanced Scripting Projects
    - Real-World Scenarios and Projects
5. Hands-on Exercises
    - Network configuration, security, and automation scripts
6. Mini-Projects
    - **Project 1**: Write a script to automate user account creation and setup with predefined configurations.
    - **Project 2**: Create a script to set up a basic firewall configuration and monitor network traffic.

Day 10: Putting It All Together and Best Practices

1. Review and Q&A
   - Recap of Key Concepts and Techniques
   - Open Q&A Session
2. Best Practices in Shell Scripting
   - Writing Maintainable Code
   - Documentation and Comments
   - Version Control with Git
3. Advanced Shell Scripting Topics
   - Using cron for Scheduling Tasks
   - Error Handling and Exit Codes
   - Using Libraries and Modules
4. Capstone Project
   - Comprehensive Project Incorporating All Learned Concepts
5. Additional Resources
   - Recommended Reading and Tools
   - Community and Support Channels
6. Mini-Projects
   - **Project 1**: Write a comprehensive backup and restore script that includes logging and error handling.
   - **Project 2**: Create a script to automate system updates and send a report of the update status.

Advanced Projects (Throughout the Course)

1. **Project 1**: Develop a complete system monitoring script that checks CPU usage, memory usage, disk space, and running processes, and sends alerts if any thresholds are exceeded.
2. **Project 2**: Create a deployment script that sets up a web server, configures necessary services, and deploys a sample web application.

3. **Project 3**: Build an automation script for managing and rotating logs, including archiving old logs and cleaning up outdated files.
4. **Project 4**: Develop a network monitoring tool that uses tcpdump and other network commands to capture and analyze network traffic, and generate reports.
5. **Project 5**: Create an advanced data processing pipeline using shell scripting to automate the ingestion, processing, and output of large datasets, incorporating tools like awk, sed, and custom scripts.
6. **Project 6**: Implement a comprehensive user management system that automates user account creation, password resets, and permissions management across multiple servers.
7. **Project 7**: Write a script that interfaces with a REST API to fetch, process, and store data, demonstrating integration between shell scripts and web services.
8. **Project 8**: Develop a robust backup and restore system for a database, including features for incremental backups, integrity checks, and automated scheduling using cron.
9. **Project 9**: Build a deployment script for a multi-tier application, handling setup for database, application server, and web server, including configuration and security hardening.
10. **Project 10**: Create a comprehensive disaster recovery script that performs system health checks, takes snapshots of critical data, and provides recovery procedures.

Hardware Requirements

1. **Workstations**:
    - **Processor**: Minimum Intel Core i5 or equivalent.
    - **RAM**: At least 4 GB (8 GB recommended for smooth performance).
    - **Storage**: Minimum 250 GB HDD or SSD (SSD recommended for faster performance).
    - **Network**: Reliable Ethernet or Wi-Fi connection

Software Requirements

1. **Operating System**:
    - **Preferred**: Unix-based OS (e.g., Ubuntu, CentOS, or any other Linux distribution).
    - **Alternative**: macOS or Windows with a Unix-like environment (e.g., Cygwin, WSL).
2. **Shell**:
    - **Default**: Bash (Bourne Again Shell).
    - **Alternatives**: Zsh, Ksh, or any other POSIX-compliant shell.
3. **Editors**:
    - **Basic Editors**: nano, vi (or vim), emacs.
    - **IDE**: Optional, for advanced users who prefer using an Integrated Development Environment (e.g Visual Studio Code with shell scripting plugins).

Additional Requirements

1. **Internet Access**:
    - Reliable high-speed internet for downloading software, accessing online resources, and collaborating on projects.
2. **Accounts and Permissions**:
    - User accounts with necessary permissions for installing software and accessing system configurations