Linux Essentials and Advanced Topics

1. Basics of Navigating the File System

Commands:

- **ls**: Lists the contents of the current directory.
- **cd**: Changes the current directory. For example, `cd Documents` moves to the "Documents" directory.
- **pwd**: Prints the current working directory, showing you where you are in the file system.

2. File Manipulation Commands

Commands:

- **mkdir**: Creates a new directory. For example, `mkdir my_folder` creates a directory named "my_folder".
- **touch**: Creates a new empty file or updates the timestamp of an existing file. For example, `touch my_file.txt` creates a new text file named "my_file.txt".
- **rm**: Removes files or directories. Be cautious with this command, as it permanently deletes files. For example, `rm my_file.txt` deletes the file "my_file.txt".
- **cp**: Copies files or directories. For example, `cp file1.txt file2.txt` copies "file1.txt" to "file2.txt".
- **mv**: Moves or renames files or directories. For example, `mv old_file.txt new_file.txt` renames "old_file.txt" to "new_file.txt", and `mv file.txt directory/` moves "file.txt" into the "directory" directory.

3. Basic Text Editing

Commands:

- **nano**: A simple and user-friendly text editor. Use `nano filename` to open a file in nano for editing. It provides on-screen instructions for basic operations.
- **vim**: A powerful and customizable text editor with a steeper learning curve. Use `vim filename` to open a file in vim. Press `i` to enter insert mode for editing, `Esc` to exit insert mode, and `:wq` to save and exit.

4. User Management

Adding Users:

- **useradd**: Adds a new user account to the system. Syntax: `useradd username`.
- **adduser**: A user-friendly interface for adding users, often with additional configuration options.

Deleting Users:

- **userdel**: Deletes a user account from the system. Syntax: `userdel username`.

- **deluser**: A user-friendly interface for deleting users, often handling additional cleanup tasks.

Modifying User Attributes:

- **usermod**: Modifies user account attributes, such as username, home directory, or group membership. Syntax: `usermod options username`.

Changing User Passwords:

- **passwd**: Allows users to change their passwords. As an administrator, you can use it to change another user's password by typing `passwd username`.

Viewing User Information:

- **id**: Displays user and group IDs, as well as additional information about a specified user. Syntax: `id username`.
- **finger**: Provides detailed user information, including login name, real name, terminal, and more. Syntax: `finger username`.

Switching Users:

- **su**: Allows you to switch to another user account or execute commands as another user. Syntax: `su username`.

Listing Users:

- **who**: Displays information about users who are currently logged in.
- **w**: Provides detailed information about currently logged-in users, including what they're doing.

5. System Information

Displaying Basic System Information:

- **uname**: Displays system information such as kernel name, network node hostname, kernel release, kernel version, machine hardware name, and processor type. For example, `uname -a` displays all available system information.

Viewing System Hardware Information:

- **lscpu**: Provides information about the CPU architecture and processor details.
- **lshw**: Lists detailed hardware configuration, including memory, processor, disk, and network information. Requires root privileges or sudo access.
- **lspci**: Shows information about PCI buses and connected devices.
- **lsusb**: Displays information about USB buses and connected devices.
- **lsblk**: Lists block devices, such as hard drives and partitions, along with their mount points.

Monitoring System Performance:

- **top**: Displays dynamic real-time information about system processes, CPU usage, memory usage, and more.
- **htop**: An interactive process viewer that provides an overview of system resources and allows for easy process management.

Checking System Memory Usage:

- **free**: Displays the amount of free and used memory in the system, including total, used, and free memory, as well as buffers and cache.
- **vmstat**: Reports information about processes, memory, paging, block IO, traps, and CPU activity.

Checking Disk Usage:

- **df**: Displays disk space usage for all mounted filesystems.
- **du**: Estimates disk usage for directories and files.

Viewing Network Information:

- **ifconfig** or **ip addr**: Shows network interface information, including IP addresses, MAC addresses, and network configuration.
- **netstat**: Displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
- **ss**: Another utility to investigate sockets, displaying more detailed information than netstat.

6. Package Management

Debian/Ubuntu-based Systems (using APT):

- **apt update**: Updates the local package index to reflect the latest changes made in the repositories.
- **apt upgrade**: Upgrades all installed packages to their latest versions.
- **apt install** : Installs the specified package.
- **apt remove** : Removes the specified package, along with its configuration files.
- **apt purge** : Completely removes the specified package, including its configuration files.
- **apt search** : Searches for packages matching the specified keyword.
- **apt list --installed**: Lists all installed packages.

Red Hat/CentOS-based Systems (using YUM or DNF):

- **yum update** or **dnf update**: Updates all installed packages to their latest versions.
- **yum install** or **dnf install** : Installs the specified package.
- **yum remove** or **dnf remove** : Removes the specified package.
- **yum search** or **dnf search** : Searches for packages matching the specified keyword.
- **yum list installed** or **dnf list installed**: Lists all installed packages.

Common Package Management Commands for All Systems:

- **apt-cache search** : Searches for packages matching the specified keyword in Debian/Ubuntu-based systems.
- **rpm -qa**: Lists all installed packages in Red Hat/CentOS-based systems.

- **dpkg -l**: Lists all installed packages in Debian/Ubuntu-based systems.

7. File Management

Commands:

- **find**: Searches for files and directories in a directory hierarchy based on various criteria such as name, size, or permissions.
- **grep**: Searches for patterns in files or standard input. It's commonly used to filter lines containing a specific pattern.
- **awk**: A versatile text processing tool that operates on lines of input and can perform actions based on patterns.
- **sed**: A stream editor used to perform text transformations on an input stream. It's often used for search and replace operations.
- **tar**: Archives files into a single file (often called a "tarball") and optionally compresses them.
- **gzip**: Compresses files using the gzip compression algorithm. It replaces the original file with a compressed version.
- **zip**: Compresses files into a zip archive, which can include multiple files and directories.

8. Networking

Network Configuration:

- **ifconfig** or **ip addr**: Displays or configures network interfaces, including IP addresses, MAC addresses, and network configuration.
- **iwconfig**: Configures wireless network interfaces.

Network Connectivity:

- **ping**: Tests connectivity to a remote host by sending ICMP echo request packets.
- **traceroute** or **traceroute6**: Traces the route packets take to reach a destination host.
- **mtr**: Combines the functionality of ping and traceroute to provide real-time network diagnostics.
- **netcat** or **nc**: Reads and writes data across network connections, often used for debugging and network exploration.
- **telnet**: Allows users to communicate with remote systems using the Telnet protocol.
- **ssh**: Securely connects to a remote system using the SSH protocol.

DNS Configuration and Resolution:

- **dig**: A versatile DNS lookup utility for querying DNS servers and retrieving DNS records.
- **nslookup**: Another DNS lookup utility for querying DNS servers and resolving domain names.

Network Services and Ports:

- **netstat**: Displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
- **ss**: Another utility to investigate sockets, providing more detailed information than netstat.
- **nmap**: A powerful network scanning tool for discovering hosts and services on a network.

Network Diagnostics:

- **arp**: Displays or manipulates the IP-to-MAC address translation tables.
- **tcpdump**: Captures and analyzes network packets in real-time.
- **wireshark**: A graphical network protocol analyzer that allows for deep inspection of network traffic.

Firewall and Security:

- **iptables**: Controls packet filtering, network address translation (NAT), and other firewall-related tasks.
- **ufw**: A user-friendly interface for managing iptables firewall rules.
- **fail2ban**: Monitors system logs and dynamically blocks IP addresses exhibiting malicious behavior.
- **selinux**: Security-Enhanced Linux, a set of kernel modifications and tools for enforcing security policies.

9. Automation and Scripting

Basic Shell Scripting:

- **#!/bin/bash**: The shebang line at the beginning of a script that indicates the interpreter to be used.
- **Variables**: Used to store values in a script. For example, `my_variable="Hello"` assigns the value "Hello" to the variable `my_variable`.
- **Conditional Statements**: Use `if`, `else`, and `elif` to perform conditional operations based on certain conditions.
- **Loops**: Use `for`, `while`, and `until` loops to iterate over a series of commands or values.
- **Functions**: Group commands into reusable blocks of code by defining functions.

Advanced Scripting Techniques:

- **Case Statements**: Use `case` to execute different commands based on the value of a variable.
- **Here Documents**: Use `<<EOF` to create a multiline string or command block within a script.
- **Cron Jobs**: Schedule scripts or commands to run at specific intervals using the cron daemon.

10. Security

File Permissions:

- **chmod**: Changes the permissions of files or directories. Use `chmod 755 filename` to set permissions where the owner has full control, and others have read and execute permissions.
- **chown**: Changes the ownership of files or directories. For example, `chown user:group filename` changes the owner to "user" and the group to "group".
- **umask**: Sets the default file creation permissions.

User Authentication:

- **passwd**: Changes the user's password.
- **ssh-keygen**: Generates SSH keys for secure authentication without passwords.
- **sudo**: Grants temporary superuser privileges to a regular user for executing commands requiring root access.

Encryption and Decryption:

- **gpg**: Encrypts and signs files using the GNU Privacy Guard.
- **openssl**: A toolkit for secure communications that supports encryption, decryption, and certificate management.

Security Auditing and Monitoring:

- **logwatch**: Summarizes and reports system log files for potential security issues.
- **chkrootkit**: Scans for rootkits, a type of malware that hides its presence on a system.
- **rkhunter**: Scans for rootkits, backdoors, and possible local exploits.

Firewall Configuration:

- **iptables**: Configures the Linux kernel's built-in firewall.
- **ufw**: A user-friendly front-end for managing iptables firewall rules.
- **firewalld**: A dynamic firewall management tool that provides a D-Bus interface for managing firewall rules.

11. Process Management

11.1 Viewing Processes

- **ps**: Displays information about active processes. Common options include:
  - `ps -aux`: A detailed list of all processes.
  - `ps -ef`: A full listing of processes.
- **top**: Provides a dynamic real-time view of system processes, CPU usage, and memory usage.
- **htop**: An interactive process viewer that provides an overview of system resources and allows for easy process management.
- **pgrep**: Searches for processes based on name or other attributes and prints their process IDs.
- **pstree**: Displays a tree diagram of processes, showing their hierarchical relationship.

11.2 Killing Processes

- **kill**: Terminates a process by sending a signal to it. By default, it sends the TERM signal, but other signals like KILL or HUP can be specified.
- **killall**: Terminates processes by name rather than PID.
- **pkill**: Similar to killall, but more versatile as it allows specifying processes by name or other attributes using regular expressions.
- **xkill**: A graphical utility that allows users to kill a window or process by clicking on it.

11.3 Background and Foreground Processes

- **bg**: Puts a stopped or backgrounded process into the background.
- **fg**: Brings a backgrounded process to the foreground.
- **jobs**: Lists active jobs (background processes) associated with the current shell.

11.4 Process Priority and Control

- **nice**: Launches a process with a specified priority level.
- **renice**: Changes the priority of an existing process.
- **ionice**: Sets the I/O scheduling priority for a process.

11.5 Monitoring and Debugging

- **strace**: Traces system calls and signals made by a process, helpful for debugging.
- **lsof**: Lists open files and the processes that opened them, useful for troubleshooting.
- **pidof**: Returns the process ID of a running program.
- **pgrep**: Searches for processes based on name and other attributes and prints their process IDs.
- **killall**: Terminates processes by name rather than PID.

11.6 System Resource Usage

- **free**: Displays the amount of free and used memory in the system.
- **vmstat**: Reports information about processes, memory, paging, block IO, traps, and CPU activity.
- **sar**: Collects, reports, and saves system activity information, including CPU, memory, disk, and network usage.

12. System Administration

12.1 Managing System Services

- **systemctl**: Controls systemd services, including starting, stopping, restarting, enabling, disabling, and viewing service status. For example:
  - Start a service: `sudo systemctl start serviceName`
  - Stop a service: `sudo systemctl stop serviceName`
  - Restart a service: `sudo systemctl restart serviceName`
  - Enable a service: `sudo systemctl enable serviceName`
  - Disable a service: `sudo systemctl disable serviceName`
  - Check service status: `sudo systemctl status serviceName`
- **service**: A command-line tool for managing system services, commonly used in traditional SysVinit systems. It can start, stop, restart, reload, enable, and disable services. For example:
  - Start a service: `sudo service serviceName start`
  - Stop a service: `sudo service serviceName stop`
  - Restart a service: `sudo service serviceName restart`
  - Reload a service: `sudo service serviceName reload`
  - Enable a service: `sudo service serviceName enable`
  - Disable a service: `sudo service serviceName disable`

12.2 System Backup and Restore

12.2.1 Backup

- **tar**: Archives files into a single file (tarball) and optionally compresses them. For example:
  - Create a tarball: `tar -cvf backup.tar /path/to/backup`
- **rsync**: Syncs files and directories between different locations, often used for backup purposes. For example:
  - Backup files: `rsync -av /source/directory/ /destination/directory/`

12.2.2 Restore

- **tar**: Extracts files from a tarball. For example:
  - Extract files: `tar -xvf backup.tar -C /path/to/restore`
- **rsync**: Restores files from a backup location. For example:
  - Restore files: `rsync -av /backup/source/ /restore/destination/`

12.3 Disk Management

- **fdisk**: A command-line utility for disk partitioning. It allows creating, deleting, and managing disk partitions. For example:
  - Start fdisk: `sudo fdisk /dev/sdX`
- **mkfs**: Creates a filesystem on a disk partition. It is used to format partitions with various filesystem types. For example:
  - Format partition with ext4: `sudo mkfs.ext4 /dev/sdX1`
- **mount**: Mounts filesystems to the directory tree, making them accessible. For example:
  - Mount a filesystem: `sudo mount /dev/sdX1 /mnt`
- **df**: Displays disk space usage for all mounted filesystems. For example:
  - Show disk space usage: `df -h`

13. Advanced Topics

13.1 Shell Scripting (Bash Scripting Basics)

Bash scripting allows users to automate tasks by writing scripts that execute commands and perform operations. Basic elements include variables, loops, conditionals, functions, and command substitution.

Example:

```
#!/bin/bash
# This is a simple Bash script

# Define variables
greeting="Hello, world!"

# Print greeting
echo $greeting
```

13.2 Remote Administration (SSH, rsync)

- **SSH**: Secure Shell (SSH) allows users to securely access and manage remote systems over a network. Example:
  - `ssh username@hostname`
- **rsync**: A utility for efficiently syncing files and directories between systems. Example:
  - `rsync -av /local/path/ username@remotehost:/remote/path/`

13.3 System Security (Firewall Configuration, SELinux, AppArmor)

13.3.1 Firewall Configuration

- Tools like **iptables** (legacy) or **firewalld** (modern) are used to configure firewall rules.

13.3.2 SELinux (Security-Enhanced Linux)

- Provides access control security policies, including mandatory access controls (MAC). Commands include:
  - `sestatus`: Check the status.
  - `setenforce`: Change the enforcement mode.

13.3.3 AppArmor

- A mandatory access control framework for restricting programs' capabilities. Commands include:
  - `apparmor_status`: Check the status.
  - `aa-enforce`: Enforce policies.

13.4 Virtualization and Containerization (VirtualBox, Docker)

13.4.1 VirtualBox

- Allows users to create and manage virtual machines (VMs) on a host system. Provides both a graphical user interface (GUI) and command-line interface (CLI) tools.

13.4.2 Docker

- A platform for developing, shipping, and running applications in containers. Commands include:
  - `docker run`: Run containers.
  - `docker build`: Build images.
  - `docker-compose`: Manage multi-container applications.