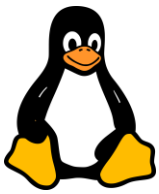
# Linux Commands



**Disclaimer**

This document is intended primarily for internal use. JP Robotic LLC accepts no responsibility for errors, omissions, or any damages resulting from its use. Use at your own risk.

## Table of Contents

## 1. keyboard Shortcuts

**Ctrl + C** # Forcefully terminate the active process in the terminal.

**Ctrl + Z** # Pause the current task and push it to the background. Resume with fg (foreground) or bg (background).

**Ctrl + W** # Delete the word before the cursor and copy it to the clipboard.

**Ctrl + U** # Cut everything from the cursor to the beginning of the line and copy to clipboard.

**Ctrl + K** # Cut everything from the cursor to the end of the line and copy to clipboard.

**Ctrl + Y** # Paste the most recently cut text from the clipboard.

**Ctrl + R** # Search the command history for a match to the typed characters.

**Ctrl + O** # Execute the command found by Ctrl + R search.

**Ctrl + G** # Exit command search mode without executing any command.

**Clear** # Clear all text displayed in the terminal window.

**!!** # Repeat and run the last command.

**Exit** # Close the terminal session and log out.

## 2. Basic Shell Commands

**alias [aliasname]='[command]'** # Create a shortcut for a command.

**watch -n [interval-inseconds] [command]** # Execute a command repeatedly at a specified time interval.

**sleep [time-interval] && [command]** # Delay the execution of a command for a set period.

**at [hh:mm]** # Schedule a task to run at a specific time (press ctrl+d to confirm and exit prompt).

**man [command]** # Display the manual page for a given command.

**history** # List the previously executed commands in the current terminal session.

## 3. System Information

**uname -a** # Display comprehensive system information, including kernel version, machine, and OS details.

**uname -r** # Show the kernel release version for insight into the currently running kernel.

**cat /etc/redhat-release** # Display the version and details of the installed Red Hat distribution.

**lsb_release -a** # Provide detailed information about the installed Ubuntu version, including codename and release.

**uptime** # Display the system's running duration, along with the current load average over 1, 5, and 15 minutes.

**hostname** # Present the system's network name.

**hostname -I** # List all local IP addresses assigned to the system.

**last reboot** # Display the history of system reboots with timestamps.

**date** # Show the current system date and time with detailed formatting.

**cal** # Present the current month's calendar with the current day highlighted. w # Display who is currently logged into the system along with their activity.

**whoami** # Reveal the current user you are logged in as.

**timedatectl** # Display or adjust the system clock settings and time zone information.

**finger [user_name]** # Show detailed information about a specified user, such as login details and full name.

**ulimit [flags] [limit]** # Inspect or set resource limitations for the current user session.

**shutdown [hh]** # Schedule a system shutdown at a specified time.

**shutdown now** # Immediately shut down the system for maintenance or power off.

**modprobe [module_name]** # Load or add a kernel module dynamically to extend system capabilities.

**dmesg** # Display kernel ring buffer messages, typically used for viewing boot-time logs and system diagnostics.

## 4. Hardware Information and Management

**dmesg** # Display kernel ring buffer messages, useful for troubleshooting hardware and system issues.

**cat /proc/cpuinfo** # Present detailed information about the CPU, including model, cores, and speed.

**cat /proc/meminfo** # View comprehensive details about the system's memory usage and status.

**free -h** # Display available and used memory in a human-readable format (use -m for MB and -g for GB).

**lspci -tv** # Show PCI devices in a hierarchical, tree-like format for easy visualization of device connections.

**lsusb -tv** # List USB devices connected to the system in a tree-structured view.

**dmidecode** # Extract and display DMI/SMBIOS data, providing hardware information directly from the BIOS.

**hdparm -i /dev/sda** # Present technical specifications and information about disk sda.

**hdparm -tT /dev/sda** # Perform a read speed benchmark test on disk sda for performance evaluation.

**badblocks -s /dev/sda** # Scan disk sda for unreadable or defective blocks, displaying progress during the test.

**lshw** # Enumerate detailed hardware configuration, including CPU, memory, storage, and network interface information.

**lscpu** # Display a brief overview of the CPU architecture and specifications.

**lsblk** # Provide a structured list of block devices, such as storage drives and partitions.

**lspci -tv** # Display PCI device details in a tree-like format for a clear view of connections.

**lsusb -tv** # Present USB devices in a structured, tree-like format.

**lshw** # Display detailed hardware configuration, offering insights into CPU, memory, and network components.

**cat /proc/cpuinfo** # Show detailed information about the CPU, including cache size and flags.

**cat /proc/meminfo** # Examine detailed information on system memory allocation and availability.

**cat /proc/mounts** # List all currently mounted file systems with relevant details.

**free -h** # Show memory usage in a human-friendly format.

**sudo dmidecode** # Run as superuser to access and display hardware details directly from BIOS/firmware.

**hdparm -i /dev/[device_name]** # Display disk-specific information for the specified device.

**hdparm -tT /dev/[device_name]** # Run a read performance test on the specified device.

**badblocks -s /dev/[device_name]** # Check the specified device for any bad sectors, showing progress.

**fsck /dev/[device_name]** # Run a file system check and repair on an unmounted disk or partition.

## 5. Performance Monitoring and Statistics

**top** # Display a dynamic and real-time overview of the most resource-intensive processes running on the system, allowing process management.

**htop** # Enhanced, interactive process viewer with a more user-friendly interface compared to top, featuring color-coded metrics and easier navigation.

**mpstat 1** # Continuously display processor-related statistics at 1-second intervals for monitoring CPU performance.

**vmstat 1** # Report virtual memory statistics and system performance metrics at 1-second intervals for resource monitoring.

**iostat 1** # Display CPU and input/output statistics every second to monitor device and system performance.

**tail -100 /var/log/messages** # Output the last 100 lines of the system log file for reviewing recent system activity (use /var/log/syslog for Debian-based systems).

**tcpdump -i eth0** # Capture and display all network packets on the eth0 interface for traffic analysis.

**tcpdump -i eth0 'port 80'** # Monitor and display all HTTP traffic on the eth0 interface by filtering packets on port 80.

**lsof** # List all open files on the system, including files, network connections, and devices.

**lsof -u user** # Display all open files associated with a specific user, useful for monitoring user activity.

**watch df -h** # Run df -h repeatedly, displaying disk space usage with updates at regular intervals.

**mpstat** # Report detailed CPU usage statistics, showing per-processor data for comprehensive performance analysis.

**pidstat** # Provide real-time statistics on currently running processes, including CPU and memory usage, for process monitoring.

## 6. Process Management

**ps** # List currently active processes running under the user's session.

**ps -ef** # Show a comprehensive list of all processes running on the system with full details.

**ps -ef | grep processname** # Search and display details of processes matching processname.

**pstree** # Display running processes in a hierarchical tree format to visualize parent-child relationships.

**pmap** # Show detailed memory map and usage for a specific process.

**top** # Display a dynamic and real-time list of all running processes, including their resource usage.

**htop** # An enhanced and interactive process viewer with a colorful interface, allowing easy process management.

**kill [process_id]** # Forcefully stop a running process by specifying its process ID (PID).

**pkill [process_name]** # Terminate processes by their name, useful for stopping multiple instances at once.

**killall [label]** # Kill all processes associated with a specific label or name.

**pgrep [keyword]** # Search for and display process IDs that match a given keyword.

**pidof [process_name]** # Show the PID of a specific running process.

**bg** # List and resume jobs that have been stopped or are running in the background.

**fg** # Bring the most recently suspended or backgrounded job to the foreground for interaction.

**fg n** # Bring the specified job number n to the foreground.

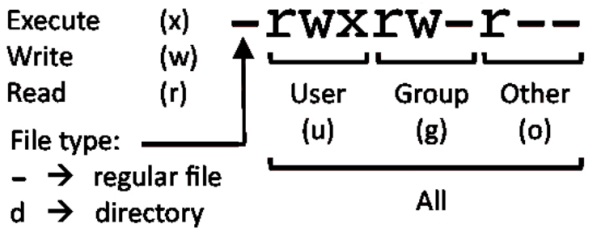
**lsof** # List open files associated with running processes, useful for identifying file usage and network connections.

**trap "[commands]" [signal]** # Execute specified commands when a given signal (e.g., termination) is caught in a script.

**wait** # Suspend execution in the terminal or a script until a specified background process completes.

**nohup [command] &** # Run a command in the background while ignoring hangup signals, ensuring it continues running after logout.

**program &** # Start a program and run it in the background, freeing up the terminal for other commands.




| U | G | W | |
|---|---|---|---|
| rwx | rwx | rwx | chmod 777 filename |
| rwx | rwx | r-x | chmod 775 filename |
| rwx | r-x | r-x | chmod 755 filename |
| rw- | rw- | r-- | chmod 664 filename |
| rw- | r-- | r-- | chmod 644 filename |

**LEGEND**

**U = User**
**G = Group**
**W = World**

**r = Read**
**w = write**
**x = execute**
**- = no access**

**PERMISSION EXAMPLE**

**chown Joe /path/to/file** # Change the ownership of /path/to/file so that Joe becomes the file owner.

**chgrp sales /path/to/file** # Change the group ownership of /path/to/file to the group sales.

**chmod 777 [file_name]** # Assign read, write, and execute permissions to everyone for [file_name] (permissions set to rwxrwxrwx).

**chmod 755 [file_name]** # Grant full permissions (read, write, execute) to the owner, and read and execute permissions to the group and others (rwxr-xr-x).

**chmod 766 [file_name]** # Provide full permissions to the owner, and read and write permissions to the group and others (rwxrw-rw-).

**chown [user_name] [file_name]** # Change the ownership of [file_name] so that [user_name] is the new file owner.

**chown [user_name]:[group_name] [file_name]** # Modify both the owner and group ownership of [file_name] to [user_name] and [group_name], respectively.

## 7. User Information and Management

**id** # Display the user ID (UID), group ID (GID), and groups associated with the current user.

**last** # Show a list of recent user login sessions, including timestamps and session durations.

**who** # Display a list of users currently logged into the system.

**w** # Present detailed information about users currently logged in and their ongoing activities.

**groupadd test** # Create a new group named "test" for managing user permissions.

**useradd -c "Joe Snuffy" -m Joe** # Create a new user account named "Joe" with the comment "Joe Snuffy" and generate a home directory for the user.

**userdel Joe** # Remove the user account named "Joe" from the system, deleting all associated user data.

**usermod -aG sales Joe** # Append the user "Joe" to the "sales" group, granting him group permissions without affecting existing group memberships.

## 8. File and Directory Operations

**cd ..** # Navigate up one level in the directory tree, changing to the parent directory.

**cd** # Move directly to the user's $HOME directory.

**cd /etc** # Change the current working directory to /etc.

**cd ~** # Change to the $HOME directory.

**cd -** # Switch back to the previous directory you were in.

**cd [directory_path]** # Change to a specified directory path.

**alias goto='cd /etc/'** # Create an alias goto that changes the current directory to /etc.

**ls** # List the files and directories in the current working directory.

**ls -a** # List all files and directories in the current directory, including hidden ones.

**ls -l** # Display the contents of the current directory in a detailed list format, showing permissions, ownership, and more.

**ls -al** # List all files and directories in the current directory with detailed information, including permissions, ownership, and timestamps.

**pwd** # Print the current working directory's full path.

**dirs** # Display the current directory stack, showing the list of remembered directories.

**pwd** # Display the full path of the current working directory.

**mkdir directory** # Create a new directory named "directory".

**rm file** # Delete a file from the system.

**rm -r directory** # Recursively remove a directory and all its contents.

**rm -f file** # Force deletion of a file without any confirmation prompt.

**rm -rf directory** # Forcefully and recursively delete a directory and its contents without confirmation.

**cp file1 file2** # Copy the contents of file1 to a new file named file2.

**cp -r source_directory destination** # Copy the entire source_directory and its contents to destination. If destination exists, source_directory will be copied into it; otherwise, destination will be created with the contents of source_directory.

**mv file1 file2** # Move or rename file1 to file2. If file2 is an existing directory, move file1 into it.

**ln -s /path/to/file linkname** # Create a symbolic link named linkname that points to the specified file.

**touch file** # Create an empty file or update the access and modification times of an existing file.

**cat file** # Display the contents of a file in the terminal.

**less file** # Open a file for viewing, with navigation options for moving up and down.

**head file** # Display the first 10 lines of a file by default.

**tail file** # Display the last 10 lines of a file by default.

**tail -f file** # Continuously display new lines added to a file in real-time.

**cat source_file >> destination_file** # Append the contents of source_file to destination_file.

**more file_name** # Display file content page by page, allowing navigation with keyboard commands.

**nano file_name** # Open or create a file with the nano text editor for quick editing.

**vi file_name** or **vim file_name** # Open or create a file with the vi or vim text editor for more advanced editing features.

**gpg -c file_name** # Encrypt a file using symmetric encryption.

**gpg file_name.gpg** # Decrypt an encrypted .gpg file.

**wc -w file_name** # Display the number of words, lines, and bytes in a file.

**ls | xargs wc** # Count the number of lines, words, and characters in each file within a directory.

**cut -d [delimiter] file_name** # Extract and display sections of a file separated by a specified delimiter.

**[data] | cut -d [delimiter]** # Process and display a specific section of piped data based on the delimiter.

**shred -u file_name** # Overwrite a file with random data to securely erase it, and then delete it.

**diff first_file second_file** # Compare two files and display their differences line by line.

**source file_name** # Execute the commands within a file in the current shell environment.

**[command] | tee file_name >/dev/null** # Save the output of a command to a file while discarding the terminal output.

## 9. File and Text Search Utilities

**grep pattern file** # Search for occurrences of pattern within file and display matching lines.

**grep -r pattern directory** # Recursively search for pattern in all files within directory. find /home/Joe -name 'prefix'* # Locate files in /home/Joe that start with the name prefix.

**find /home -size +100M** # Search for files larger than 100 MB within the /home directory.

**whereis program** # Show the paths to the binary, source, and manual page files for the specified program.

**which program** # Display the full path to the executable that would be run if program is executed, based on the current environment.

**find [path] -name [search_pattern]** # Search for files and directories matching the specified search_pattern within the given path.

**find [path] -size [+100M]** # Locate files larger than 100 MB in the specified path.

**grep [search_pattern] [file_name]** # Search for a pattern in [file_name] and display lines containing matches.

**grep -r [search_pattern] [directory_name]** # Perform a recursive search for search_pattern across all files in [directory_name].

**locate [name]** # Quickly find paths for files or directories containing the specified name.

**which [command]** # Show the path to the executable file for [command] as found in the $PATH environment variable.

**whereis [command]** # Display the locations of the binary, source, and manual files for [command].

**awk '[search_pattern] {print $0}' [file_name]** # Print lines in [file_name] that match the search_pattern using awk.

**sed 's/[old_text]/[new_text]/' [file_name]** # Replace occurrences of old_text with new_text in [file_name].

## 10. File Transfers and Synchronization

**scp file.txt server:/tmp** # Securely copy file.txt from the local system to the /tmp folder on the remote server.

**scp server:/var/www/*.html /tmp** # Copy all .html files from the /var/www directory on server to the local /tmp folder.

**scp -r server:/var/www /tmp** # Recursively copy all files and subdirectories from the /var/www directory on server to the local /tmp folder.

**rsync -a /home /backups/** # Synchronize the entire /home directory to /backups/, preserving file attributes and structure.

**rsync -avz /home server:/backups/** # Synchronize /home to the /backups/ directory on server, with compression enabled to speed up the transfer.

**ftp host** # Connect to an FTP server at host for transferring files interactively.

**scp [source_file] [user]@[remote_host]:[destination_path]** # Securely copy [source_file] from the local system to [destination_path] on the remote host as [user].

**rsync -a [source_directory] [user]@[remote_host]:[destination_directory]** # Synchronize the contents of [source_directory] to [destination_directory] on the remote host while preserving permissions and timestamps.

**wget [link]** # Download a file or webpage from an HTTP, HTTPS, or FTP server using the specified [link].

**curl -O [link]** # Download and save a file from a server to the local system using various supported protocols.

**ftp [remote_host]** # Establish an FTP session with [remote_host] to transfer files interactively between local and remote systems.

**sftp [user]@[remote_host]** # Securely transfer files between local and remote systems using the SFTP protocol for encrypted sessions.

## 11. Disk Usage and Monitoring

**df -h** # Display the free and used disk space on all mounted filesystems in a human-readable format.

**df -i** # Show the number of free and used inodes on mounted filesystems, useful for tracking inode exhaustion.

**fdisk -l** # List all disk partitions, their sizes, and types across available drives.

**du -ah** # Display disk usage for all files and directories in a human-readable format, providing detailed space consumption.

**du -sh** # Summarize the total disk usage of the current directory in a human-readable format.

**du -a directory** # List the disk usage of all files within the specified directory, displaying their sizes.

**findmnt** # Enumerate and show detailed information about all currently mounted file systems, including mount points and options.

## 12. Networking and Connection Management

**ip a** # Display all network interfaces and their associated IP addresses for the system.

**ip addr show dev eth0** # Show detailed information about the eth0 network interface, including its IP address and status.

**ethtool eth0** # Retrieve and adjust network driver and hardware settings for the eth0 interface.

**ping host** # Send ICMP echo requests to host to check network connectivity and measure response time.

**whois domain** # Fetch WHOIS information for domain to show domain registration and ownership details.

**dig domain** # Query DNS information for domain, providing records like A, MX, and NS.

**dig -x IP_ADDRESS** # Perform a reverse DNS lookup to find the domain name associated with IP_ADDRESS.

**host domain** # Display the DNS IP address for domain and associated records.

**hostname -i** # Present the primary network address associated with the system's hostname.

**hostname -I** # Show all local IP addresses assigned to the system.

**wget http://domain.com/file** # Download a file from http://domain.com/file to the local system.

**netstat -nutlp** # List all currently listening TCP and UDP ports along with their associated programs.

**ifconfig** # Display detailed information about all network interfaces on the system.

**traceroute host** # Trace the path that packets take to reach host, displaying each hop along the way.

**tcpdump** # Capture and analyze packets on the network, useful for troubleshooting and monitoring traffic.

**ip addr show** # Display IP addresses and information for all active network interfaces.

**ip address add [IP_address]** # Assign a specific IP_address to an interface, such as eth0.

**ifconfig** # Display network interface details, including IP addresses and status.

**ping [remote_host]** # Send ICMP echo requests to [remote_host] to test connectivity.

**netstat -pnltu** # Show active listening ports and their protocols with associated processes.

**netstat -tuln** # List all listening TCP and UDP ports and the processes using them.

**whois [domain_name]** # Display detailed registration and ownership information for [domain_name].

**dig [domain_name]** # Use the dig command to show detailed DNS information for [domain_name].

**dig -x [domain_name]** # Perform a reverse DNS lookup for the specified [domain_name].

**dig -x [IP_address]** # Conduct a reverse DNS lookup for the given [IP_address] to find the associated domain.

**host [domain_name]** # Perform a DNS lookup to retrieve the IP address for [domain_name].

**hostname -I** # Display all local IP addresses of the host system.

**nslookup [domain_name]** # Query DNS servers for information about [domain_name], such as its IP address.

## 13. SSH and Remote Access

**ssh host** # Connect to host as the current local user over SSH.

**ssh user@host** # Establish an SSH connection to host as the specified user.

**ssh -p port user@host** # Connect to host as user using a custom port for the SSH session.

**ssh-keygen** # Create a new pair of SSH keys (public and private) for secure authentication.

**ssh-copy-id user@host** # Copy your public SSH key to host to enable passwordless login for user.

**ssh [user_name]@[host]** # Connect securely to a remote host as user using the SSH protocol.

**ssh [host]** # Connect to a remote host over SSH using the default port 22.

**ssh -p [port] [user_name]@[host]** # Connect to host over SSH using the specified port as user.

**ssh-keygen** # Generate a public and private SSH key pair for secure access.

**sudo service sshd start** # Start the SSH server daemon to enable incoming SSH connections.

**scp [file_name] [user_name]@[host]:[remote_path]** # Securely copy file_name from the local system to a remote system at remote_path using SSH.

**sftp [user_name]@[host]** # Initiate an interactive SFTP (Secure File Transfer Protocol) session to transfer files securely over SSH.

**telnet [host]** # Connect to host using the Telnet protocol on the default port 23 for unencrypted communication.

## 14. Archive and Compression Management

**tar cf archive.tar directory** # Generate tar named archive.tar containing directory.

**tar xf archive.tar** # Extract the contents from archive.tar.

**tar czf archive.tar.gz directory** # Generate a gzip compressed tar file name archive.tar.gz.

**tar xzf archive.tar.gz** # Extract a gzip compressed tar file.

**tar cjf archive.tar.bz2 directory** # Generate a tar file with bzip2 compression

**tar xjf archive.tar.bz2** # Extract a bzip2 compressed tar file.

**tar cf [archive.tar] [file/directory]** # Archive an existing file or directory.

**tar xf [archive.tar]** # Extract an archived file.

**tar czf [archive.tar.gz]** # Generate a .gz compressed tar archive.

**gzip [file_name]** # Compress .gz files.

**gunzip [file_name.gz]** # Decompress .gz files.

**bzip2 [file_name]** # Compress .bz2 files.

**bunzip2 [file_name.bz2]** # Decompress .bz2 files.

## 15. Package Management Overview

**yum search keyword** # Search for a package by a specific keyword to find matching software in the repository for RHEL-based systems.

**yum install package** # Install a package by its name on RHEL-based systems, resolving and downloading dependencies automatically.

**yum info package** # Display a detailed description and summary information about a package, including its version and size, for RHEL-based systems.

**rpm -i package.rpm** # Install a package from a local .rpm file named package.rpm on RHEL-based systems.

**yum remove package** # Uninstall or remove a package from the system on RHEL-based distributions, including all dependencies no longer needed.

**yum update package** # Update the specified package to its latest version for RHEL-based systems.

**tar zxvf sourcecode.tar.gz** # Extract the contents of the sourcecode.tar.gz compressed archive, creating the source code directory.

**cd sourcecode** # Change the current directory to the sourcecode directory.

**./configure** # Run the configuration script to check for system requirements and set up the build environment for the source code.

**make** # Compile the source code into executable binaries.

**make install** # Install the compiled program into the appropriate system directories.

**apt-get update** # Update the local package index to ensure the latest package information is available for Debian-based systems.

**apt-get upgrade** # Upgrade all currently installed packages to their newest available versions on Debian-based systems.

**apt-get install package** # Install a package by its name, ensuring dependencies are resolved, on Debian-based systems.

**apt-remove package** # Remove a package and its associated configuration files from the system on Debian-based distributions.

## 16. Packages Management (Debian/Ubuntu)

**sudo apt-get install [package_name]** # Install a package using the apt-get utility, resolving dependencies automatically for Debian-based systems.

**sudo apt install [package_name]** # Install a package using the more modern apt package manager, offering improved user interaction.

**apt search [keyword]** # Search for a package in the APT repositories that matches the specified keyword.

**apt list** # List all packages installed on the system or available in the repositories using APT.

**apt show [package_name]** # Display detailed information about a specific package, including its description, version, and dependencies.

**sudo dpkg -i [package_name.deb]** # Install a local .deb package file using the dpkg command.

**sudo dpkg -l** # List all packages installed on the system that were managed by dpkg, showing their names and versions.

## 17. Packages Management (Red Hat, CentOS, Fedora)

**sudo yum install [package_name]** # Install a package using the YUM package manager, automatically resolving dependencies on RHEL-based systems.

**yum search [keyword]** # Search for packages in the YUM repositories that match the provided keyword.
**yum list installed** # Display a comprehensive list of all packages currently installed with YUM on the system.
**yum info [package_name]** # Show detailed information about a specific package, including its description, version, and repository source.
**sudo dnf install [package_name]** # Install a package using the DNF package manager, a modern replacement for YUM with improved performance and dependency management.
**sudo rpm -i [package_name.rpm]** # Install a local .rpm package using the rpm command, which does not resolve dependencies automatically.

## 18. Security and Access Control

**passwd** # Change the password for the current user or specify a user to modify their password.
**sudo -i** # Switch to the root account and start a root login shell, applying root's environment variables.
**sudo -s** # Run the current shell with root privileges, without initiating a login shell.
**sudo -l** # Display a list of sudo privileges for the current user, showing allowed commands.
**visudo** # Safely edit the sudoers file to configure or modify user sudo permissions.
**getenforce** # Display the current mode of SELinux (e.g., Enforcing, Permissive, or Disabled).
**sestatus** # Show detailed information about the current state of SELinux, including mode and policy details.
**setenforce 0** # Change the current SELinux mode to Permissive, allowing policy violations but logging them (does not persist after reboot).
**setenforce 1** # Change the current SELinux mode to Enforcing, ensuring policy rules are enforced (does not persist after reboot).
**SELINUX=enforcing** # Configure SELinux to start in Enforcing mode on boot by setting this in the /etc/selinux/config file.
**SELINUX=permissive** # Configure SELinux to start in Permissive mode on boot, where policy violations are logged but not enforced.

**SELINUX=disabled** # Configure SELinux to be completely disabled on boot by setting this in the /etc/selinux/config file.

## 19. System Logging and Variable Management

**dmesg** # Display kernel ring buffer messages, typically used for reviewing system boot and hardware-related logs.

**journalctl** # View logs maintained by the systemd journal, including boot messages, application logs, and system events.

**journalctl -u servicename** # Display logs specific to a particular service or unit, making it easier to troubleshoot service issues.

**Variables let "[variable_name]=[value]"** # Assign an integer value to a variable, useful for arithmetic operations in bash.

**export [variable_name]** # Mark a variable to be available for child processes of the shell, making it an environment variable.

**declare [variable-name]="[value]"** # Declare and initialize a variable, optionally adding attributes like -i for integers.

**set** # Display all shell variables and functions currently defined in the session.

**unset [variable_name]** # Remove a shell or environment variable, effectively deleting it from the session.

**echo $[variable-name]** # Print the value stored in the variable to the terminal.