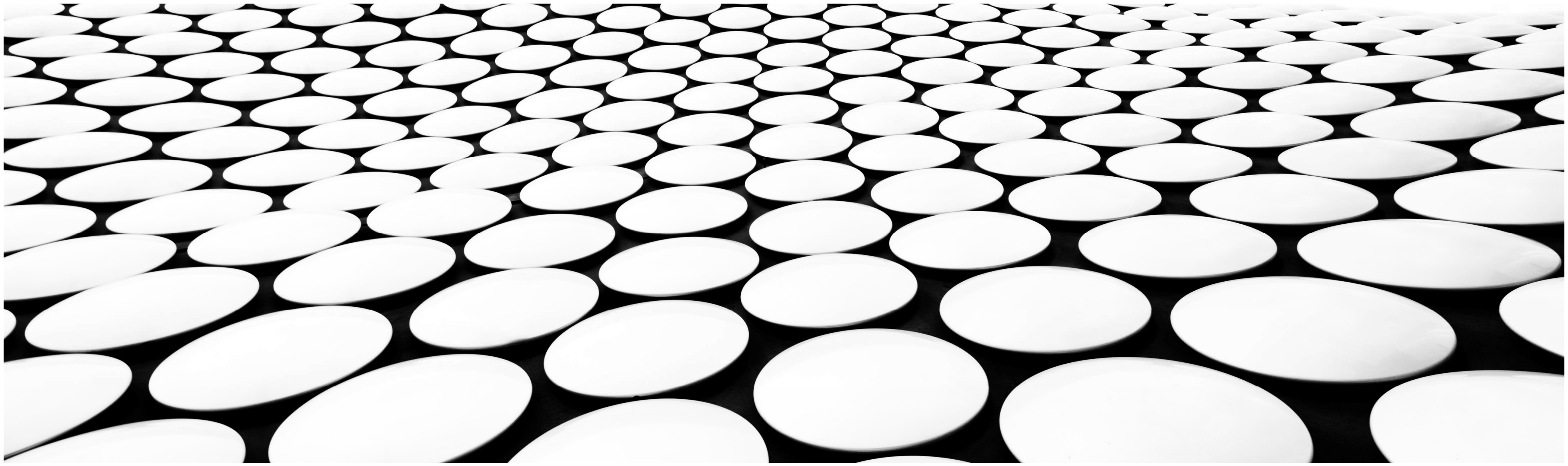
LINUX FOR DEVOPS

BY JAI



SECTION 01

WHY YOU NEED LINUX FOR DEVOPS?

Linux is an essential tool for DevOps because it is a stable, reliable, and secure operating system. DevOps teams need an operating system that can support their tools and processes, and Linux offers the flexibility and scalability they require.

Here are a few reasons why Linux is important for DevOps:

1. Open source: Linux is an open-source operating system that allows DevOps teams to customize and modify it to meet their specific needs. This flexibility enables DevOps teams to develop, test, and deploy applications quickly and efficiently.
2. Automation: DevOps teams rely heavily on automation to streamline their processes and reduce manual tasks. Linux provides a rich set of command-line tools and scripting languages, such as Bash, Python, and Perl, that enable automation of various tasks.
3. Scalability: Linux can run on a wide range of hardware, from small embedded devices to large server clusters, making it an ideal platform for DevOps teams that need to scale their applications quickly.
4. Security: Linux is known for its robust security features, which are essential for DevOps teams that handle sensitive data and need to protect against cyber threats.
5. Containers: Linux provides excellent support for containerization technologies, such as Docker and Kubernetes, which are essential for DevOps teams that need to package, deploy, and manage applications across multiple environments.

In summary, Linux is essential for DevOps because it provides the flexibility, scalability, and security that teams need to develop, deploy, and manage applications efficiently.

LEARN LINUX

Learning Linux for DevOps can seem intimidating, especially if you're new to the operating system. However, with the right resources and a bit of practice, you can master the basics of Linux and start using it for DevOps tasks.

Learn Linux for DevOps easily:

1. Start with the basics: Begin by learning the fundamental concepts of Linux, such as file systems, permissions, and command-line tools. You can find plenty of online tutorials, courses, and documentation that cover these topics.
2. Set up a Linux environment: To get hands-on experience with Linux, you'll need to set up a Linux environment on your computer. You can either install a Linux distribution, such as Ubuntu or CentOS, on your machine or use a virtual machine or containerized environment.
3. Practice regularly: The more you practice using Linux, the more comfortable you'll become with its command-line interface and tools. Try performing basic tasks, such as navigating directories, creating and editing files, and installing software.
4. Use DevOps tools: Start using DevOps tools that run on Linux, such as Docker and Kubernetes, to gain practical experience with Linux in a DevOps context.
5. Join a community: Join online communities, such as forums and social media groups, where you can ask questions, share tips, and connect with other DevOps professionals who use Linux.
6. Take online courses: There are many online courses available that can teach you Linux for DevOps. Look for courses that cover topics such as shell scripting, containerization, and DevOps toolchains.

SECTION 02

INSTALLATION

- Using cloud
- Using Vmware or VBox

SECTION 03
PRACTICAL COMMANDS

NAVIGATE IN THE LINUX FILE SYSTEM

Three commonly used Linux commands with examples:

- ls - The ls command is used to list files and directories in the current working directory.
- Example: To list all the files and directories in the current directory, simply type ls and press enter. You can also use options with the ls command to get more detailed information, such as ls -l to get a long listing of files and directories or ls -a to show hidden files.

- pwd - The pwd command is used to print the current working directory.
- Example: To find out which directory you are currently working in, simply type pwd and press enter. The output will show the full path of your current working directory.

- cd - The cd command is used to change the current working directory.
- Example: To change the current working directory to a specific directory, type cd followed by the directory name. For example, to change the current working directory to the "Documents" directory, type cd Documents and press enter. To move up one directory, use cd ...

- Overall, these three commands are essential for navigating and managing files and directories in a Linux environment.

LINUX AUTOCOMPLETION, HISTORY, AND CLEARING THE TERMINAL

Autocompletion:

- Autocompletion is a feature that allows Linux users to type a partial command or filename and have the shell complete it automatically.
- To use autocompletion, press the Tab key after typing a partial command or filename.
- Example: To list all files in the current directory that start with the letter 'a', type ls a and press Tab. The shell will automatically complete the command, showing all files that match the pattern.

History:

- History is a feature that allows Linux users to view and reuse previously executed commands.
- To view the command history, use the history command.
- To reuse a previously executed command, type ! followed by the command number.
- Example: To rerun the 10th command in the history list, type !10 and press enter.

Clearing the Terminal:

- Clearing the terminal is a feature that allows Linux users to remove all previous commands and output from the terminal window.
- To clear the terminal, use the clear command or press Ctrl + L.
- Example: To clear the terminal, simply type clear and press enter.

Overall, Linux Autocompletion, History, and Clearing the Terminal are powerful features that can save time and improve productivity for Linux users. By mastering these features, users can work more efficiently and effectively in a Linux environment.

FILESYSTEM OVERVIEW

Linux file system

Root directory:

- The root directory is denoted by / and is the top-level directory of the Linux file system.
- Example command: cd / - changes the current working directory to the root directory.

/bin and /sbin:

- The /bin directory contains essential binaries (executables) that are required for basic system operation, while the /sbin directory contains system binaries used for system administration tasks.
- Example command: ls /bin - lists all the files and directories in the /bin directory.

/etc:

- The /etc directory contains system configuration files.
- Example command: ls /etc - lists all the files and directories in the /etc directory.

/home:

- The /home directory contains home directories for all non-root users.
- Example command: cd /home/username - changes the current working directory to the home directory of a specific user.\

/tmp:

- The /tmp directory is used for temporary file storage.
- Example command: ls /tmp - lists all the files and directories in the /tmp directory.

/var:

- The /var directory contains variable data such as logs, mail, and printer spools.
- Example command: ls /var/log - lists all the log files in the /var/log directory.

/proc:

- The /proc directory is a virtual file system that contains information about running processes and system resources.
- Example command: cat /proc/cpuinfo - displays information about the CPU(s) in the system.

Overall, understanding the Linux file system and its directories is essential for managing and navigating the system efficiently. These example commands can help Linux users get started with exploring and working with different directories.

ABSOLUTE VS RELATIVE PATHS

Absolute Paths:

- Absolute paths start with the root directory / and specify the full path to a file or directory.
- Advantages: Provide an exact location for a file or directory, which can be useful in scripting or automation tasks.
- Disadvantages: Can be long and cumbersome to type out, and may become invalid if the file or directory is moved or renamed.
- Example Navigation Command:
  - cd /usr/share/applications - changes the current working directory to the applications directory under the share directory under the usr directory.

Relative Paths:

- Relative paths specify the path to a file or directory relative to the current working directory.
- Advantages: Shorter and easier to type than absolute paths, and can be used even if the absolute path is not known.
- Disadvantages: May be less precise than absolute paths, and may lead to errors if the current working directory is not known or changes frequently.
- Example Navigation Command:
  - cd .. - moves up one directory from the current working directory.
  - cd Documents/ - changes the current working directory to the Documents directory in the current working directory.

Overall, understanding the differences between absolute and relative paths is important for navigating and managing files and directories in Linux. By using the appropriate path type for a given task, users can work more efficiently and avoid errors.

SECTION 04
FILES AND FOLDERS MANAGEMENT

CREATE AND MANAGE FILES

Linux commands used to create and manage files:

touch:

- The touch command is used to create a new empty file or update the modification and access times of an existing file.
- Example: touch file.txt - creates a new empty file named file.txt.

rm:

- The rm command is used to remove files or directories.
- Example: rm file.txt - removes the file named file.txt.
- Regularly used attributes: -r (used to remove directories and their contents recursively), -f (used to force the removal of a file or directory).

cp:

- The cp command is used to copy files or directories.
- Example: cp myfile.txt mycopy.txt - creates a copy of the file myfile.txt named mycopy.txt.
- Regularly used attributes: -r (used to copy directories and their contents recursively), -p (used to preserve file attributes such as permissions, timestamps, and ownership).

mv:

- The mv command is used to move or rename files or directories.
- Example: mv myfile.txt mydir/ - moves the file myfile.txt to the directory mydir.
- Regularly used attributes: -i (used to prompt the user before overwriting an existing file), -v (used to display verbose output).

MAN PAGES

What are man pages?

- Man pages are the built-in documentation for most Linux commands and utilities.
- They provide detailed information about the usage, options, and syntax of a command.

How to access man pages?

- Man pages can be accessed using the man command followed by the name of the command or utility.
- Example: man ls - displays the man page for the ls command.

How to navigate man pages?

- Man pages are displayed in the terminal and can be navigated using keyboard commands.
- The most commonly used keyboard commands are:
  - q - quit the man page
  - spacebar - move down one page
  - b - move up one page
  - /searchterm - search for a specific term in the man page
  - n - move to the next occurrence of the search term
  - p - move to the previous occurrence of the search term

Sections of man pages:

- Man pages are divided into sections based on the type of command or utility being documented.
- The most commonly used sections are:
  - Section 1: User commands
  - Section 2: System calls
  - Section 3: C library functions
  - Section 4: Device files and drivers
  - Section 5: File formats and conventions
  - Section 6: Games and screensavers
  - Section 7: Miscellaneous

Overall, man pages are a valuable resource for Linux users and provide a comprehensive reference for the many commands and utilities available in the system. Understanding how to access and navigate man pages can help users efficiently and effectively use the Linux command line.

CREATE AND MANAGE DIRECTORIES

mkdir:

- The mkdir command creates a new directory.
- Example: mkdir mydir - creates a new directory named mydir.
- Regularly used attributes: -p (used to create parent directories as needed), -m (used to set permissions for the new directory).

rm -rf:

- The rm command with the -rf option is used to remove directories and their contents recursively.
- Example: rm -rf mydir - removes the directory named mydir and its contents.
- Regularly used attributes: -i (used to prompt the user before removing each file), -v (used to display verbose output).

ls -R:

- The ls command with the -R option is used to list the contents of a directory and its subdirectories recursively.
- Example: ls -R mydir - lists the contents of the directory mydir and its subdirectories.
- Regularly used attributes: -l (used to display a long format list of file information), -a (used to display hidden files).

cp:

- The cp command is used to copy directories and their contents.
- Example: cp -r mydir mycopydir - creates a copy of the directory mydir named mycopydir.
- Regularly used attributes: -r (used to copy directories and their contents recursively), -p (used to preserve file attributes such as permissions, timestamps, and ownership).

mv:

- The mv command is used to move or rename directories.
- Example: mv mydir mynewdir - renames the directory mydir to mynewdir.
- Regularly used attributes: -i (used to prompt the user before overwriting an existing file), -v (used to display verbose output).

MANAGING FILES

cat:

- The cat command is used to concatenate and display files.
- Example: cat myfile.txt - displays the contents of the file myfile.txt.
- Regularly used attributes: -n (used to display line numbers), -b (used to display line numbers for non-blank lines).

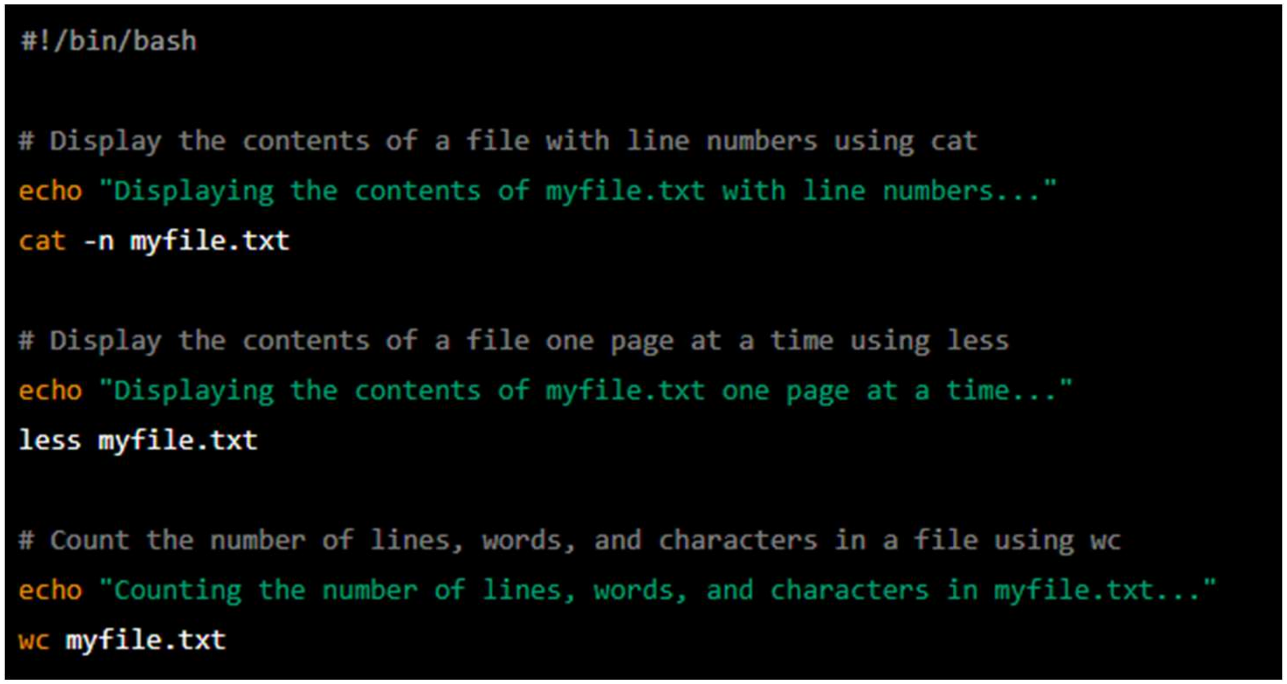
less:

- The less command is used to display files one page at a time, allowing scrolling and searching.
- Example: less myfile.txt - displays the contents of the file myfile.txt one page at a time.
- Regularly used attributes: / (used to search for a string in the file), q (used to quit less).

wc:

- The wc command is used to count the number of lines, words, and characters in a file.
- Example: wc myfile.txt - displays the number of lines, words, and characters in the file myfile.txt.
- Regularly used attributes: -l (used to count only the number of lines), -w (used to count only the number of words), -c (used to count only the number of characters).

EXAMPLE BASH SCRIPT USING CAT, LESS AND WC



```
#!/bin/bash

# Display the contents of a file with line numbers using cat
echo "Displaying the contents of myfile.txt with line numbers..."
cat -n myfile.txt

# Display the contents of a file one page at a time using less
echo "Displaying the contents of myfile.txt one page at a time..."
less myfile.txt

# Count the number of lines, words, and characters in a file using wc
echo "Counting the number of lines, words, and characters in myfile.txt..."
wc myfile.txt
```

In this script, the cat command is used to display the contents of myfile.txt with line numbers, the less command is used to display the contents of myfile.txt one page at a time, and the wc command is used to count the number of lines, words, and characters in myfile.txt.

When the script is run, the user will see the output of each command in the terminal. This script can be modified to work with other text files and commands as needed.

ECHO COMMAND

Writing text to a file:

- The **echo** command can be used to write text to a file using the redirection operator (>).
- Example: **echo "Hello World!" > myfile.txt** - writes the text "Hello World!" to a file called **myfile.txt.**
- The > operator will create a new file or overwrite the contents of an existing file.

Appending text to a file:

- The **echo** command can also be used to append text to a file using the append operator (>>).
- Example: **echo "Goodbye World!" >> myfile.txt** - appends the text "Goodbye World!" to the end of the file **myfile.txt.**
- The >> operator will not overwrite existing content in the file, but will add to the end of the file.

Overall, the echo command is a simple and effective way to write text to a file from the terminal in Linux. It can be used for creating and modifying text files, writing scripts, and more.

SECTION 05
EDIT FILES USING VI

The vi editor is a popular and powerful text editor in Linux that can be used to create and modify text files. Here are some presenting points and examples on how to use the vi editor:

Opening a file:

- To open a file in vi, use the command vi followed by the name of the file.
- Example: vi myfile.txt - opens the file myfile.txt in vi.

Modes:

- vi has two modes: command mode and insert mode. Command mode is used for navigation and other commands, while insert mode is used for inserting and editing text.
- To switch from command mode to insert mode, press the i key. To switch back to command mode, press the Esc key.

Navigation:

- In command mode, use the arrow keys to move the cursor around the file.
- Use gg to move to the beginning of the file and G to move to the end of the file.
- Use :n to move to a specific line number.

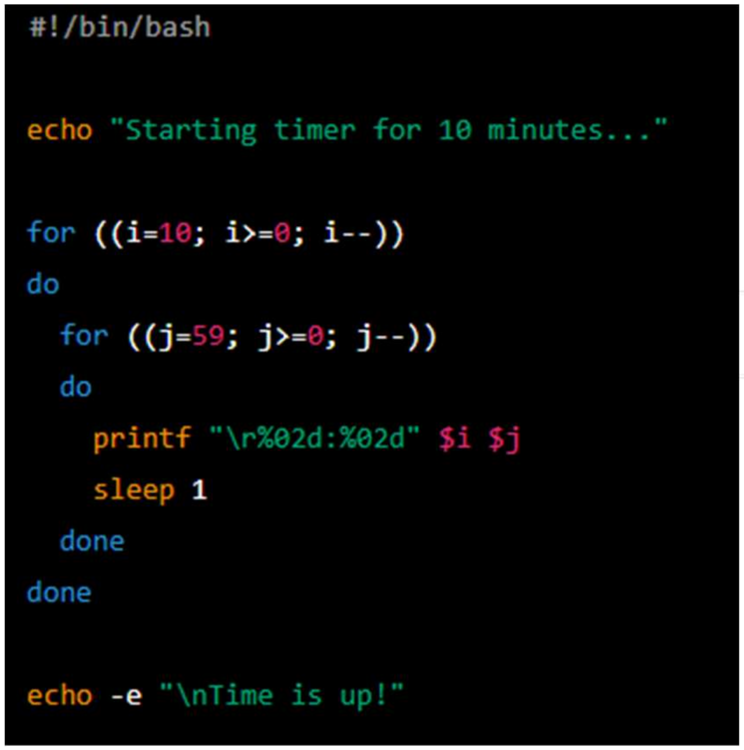
Editing:

- In insert mode, type to add or edit text in the file.
- Use dd to delete a line, yy to copy a line, and p to paste a copied or deleted line.

Saving and quitting:

- To save changes and exit vi, use the command :wq.
- To quit without saving changes, use the command :q!.

CREATE AND EXECUTE A BASH FILE TO CREATE A TIMER FOR 10MINUTS USING VI EDITOR



```
#!/bin/bash

echo "Starting timer for 10 minutes..."

for ((i=10; i>=0; i--))
do
  for ((j=59; j>=0; j--))
  do
    printf "\r%02d:%02d" $i $j
    sleep 1
  done
done

echo -e "\nTime is up!"
```

```
#!/bin/bash

echo "Starting timer for 10 minutes..."

for ((i=10; i>=0; i--))
do
  for ((j=59; j>=0; j--))
  do
    printf "\r%02d:%02d" $i $j
    sleep 1
  done
done

echo -e "\nTime is up!"
```

- This code uses nested for loops to count down the time from 10 minutes (600 seconds) to zero, updating the timer display every second using the **printf** command with the \r option to overwrite the previous output on the same line.
- The **%02d** format specifier in the **printf** command pads the numbers with leading zeros to ensure that the minutes and seconds are always displayed with two digits.
- You can modify the time duration in the script by changing the value of the **i** variable in the outer loop, or by replacing **10** with another value in the loop condition.
- Save the script using the **vi** editor and make it executable using the **chmod +x timer.sh** command as described earlier. Then run the script by typing **./timer.sh** in the terminal to start the timer.

SECTION 06
USERS AND PERMISSIONS

NORMAL USER ACCESS

What You Can perform:

- View and manipulate files that you have permission to access.
- Example: cat, less, vim, cp, mv, rm commands.

Install and use applications that don't require root access.

- Example: apt-get, yum, pip commands.

Create, modify, and delete files and directories within your own home directory.

- Example: mkdir, touch, chmod, chown, rmdir commands.

Start and stop processes that are running under your user account.

- Example: systemctl --user start <service>, systemctl --user stop <service> commands.

What You Can’t Perform:

Install software that requires root access.

- Example: sudo apt-get install <package>.

Modify system files that require root access.

- Example: /etc/hosts, /etc/sudoers, /etc/passwd, /etc/group files.

Start and stop system-level services.

- Example: systemctl start <service>, systemctl stop <service> commands.

Change the permissions or ownership of system files and directories.

- Example: chmod, chown commands.

Keep in mind that attempting to perform actions that require root access without proper permissions can result in failure or errors. It's important to understand the limits of your user account and use sudo or switch to the root user when necessary to perform administrative tasks.

For example, if you try to install a package using apt-get without using sudo, you will likely see an error message indicating that you don't have permission to perform the action. Similarly, if you try to modify a system file without proper permissions, the changes will not be saved and you may see an error message indicating that the operation failed.

ADMIN PRIVILEGE (SUDO)

- In Linux, the **sudo** command is used to run commands with administrative or "superuser" privileges. Here are some points to keep in mind when using **sudo**:
  - Prefix the command with **sudo** to run it with superuser privileges. For example: **sudo apt-get update**.
  - Enter your user password when prompted. By default, **sudo** will ask for your user password before running the command.
  - Only use **sudo** when necessary, as running commands with superuser privileges can be dangerous and potentially cause damage to the system.
  - Use the **-s** or **-i** options with **sudo** to start a new shell or session as the superuser. For example: **sudo -s** or **sudo -i**.
  - Use **sudo -k** to invalidate the current **sudo** timestamp, which will force you to enter your password again the next time you run a **sudo** command.

ADMIN PRIVILEGE (SUDO)

Installing a package with sudo:

- sudo apt-get install <package-name>

Editing a system file with sudo:

- sudo vim /etc/hosts

Restarting a service with sudo:

- sudo systemctl restart <service-name>

Running a command as another user with sudo:

- sudo -u <username> <command>

Copying a file with sudo:

- sudo cp <source> <destination>

UNDERSTAND FILE OWNER AND PERMISSIONS

- In Linux, each file and directory has an owner, which is a user or a system process that has created the file or directory. Here are some points to keep in mind when working with file ownership:
  - Use the ls -l command to view the owner and group of a file or directory. The output shows the owner and group name, along with other file attributes.
  - The owner of a file or directory has the most control over it and can read, write, and execute it by default. Other users on the system have limited access based on the file's permissions.
  - The file owner can be changed using the **chown** command. The syntax for **chown** is **chown newowner file.**
  - The file group can be changed using the **chgrp** command. The syntax for **chgrp** is **chgrp newgroup file.**

To view the owner and group of a file:

- ls -l filename

To change the owner of a file to a specific user:

- sudo chown username filename

To change the group of a file to a specific group:

- sudo chgrp groupname filename

To change the owner and group of a directory and all its contents recursively:

- sudo chown -R username:groupname directoryname

Note that the -R option is used to apply the changes recursively to all the files and directories inside the specified directory.

LINUX PERMISSIONS

In Linux, file permissions are used to control access to files and directories. Here are some points to keep in mind when working with file permissions:

- There are three types of permissions: read, write, and execute. These permissions can be set for three different types of users: the file owner, the file group, and all other users.
- Use the ls -l command to view the permissions of a file or directory. The output shows a series of letters that represent the permissions for the file owner, group, and other users.
- The letters r, w, and x represent read, write, and execute permissions, respectively. A dash (-) in place of a letter means that the permission is not granted.
- Permissions can be changed using the chmod command. The syntax for chmod is chmod [options] mode file.
- The mode can be specified in either octal or symbolic notation. Octal notation represents the permissions as a three-digit number, where each digit represents the permissions for the owner, group, and other users. Symbolic notation uses letters and symbols to specify the permissions.

To view the permissions of a file:

- ls -l filename

To give the owner of a file read and write permissions:

- chmod u+rw filename

To remove execute permissions for all other users:

- chmod o-x filename

To give the owner and group of a file full permissions and remove all permissions for other users:

- chmod ug=rwx,o= filename

To give the owner, group, and other users read and write permissions, and remove execute permissions:

- chmod a=rw filename

Note that the chmod command can also be used with the -R option to apply the changes recursively to all the files and directories inside the specified directory.

In octal notation, permissions are represented as a three-digit number, where each digit represents the permissions for the owner, group, and other users, respectively. Each digit is the sum of the following values:

- 4: read permission
- 2: write permission
- 1: execute permission

Here are some examples of octal notation for different combinations of permissions:

- 421: read, write, and execute permissions for the owner, read permission for the group, and execute permission for other users
- 644: read and write permissions for the owner, and read permission for the group and other users
- 777: full permissions for the owner, group, and other users

To set these permissions using the **chmod** command, you can use the following commands:

- **chmod 421 filename** # set read, write, and execute permissions for the owner, read permission for the group, and execute permission for other users
- **chmod 644 filename** # set read and write permissions for the owner, and read permission for the group and other users
- **chmod 777 filename** # set full permissions for the owner, group, and other users

CHOWN COMMAND

The chown command is used to change the owner of a file or directory in Linux. Here are some presenting points and examples of its usage:

Syntax: chown [OPTIONS] USER[:GROUP] FILE

Arguments:

- USER: The new owner of the file or directory
- GROUP: (Optional) The new group owner of the file or directory
- FILE: The file or directory whose owner is to be changed

Commonly used options:

- -R: Recursively change ownership of all files and subdirectories under the specified directory
- --from=CURRENT_OWNER[:CURRENT_GROUP]: Change ownership only of files or directories that are currently owned by the specified user and/or group
- --reference=REFERENCE_FILE: Change ownership of the specified file or directory to match that of the reference file

Examples:

- Change the owner of a file: sudo chown user myfile.txt
- Change the owner and group of a file: sudo chown user:group myfile.txt
- Change the owner of a directory and all its contents: sudo chown -R user mydirectory
- Change the owner of files owned by a specific user: sudo chown --from=olduser newuser myfile.txt
- Change the owner of a file to match another file: sudo chown --reference=referencefile myfile.txt

It's important to use caution when changing file ownership, as it can affect the security and accessibility of the file or directory. Be sure to check the documentation or seek guidance before making any changes, particularly on system files or directories.hown --reference=referencefile myfile.txt

SECTION 07
INSTALL AND UPDATE SOFTWARE

YUM TO INSTALL AND UPDATE PACKAGES

yum is a command-line package manager for Linux that allows users to install, update, and remove software packages

yum install [PACKAGE]: installs a package and its dependencies from the configured repositories.

- Example: yum install httpd

yum update [PACKAGE]: updates a package and its dependencies to the latest version.

- Example: yum update httpd

yum remove [PACKAGE]: removes a package and its dependencies from the system.

- Example: yum remove httpd

yum search [SEARCH TERM]: searches for packages matching the specified search term.

- Example: yum search apache

yum info [PACKAGE]: displays detailed information about a package, including its dependencies.

- Example: yum info httpd

yum list: displays a list of all installed packages.

- Example: yum list installed

yum list updates: displays a list of packages that have available updates.

- Example: yum list updates

yum clean [OPTION]: clears the package cache, freeing up disk space.

- Options:
  - all: removes all cached packages.
  - metadata: removes only the package metadata.
- Example: yum clean all

yum check-update: checks for updates to all installed packages.

- Example: yum check-update

yum upgrade: upgrades all installed packages to their latest versions.

- Example: yum upgrade

These are some of the most commonly used yum commands in Linux. It is important to ensure that you are only installing packages from trusted repositories to maintain the security and stability of your system.

APT COMMAND TO INSTALL AND UPDATE PACKAGES

apt is a command-line package manager used in Debian-based Linux distributions, such as Ubuntu. Here are the equivalent apt commands for the yum.

apt install [PACKAGE]: installs a package and its dependencies from the configured repositories.

- Example: apt install apache2

apt update: updates the package lists from the configured repositories.

- Example: apt update

apt upgrade: upgrades all installed packages to their latest versions.

- Example: apt upgrade

apt remove [PACKAGE]: removes a package and its dependencies from the system.

- Example: apt remove apache2

apt search [SEARCH TERM]: searches for packages matching the specified search term.

- Example: apt search apache

apt show [PACKAGE]: displays detailed information about a package, including its dependencies.

- Example: apt show apache2

apt list: displays a list of all installed packages.

- Example: apt list --installed

apt list --upgradeable: displays a list of packages that have available updates.

- Example: apt list --upgradeable

apt autoclean: clears the package cache, freeing up disk space by removing packages that can no longer be downloaded.

- Example: apt autoclean

apt full-upgrade: upgrades all installed packages and removes packages that are no longer needed.

- Example: apt full-upgrade

These are the equivalent apt commands for the yum commands mentioned earlier. Remember to only install packages from trusted repositories to ensure the security and stability of your system.

RPM PACKAGE MANAGER

rpm is a command-line package manager used in Red Hat-based Linux distributions, such as Fedora and CentOS.

rpm -i [PACKAGE.rpm]: installs a package from a .rpm file.

- Example: rpm -i package.rpm

rpm -U [PACKAGE.rpm]: upgrades an installed package to a new version from a .rpm file.

- Example: rpm -U package.rpm

rpm -e [PACKAGE]: removes a package from the system.

- Example: rpm -e package

rpm -qa: lists all installed packages on the system.

- Example: rpm -qa

rpm -q [PACKAGE]: queries for the installed version of a specific package.

- Example: rpm -q package

rpm -qi [PACKAGE]: displays detailed information about an installed package.

- Example: rpm -qi package

rpm -qf [FILE]: queries which package a specific file belongs to.

- Example: rpm -qf /etc/passwd

rpm -ql [PACKAGE]: lists all files included in an installed package.

- Example: rpm -ql package

rpm -qp [PACKAGE.rpm] -l: lists all files included in an uninstalled .rpm package file.

- Example: rpm -qp package.rpm -l

rpm -Va: verifies the integrity of installed packages and reports any discrepancies.

- Example: rpm -Va

These are some of the commonly used rpm commands. Remember to only install packages from trusted sources to ensure the security and stability of your system.

DPKG PACKAGE MANAGER

- In Debian-based Linux distributions, such as Ubuntu, dpkg is used as the command-line package manager.

dpkg -i [PACKAGE.deb]: installs a package from a .deb file.

- Example: dpkg -i package.deb

dpkg -r [PACKAGE]: removes a package from the system.

- Example: dpkg -r package

dpkg -l: lists all installed packages on the system.

- Example: dpkg -l

dpkg -s [PACKAGE]: displays detailed information about an installed package.

- Example: dpkg -s package

dpkg -S [FILE]: queries which package a specific file belongs to.

- Example: dpkg -S /etc/passwd

dpkg -L [PACKAGE]: lists all files included in an installed package.

- Example: dpkg -L package

dpkg -c [PACKAGE.deb]: lists all files included in an uninstalled .deb package file.

- Example: dpkg -c package.deb

dpkg --configure [PACKAGE]: configures an installed package that has been only partially configured.

- Example: dpkg --configure package

These are some of the commonly used dpkg commands. Similar to rpm, it is important to only install packages from trusted sources to ensure the security and stability of your system.

SECTION 08
COMMANDS REGULARLY USED IN REALTIME

FIND COMMAND

- The find command is a powerful tool for searching for files and directories based on various criteria.

-name: searches for files with a specific name.

- Example: find /home -name "file.txt"

-type: searches for files of a specific type.

- Example: find / -type d (searches for directories)

-size: searches for files of a specific size.

- Example: find / -size +10M (searches for files larger than 10 megabytes)

-mtime: searches for files modified within a specific time period.

- Example: find /home -mtime -7 (searches for files modified within the last 7 days)

-user: searches for files owned by a specific user.

- Example: find / -user john

-exec: executes a command on the found files.

- Example: find /home -name "*.txt" -exec cp {} /backup \; (copies all files with a .txt extension to the /backup directory)

-print: prints the names of the found files.

- Example: find /home -name "*.txt" –print

-iname: searches for files with a specific name, ignoring case.

- Example: find /home -iname "file.txt"

-perm: searches for files with specific permissions.

- Example: find / -perm 644 (searches for files with read/write permissions for the owner, and read-only permissions for group and others)

-empty: searches for empty files or directories.

- Example: find /home -type d -empty (searches for empty directories)

-maxdepth: limits the search depth to a specific number of levels.

- Example: find /home -maxdepth 2 -name "*.txt" (searches for .txt files in the first two levels of the /home directory)

-mindepth: sets the minimum search depth.

- Example: find /home -mindepth 2 -name "*.txt" (searches for .txt files in subdirectories of /home, but not in /home itself)

-delete: deletes the found files.

- Example: find /tmp -type f -name "core.*" -delete (deletes all files in the /tmp directory with a name starting with core.)

GREP COMMAND

grep is a powerful command-line utility that allows you to search for specific patterns in text files.

-i: ignore case while searching.

- Example: grep -i "hello" file.txt

-r: search recursively through subdirectories.

- Example: grep -r "error" /var/log

-w: search for whole words only.

- Example: grep -w "the" file.txt

-n: display the line numbers of the matched lines.

- Example: grep -n "error" /var/log/messages

-v: display the lines that do not match the pattern.

- Example: grep -v "debug" /var/log/messages

-E: use extended regular expressions.

- Example: grep -E "(word1|word2)" file.txt

-l: display only the names of files that contain the pattern.

- Example: grep -l "error" /var/log/*

EGREP COMMAND

egrep is a command-line utility that is similar to grep, but it uses extended regular expressions for pattern matching

Search for lines that contain two or more occurrences of the word "the" using the {} operator:

- egrep '(the){2,}' file.txt

Search for lines that contain a string that starts with "cat" and ends with "dog":

- egrep 'cat.*dog' file.txt

Search for lines that contain a string that starts with "cat" or "dog":

- egrep 'cat|dog' file.txt

Search for lines that contain a string that starts with a lowercase letter, followed by any number of characters, and ends with a period:

- egrep '[a-z].*\.' file.txt

Search for lines that contain a string that starts with a digit, followed by a period, and ends with a space:

- egrep '[0-9]\. ' file.txt

egrep is a powerful tool for searching through text files using regular expressions. By using extended regular expressions, you can create complex search patterns to find the information you need.

PIPE COMMAND WITH EXAMPLES

- In Linux, a pipe is a mechanism for connecting two commands and redirecting the output of one command as input to another command. It allows users to combine commands and create more complex commands to perform specific tasks. Here are some presenting points and examples of pipes:

1. Syntax: command1 | command2
2. The output of command1 becomes the input of command2.

- Examples:

To list all files in the current directory that have the word "example" in them:

- ls | grep example

To count the number of files that have the word "example" in them:

- ls | grep example | wc -l

To find all files in the current directory and its subdirectories that have the word "example" in them:

find . -type f | xargs grep example

To sort the output of a command and display only unique lines:

- command | sort | uniq

To display only the first 10 lines of a command's output:

- command | head

To display only the last 10 lines of a command's output:

- command | tail

Pipes are a powerful feature of the Linux command line and can be used to build complex commands for a wide variety of tasks.

TERMINAL SHORTCUTS

1. Ctrl + C: Interrupts the currently running command or process.

- Example: Suppose you are running a command that is taking too long to complete, you can press Ctrl + C to stop the command.

2. Ctrl + D: Logs out of the current session.

- Example: If you want to close the terminal window, you can use this shortcut.

3. Ctrl + Z: Puts the currently running command or process in the background.

- Example: Suppose you are running a command that is taking too long to complete, you can press Ctrl + Z to put it in the background and continue working on the terminal.

4. Ctrl + L: Clears the terminal screen.

- Example: If you want to clear the terminal screen, you can use this shortcut.

5. Tab: Autocompletes the command or file name you are typing.

- Example: Suppose you want to navigate to the "Downloads" folder in your home directory, you can type "cd Dow" and press Tab, and the command will be automatically completed to "cd Downloads".

6. Up arrow key: Displays the previous command executed in the terminal.

- Example: If you want to repeat the previous command, you can press the up arrow key.

7. Ctrl + R: Searches through the command history for a command that matches the entered characters.

- Example: Suppose you want to repeat a command that you executed a few commands ago, you can press Ctrl + R and start typing the command. The terminal will search for the command in the history and display it.

8. Ctrl + A: Moves the cursor to the beginning of the current command line.

- Example: If you want to edit the beginning of a long command you typed, you can press Ctrl + A to move the cursor to the beginning of the line.

9. Ctrl + E: Moves the cursor to the end of the current command line.

- Example: If you want to edit the end of a long command you typed, you can press Ctrl + E to move the cursor to the end of the line.

10. Ctrl + U: Clears the current command line.

- Example: If you want to clear the current command line, you can use this shortcut.

MULTIPLE TERMINALS FOR EASE OF WORK

- Open multiple terminals: To open multiple terminals, you can either use the keyboard shortcut Ctrl + Alt + T or right-click on the desktop and select "Open Terminal" from the context menu. You can open as many terminals as you need by repeating this process.
- Switch between terminals: To switch between terminals, you can use the keyboard shortcut Ctrl + Alt + Arrow Keys. This will allow you to move to the next or previous terminal.
- Split terminal window: You can split a terminal window into multiple panes using the Ctrl + Shift + D keyboard shortcut. This will create a new pane that shares the same terminal session as the original pane.
- Resize terminal window: You can resize a terminal window by clicking and dragging the borders. You can also use the keyboard shortcut Ctrl + Shift + + or Ctrl + Shift + - to increase or decrease the font size respectively.
- Run commands in multiple terminals simultaneously: You can run the same command in multiple terminals simultaneously by using the Ctrl + Shift + T keyboard shortcut. This will create a new terminal window with the same command running.
- Use terminal tabs: Some terminal emulators support tabs, allowing you to have multiple terminals in a single window. To create a new tab, you can use the keyboard shortcut Ctrl + Shift + T or click on the "New Tab" button in the terminal window.

SECTION 09
MONITORING PROCESSES AND RESOURCES

FIND AND KILL A LINUX PROCESS (PS, GREP, KILL)

In Linux, it is sometimes necessary to find and kill a process that is taking up too much memory or CPU. This can be accomplished using the following commands:

ps: This command shows a list of running processes on the system.

- Example: ps aux

grep: This command is used to search for a specific process by name or ID.

- Example: ps aux | grep firefox

kill: This command is used to terminate a process.

- Example: kill 1234
- Here, 1234 is the process ID that is to be killed.

Some important points to consider when using these commands are:

- The ps command can display a large amount of information, including the process ID, parent process ID, CPU usage, memory usage, and more.
- The grep command is used to filter the output of the ps command to find a specific process. The | (pipe) symbol is used to pass the output of the ps command to the grep command.
- The kill command sends a termination signal to the specified process ID. By default, it sends a SIGTERM signal, which allows the process to clean up before exiting. If the process does not exit within a certain amount of time, a SIGKILL signal can be sent to force the process to exit immediately.

Some commonly used options and arguments for these commands are:

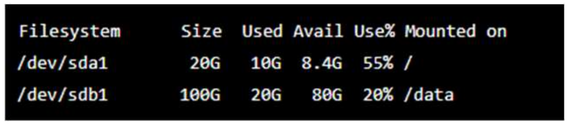
- ps aux: This command displays a list of all running processes on the system, along with their CPU and memory usage.
- ps aux | grep <process-name>: This command displays information about a specific process, filtered by the process name.
- kill <process-id>: This command sends a termination signal to the specified process ID.
- kill -9 <process-id>: This command sends a SIGKILL signal to the specified process ID, forcing it to terminate immediately.
- killall <process-name>: This command sends a termination signal to all processes with the specified name.

Overall, the combination of ps, grep, and kill commands provides an efficient way to find and terminate unwanted processes in Linux.

MONITOR SPACE AND USAGE (DF, HTOP)

df: The df command is used to display the amount of free and used disk space on a file system. The most common usage of this command is simply to type df -h to get a human-readable summary of disk usage for all file systems.

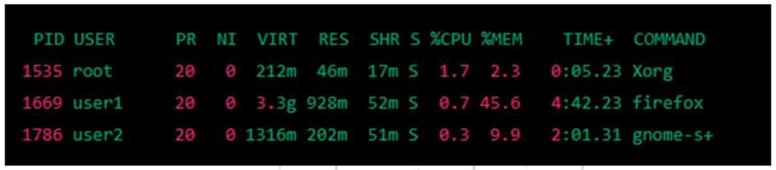
- Example: df –h



```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        20G   10G  8.4G  55% /
/dev/sdb1       100G   20G   80G  20% /data
```

htop: The htop command is used to monitor system resources such as CPU and memory usage in real-time. It provides a user-friendly interface for viewing and controlling running processes.

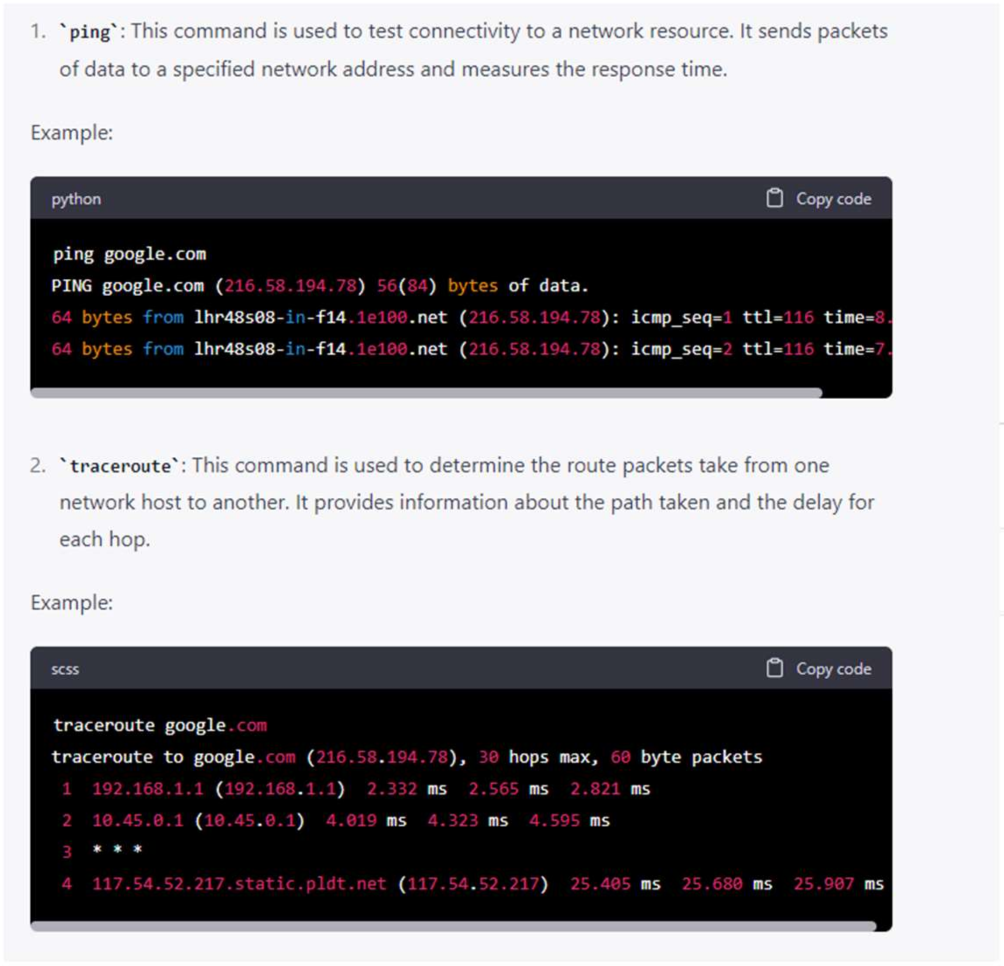
- Example: htop



```
  PID USER       PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1535 root       20   0  212m  46m  17m S  1.7  2.3  0:05.23 Xorg
 1669 user1      20   0  3.3g 928m  52m S  0.7 45.6  4:42.23 firefox
 1786 user2      20   0 1316m 202m  51m S  0.3  9.9  2:01.31 gnome-s+
```

By monitoring disk space and system resources, you can ensure that your system is running smoothly and avoid any potential performance issues

SECTION 10
TROUBLESHOOTING COMMANDS



1. `ping`: This command is used to test connectivity to a network resource. It sends packets of data to a specified network address and measures the response time.

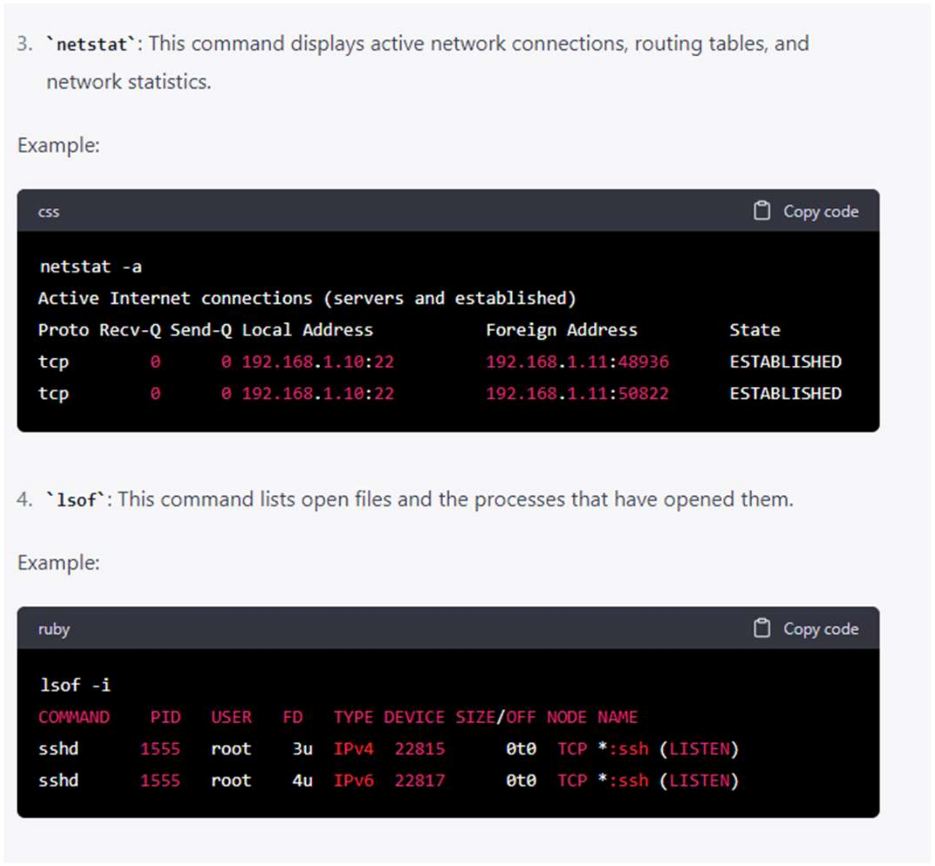
Example:

```
ping google.com
PING google.com (216.58.194.78) 56(84) bytes of data.
64 bytes from lhr48s08-in-f14.1e100.net (216.58.194.78): icmp_seq=1 ttl=116 time=8.
64 bytes from lhr48s08-in-f14.1e100.net (216.58.194.78): icmp_seq=2 ttl=116 time=7.
```

2. `traceroute`: This command is used to determine the route packets take from one network host to another. It provides information about the path taken and the delay for each hop.

Example:

```
traceroute google.com
traceroute to google.com (216.58.194.78), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  2.332 ms  2.565 ms  2.821 ms
 2  10.45.0.1 (10.45.0.1)  4.019 ms  4.323 ms  4.595 ms
 3  * * *
 4  117.54.52.217.static.pldt.net (117.54.52.217)  25.405 ms  25.680 ms  25.907 ms
```



3. `netstat`: This command displays active network connections, routing tables, and network statistics.

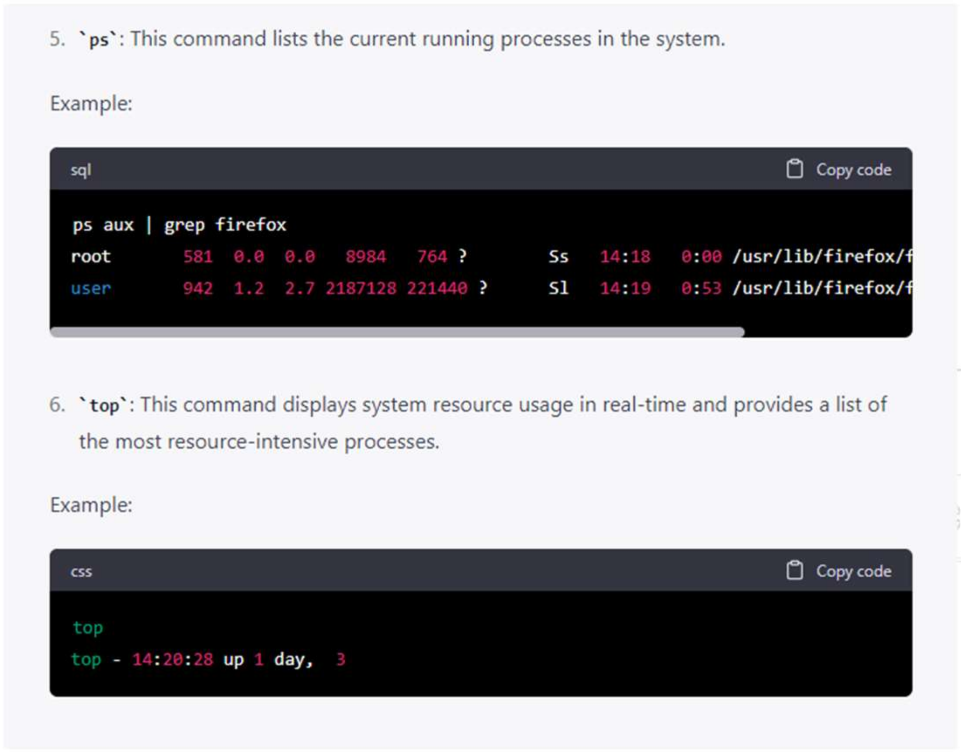
Example:

```
netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.1.10:22         192.168.1.11:48936      ESTABLISHED
tcp        0      0 192.168.1.10:22         192.168.1.11:50822      ESTABLISHED
```

4. `lsof`: This command lists open files and the processes that have opened them.

Example:

```
lsof -i
COMMAND    PID   USER    FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd      1555   root    3u  IPv4  22815      0t0  TCP *:ssh (LISTEN)
sshd      1555   root    4u  IPv6  22817      0t0  TCP *:ssh (LISTEN)
```



5. `ps`: This command lists the current running processes in the system.

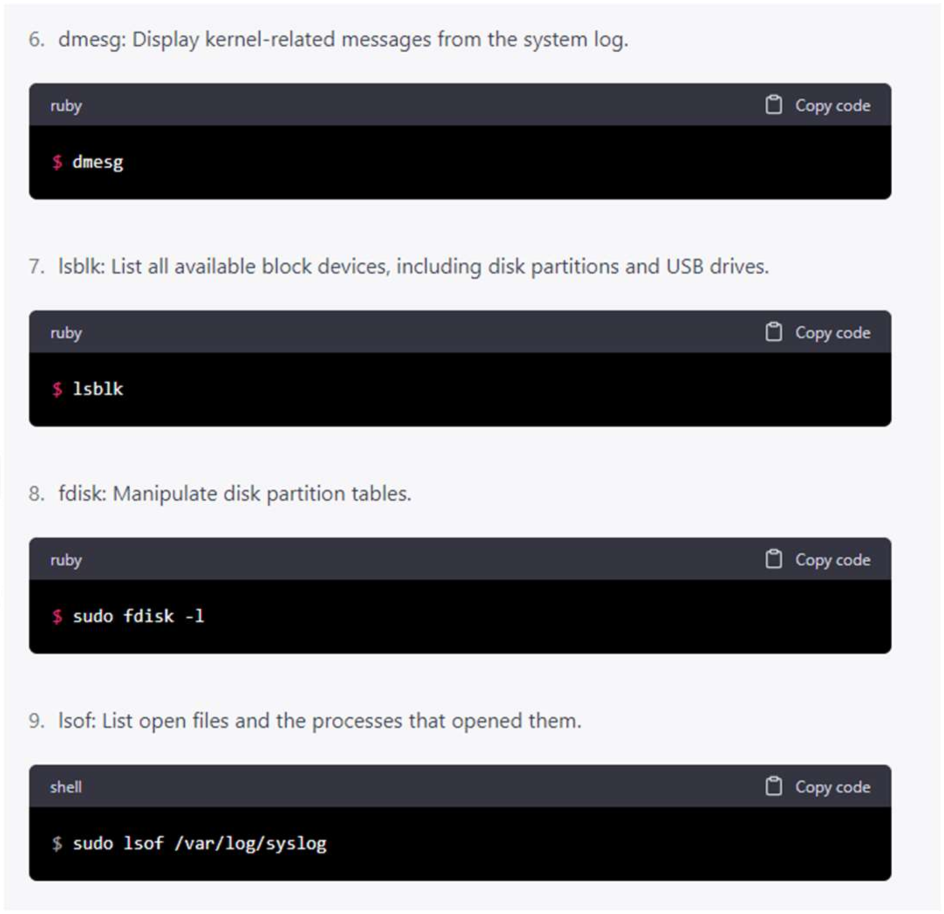
Example:

```
ps aux | grep firefox
root       581  0.0  0.0   8984   764 ?        Ss   14:18   0:00 /usr/lib/firefox/f
user       942  1.2  2.7 2187128 221440 ?      Sl   14:19   0:53 /usr/lib/firefox/f
```

6. `top`: This command displays system resource usage in real-time and provides a list of the most resource-intensive processes.

Example:

```
top
top - 14:20:28 up 1 day,  3
```



6. dmesg: Display kernel-related messages from the system log.

```
$ dmesg
```

7. lsblk: List all available block devices, including disk partitions and USB drives.

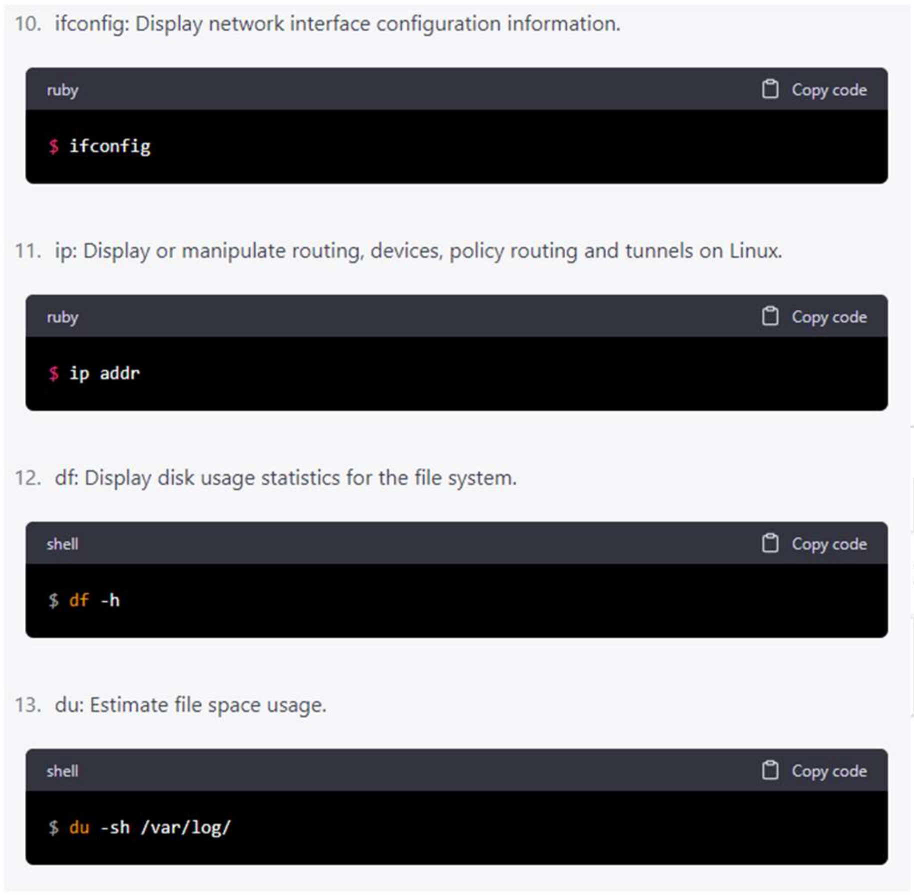
```
$ lsblk
```

8. fdisk: Manipulate disk partition tables.

```
$ sudo fdisk -l
```

9. lsof: List open files and the processes that opened them.

```
$ sudo lsof /var/log/syslog
```



10. ifconfig: Display network interface configuration information.

```ruby
$ ifconfig
```

11. ip: Display or manipulate routing, devices, policy routing and tunnels on Linux.

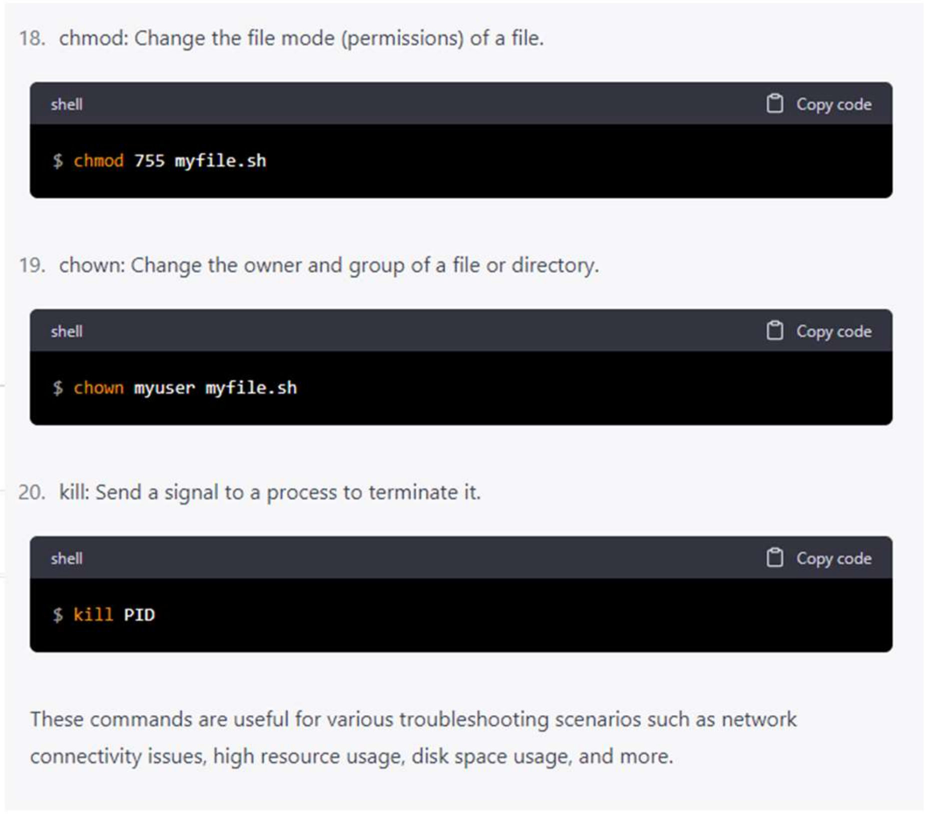
```ruby
$ ip addr
```

12. df: Display disk usage statistics for the file system.

```shell
$ df -h
```

13. du: Estimate file space usage.

```shell
$ du -sh /var/log/
```



18. chmod: Change the file mode (permissions) of a file.

```shell
$ chmod 755 myfile.sh
```

19. chown: Change the owner and group of a file or directory.

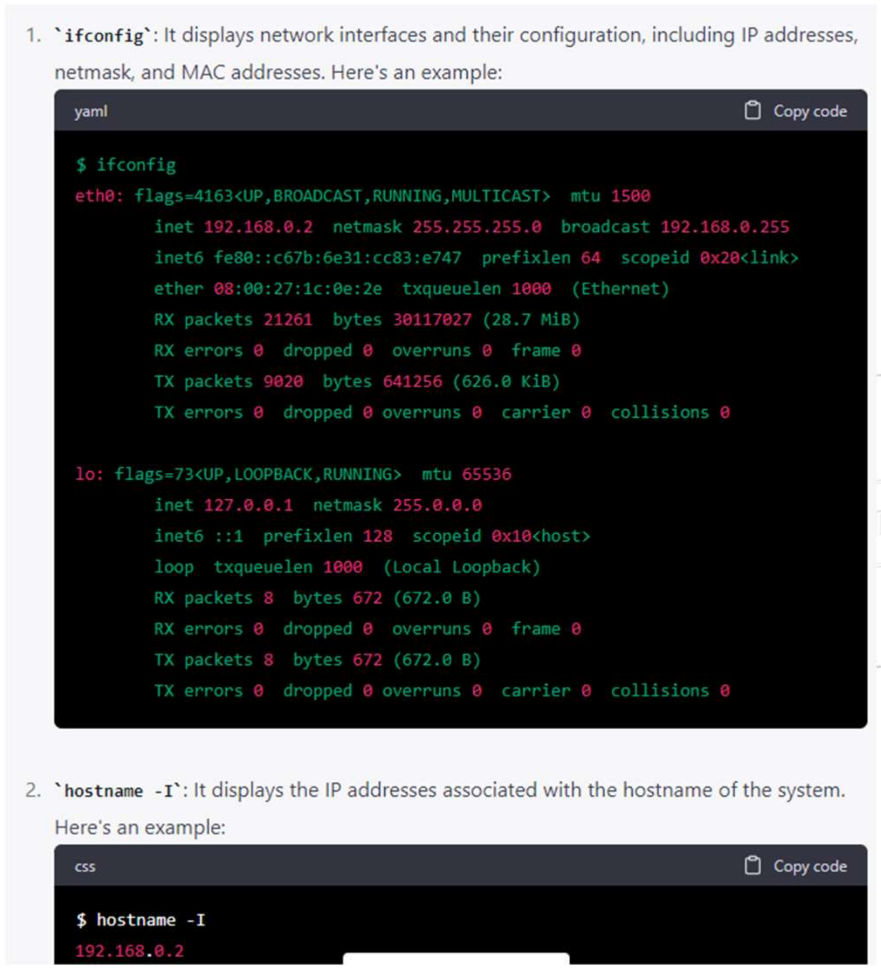
```shell
$ chown myuser myfile.sh
```

20. kill: Send a signal to a process to terminate it.

```shell
$ kill PID
```

These commands are useful for various troubleshooting scenarios such as network connectivity issues, high resource usage, disk space usage, and more.

NETWORKING COMMANDS



1. `ifconfig`: It displays network interfaces and their configuration, including IP addresses, netmask, and MAC addresses. Here's an example:

yaml Copy code

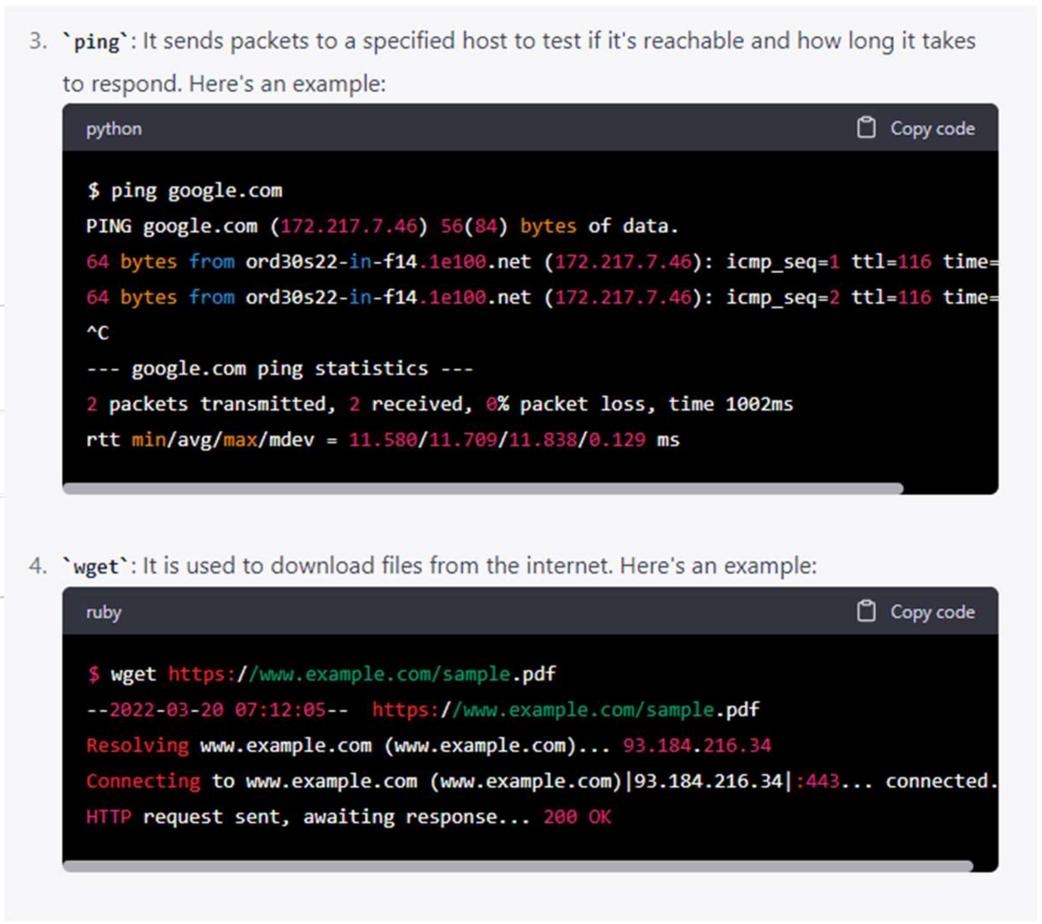
```
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.2  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::c67b:6e31:cc83:e747  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:1c:0e:2e  txqueuelen 1000  (Ethernet)
        RX packets 21261  bytes 30117027 (28.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9020  bytes 641256 (626.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 672 (672.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 672 (672.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. `hostname -I`: It displays the IP addresses associated with the hostname of the system. Here's an example:

css Copy code

```
$ hostname -I
192.168.0.2
```



3. `ping`: It sends packets to a specified host to test if it's reachable and how long it takes to respond. Here's an example:

python Copy code

```
$ ping google.com
PING google.com (172.217.7.46) 56(84) bytes of data.
64 bytes from ord30s22-in-f14.1e100.net (172.217.7.46): icmp_seq=1 ttl=116 time=
64 bytes from ord30s22-in-f14.1e100.net (172.217.7.46): icmp_seq=2 ttl=116 time=
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 11.580/11.709/11.838/0.129 ms
```

4. `wget`: It is used to download files from the internet. Here's an example:

ruby Copy code

```
$ wget https://www.example.com/sample.pdf
--2022-03-20 07:12:05--  https://www.example.com/sample.pdf
Resolving www.example.com (www.example.com)... 93.184.216.34
Connecting to www.example.com (www.example.com)|93.184.216.34|:443... connected.
HTTP request sent, awaiting response... 200 OK
```

CONNECTING SERVERS WITH SSH

SSH (Secure Shell) is a network protocol that provides a secure way to remotely access a Linux terminal. It allows users to securely log in to a remote machine over an unsecured network, such as the Internet. Here is a brief on remotely connecting to your Linux terminal using SSH with examples in an organizational real-world perspective:

- Install SSH server: First, make sure that the SSH server is installed on the remote machine. If it is not installed, you can install it using the following command:
  - sudo apt-get install openssh-server

- Configure SSH server: After installing the SSH server, it's important to configure it properly. Open the /etc/ssh/sshd_config file and make sure the following options are set correctly:
  - Port: specifies the port number on which the SSH server listens (default is 22)
  - PermitRootLogin: specifies whether root can log in using SSH (should be set to no)
  - PasswordAuthentication: specifies whether users can log in using a password (should be set to no and use SSH key authentication instead)

Connect to remote machine: To connect to the remote machine, open a terminal on your local machine and use the following command:

  - ssh username@remote-machine-ip

- Authenticate with SSH key: Once you run the above command, you will be prompted to enter the password for the user on the remote machine. However, it's more secure to use SSH key authentication. To use SSH key authentication, first, generate an SSH key pair on your local machine using the following command:
  - ssh-keygen

- Copy public key to remote machine: After generating the SSH key pair, copy the public key to the remote machine using the following command:
  - sh-copy-id username@remote-machine-ip

- Authenticate with SSH key: Now, when you connect to the remote machine using SSH, you will not be prompted for a password. Instead, you will be authenticated using your SSH key.

- Disconnect from remote machine: To disconnect from the remote machine, simply type exit or press Ctrl + D.

- In an organizational real-world perspective, SSH is commonly used to remotely access servers and network devices, such as routers, switches, and firewalls. It allows system administrators to perform maintenance tasks and troubleshoot issues without having to physically be at the location of the device. Additionally, SSH tunnels can be used to securely transfer data between two machines over an unsecured network.

SECTION 11
SCHEDULING AND AUTOMATION

CRONJOB

Cron is a time-based job scheduler in Linux that runs processes or scripts at specified intervals. Cron jobs can be used to automate repetitive tasks, such as backups, system updates, and data processing.

- Here are some presenting points and examples for scheduling tasks with Cron Jobs:

Cron job syntax: Cron job syntax consists of six fields that represent the timing of the job. The fields are Minute, Hour, Day of the Month, Month, Day of the Week, and Command. For example, the following job runs every day at 1 AM:

- 0 1 * * * /path/to/command

Scheduling a cron job: To schedule a cron job, use the crontab command followed by the -e option to edit the crontab file. Each line in the crontab file represents a single job. Here's an example of how to schedule a job to run every weekday at 5 PM:

- 0 17 * * 1-5 /path/to/command

Listing cron jobs: To list all the cron jobs for the current user, use the crontab command followed by the -l option:

- crontab -l

Removing a cron job: To remove a cron job, use the crontab command followed by the -r option:

- crontab -r

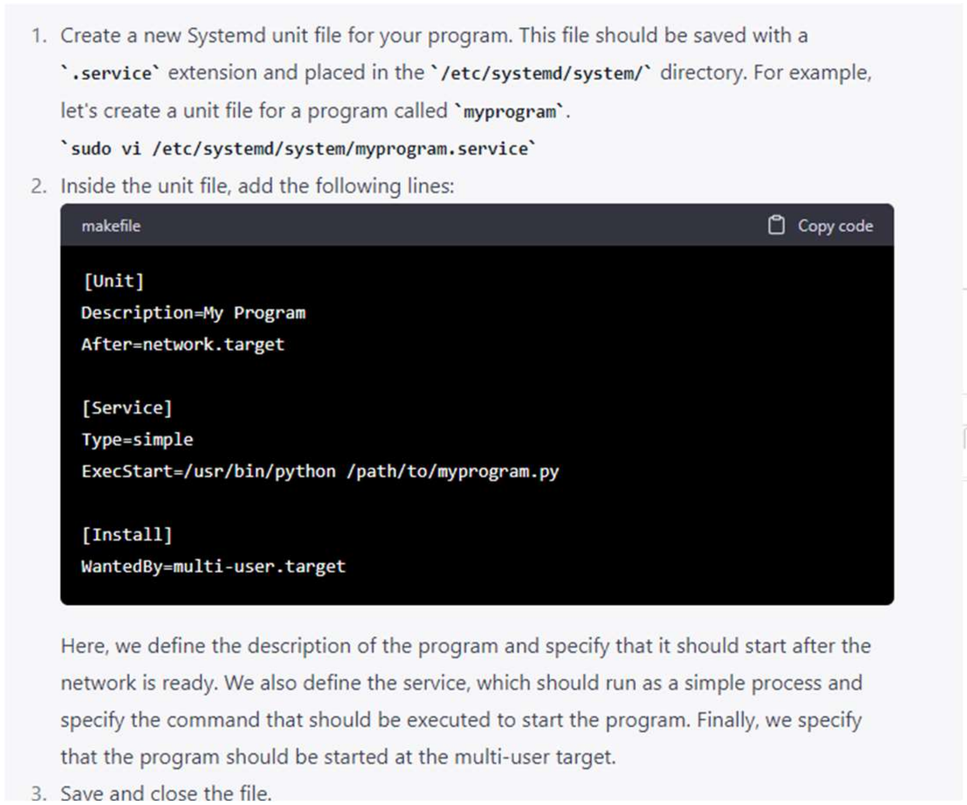
Troubleshooting cron jobs: If a cron job is not working as expected, check the system log file for error messages. You can also redirect the output of the command to a file to see if there are any error messages. For example, to redirect the output of a command to a file, use the following syntax:

- 0 1 * * * /path/to/command >> /path/to/output.log 2>&1

This will append the output and error messages to the specified log file.

Overall, Cron Jobs are a powerful tool for automating tasks on a Linux system. With the right configuration and scheduling, Cron Jobs can help streamline and simplify routine tasks, improving system efficiency and reliability.

BOOT WITH SYSTEMD



1. Create a new Systemd unit file for your program. This file should be saved with a `` `.service` `` extension and placed in the `` `/etc/systemd/system/` `` directory. For example, let's create a unit file for a program called `` `myprogram` ``.
   `` `sudo vi /etc/systemd/system/myprogram.service` ``
2. Inside the unit file, add the following lines:

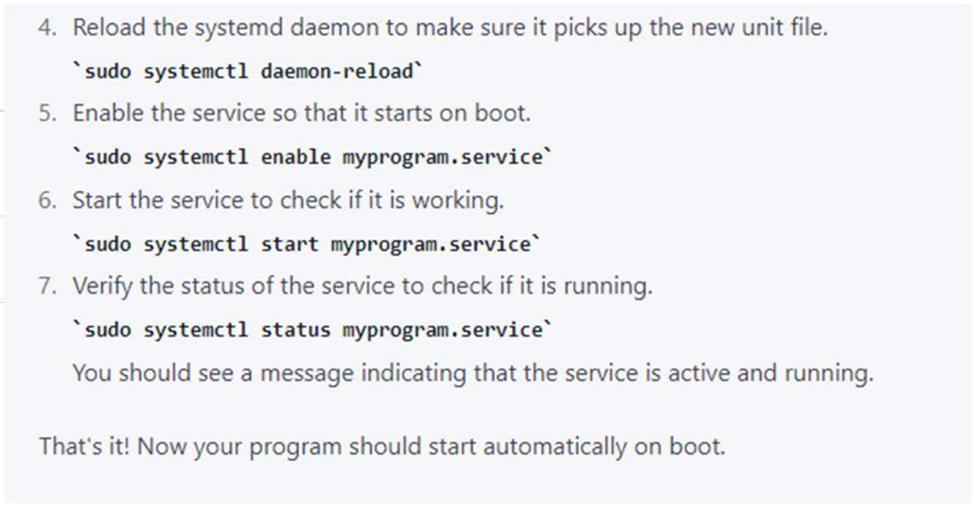
   makefile   Copy code

```makefile
[Unit]
Description=My Program
After=network.target

[Service]
Type=simple
ExecStart=/usr/bin/python /path/to/myprogram.py

[Install]
WantedBy=multi-user.target
```

   Here, we define the description of the program and specify that it should start after the network is ready. We also define the service, which should run as a simple process and specify the command that should be executed to start the program. Finally, we specify that the program should be started at the multi-user target.
3. Save and close the file.



4. Reload the systemd daemon to make sure it picks up the new unit file.
   `` `sudo systemctl daemon-reload` ``
5. Enable the service so that it starts on boot.
   `` `sudo systemctl enable myprogram.service` ``
6. Start the service to check if it is working.
   `` `sudo systemctl start myprogram.service` ``
7. Verify the status of the service to check if it is running.
   `` `sudo systemctl status myprogram.service` ``
   You should see a message indicating that the service is active and running.

That's it! Now your program should start automatically on boot.

SECTION 12
REALTIME USER MANAGEMENT

HOW USERS ARE CREATED IN REALTIME

- In real-world organizations with thousands of users, user management in Linux can be a challenging task. However, various tools and techniques can simplify this process.
- One common approach is to use a centralized authentication system like LDAP or Active Directory. With LDAP, all user credentials are stored in a centralized database, and Linux servers can authenticate users against this database. This way, users only need to be created, managed, and deleted in one place.
- Another approach is to use configuration management tools like Puppet, Chef, or Ansible. These tools can automate the process of creating and managing users across multiple servers by defining user accounts as code. Whenever a new server is added to the infrastructure, the configuration management tool can automatically create the required user accounts.
- In addition, Linux has several built-in tools to manage users, such as useradd, usermod, and userdel. These tools allow you to create, modify, and delete user accounts on a local Linux system.
- For security purposes, organizations can also enforce password policies that require users to create strong passwords and change them regularly. Moreover, two-factor authentication can be implemented to provide an additional layer of security to user accounts.
- Overall, managing users in Linux in a large-scale organization requires a combination of tools, policies, and best practices to ensure security, scalability, and ease of management.

LINUX LOGIN ISSUES IN REALTIME

- Troubleshooting Linux logins can involve several steps. Here are some common troubleshooting steps:

1. Verify that the correct username and password are being used: This may seem obvious, but it's important to ensure that the username and password being used are correct. It's easy to mistype a password, and sometimes usernames can be similar, leading to confusion.
2. Check for locked accounts: Linux systems can lock out user accounts if there are too many failed login attempts. Check if the account is locked and if so, unlock it.
3. Check for password expiration: If password aging is enabled, a user's password may expire. Check the user's password expiration settings and reset the password if necessary.
4. Check for network issues: If you're trying to log in remotely, check the network connection between the client and server. Ensure that the server is reachable and that firewalls are not blocking the connection.
5. Check the system logs: Check the system logs (e.g., /var/log/auth.log) for any error messages related to login attempts. This can provide valuable information about what might be causing the login issues.
6. Check the authentication settings: Ensure that the authentication settings are correctly configured. For example, if LDAP or Kerberos is being used for authentication, check that the settings are correct.
7. Check for filesystem issues: If there are filesystem issues on the server, this can cause login issues. Check the filesystem for errors and fix them if necessary.

By following these troubleshooting steps, you should be able to diagnose and resolve most Linux login issues. However, if the problem persists, it may be necessary to escalate the issue to a higher-level support team.

TROUBLESHOOTING NETWORK

Troubleshooting network issues in Linux involves identifying the root cause and applying appropriate solutions. The following steps, examples, and presenting points can help you diagnose and resolve common network problems:

Verify network interface status:

- Use the "ip link" or "ifconfig" command to check the status of your network interfaces.
- Example: $ ip link
- Look for the "UP" or "DOWN" status, and ensure that the relevant interface is "UP".
- If the interface is "DOWN", use the "ip link set <interface> up" or "ifconfig <interface> up" command to bring it up.

Test network connectivity:

- Use the "ping" command to check connectivity to your local network, gateway, and external hosts.
- Examples:
- $ ping 192.168.1.1 (gateway IP)
- $ ping 8.8.8.8 (Google DNS)
- If you can't reach external hosts, it may indicate an issue with your local network, gateway, or internet connection.

Test network connectivity:

- Use the "ping" command to check connectivity to your local network, gateway, and external hosts.
- Examples:
- $ ping 192.168.1.1 (gateway IP)
- $ ping 8.8.8.8 (Google DNS)
- If you can't reach external hosts, it may indicate an issue with your local network, gateway, or internet connection.

Check IP address and network configuration:

- Use the "ip addr" or "ifconfig" command to verify your IP address and network settings.
- Example: $ ip addr
- Ensure that you have a valid IP address, subnet mask, and gateway.
- If you have an incorrect or missing IP address, check your DHCP configuration or set a static IP address using the "ip addr add" or "ifconfig" command.

Verify DNS resolution:

- Test DNS resolution using the "nslookup" or "dig" command.
- Examples:
- $ nslookup google.com
- $ dig google.com
- If DNS resolution fails, check your DNS settings in "/etc/resolv.conf" or your network manager configuration.
- You can try using a public DNS server, such as Google DNS (8.8.8.8, 8.8.4.4) or Cloudflare DNS (1.1.1.1, 1.0.0.1).

Examine routing table:

- Use the "ip route" or "route" command to view your routing table.
- Example: $ ip route
- Ensure that the default gateway is properly configured.
- Check for incorrect or missing routes that might affect network connectivity.

Inspect firewall settings:

- Review your firewall configuration using tools like "iptables", "ufw", or "firewalld".
- Example: $ sudo iptables -L
- Look for rules that might be blocking network traffic.
- Temporarily disable the firewall to see if it resolves the issue. If it does, adjust the rules accordingly.

Analyze network traffic:

- Use tools like "tcpdump" or "wireshark" to capture and analyze network traffic.
- Example: $ sudo tcpdump -i <interface> -n
- Look for patterns that might indicate network issues, such as packet loss or high latency.

Check system logs:

- Examine system logs in "/var/log/" for clues related to network issues, such as "/var/log/syslog" or "/var/log/messages".
- Example: $ sudo tail -n 100 /var/log/syslog
- Look for error messages or unusual events that might indicate a problem.

Test network services:

- Use tools like "telnet", "nc" (netcat), or "nmap" to test the availability and responsiveness of network services on remote hosts.
- Examples:
- $ telnet <hostname> <port>
- $ nc -vz <hostname> <port>
- $ nmap -p <port> <hostname>
- If a service is unreachable or unresponsive, check the service configuration and logs on the remote host, and ensure that firewalls or security groups are not blocking the traffic.

Check for network congestion:

- Use tools like "mtr", "traceroute", or "ping" to identify network congestion or high latency.
- Examples:
- $ mtr <hostname>
- $ traceroute <hostname>
- Look for patterns that might indicate network congestion, such as high latency or packet loss between specific hops.

Review network hardware and infrastructure:

- Examine the configuration and logs of network devices, such as routers, switches, or wireless access points, for any issues or misconfigurations.
- Check for firmware updates, and ensure that devices are running the latest firmware versions.

Always keep in mind that network issues can be caused by various factors, such as hardware failures, software misconfigurations, or external network problems. It's essential to approach troubleshooting systematically and isolate the problem step by step. Remember to consult the documentation for your specific Linux distribution, network hardware, and software when troubleshooting network issues.

LINUX ADMINISTRATION DAILY TASKS

1. System Updates: Linux systems need to be regularly updated with the latest security patches and software updates. Updating the system can be time-consuming and may cause compatibility issues with existing applications. The solution is to use a package manager like yum or apt-get to automate the update process and test updates in a development environment before deploying them to production.
2. Network Issues: Linux systems need to be connected to a network to function properly. Network issues like slow connectivity, DNS resolution issues, or firewall blocking can cause disruptions in the system. The solution is to use network monitoring tools like ping or traceroute to diagnose network issues and configure firewalls and network settings to optimize performance.
3. Storage Management: Linux systems have complex storage requirements that need to be managed effectively. Issues like low disk space, storage failures, or corrupted file systems can cause data loss and system downtime. The solution is to use storage management tools like LVM or RAID to manage storage and monitor disk usage and configure backups and disaster recovery solutions.
4. User Management: Linux systems have multiple users with different roles and permissions. Issues like user authentication failures, password expiration, or unauthorized access can cause security breaches and data loss. The solution is to use user management tools like PAM or LDAP to manage user accounts and enforce security policies like password complexity and two-factor authentication.
5. Application Management: Linux systems have a variety of applications with different dependencies and configurations. Issues like application crashes, missing libraries, or outdated dependencies can cause application failures and system instability. The solution is to use application management tools like Docker or Kubernetes to manage applications and dependencies and automate application deployment and scaling.

By using a combination of monitoring and automation tools, Linux administrators can effectively manage and maintain Linux systems and prevent or remediate daily issues like system updates, network issues, storage management, user management, and application management.

RESUME

Rakesh A

Contact Information:

Email: trietreeinfo@gmail.com

Phone: 6305598849

- Summary:
  - Experienced Linux Administrator with 5+ years of experience managing and maintaining Linux systems. Proficient in system administration tasks, patch management, network management, and security. Strong problem-solving skills with a track record of achieving business objectives.
- Daily Tasks:
  - Manage and maintain Linux systems to ensure high uptime and performance.
  - Monitor system logs and performance metrics to detect issues and prevent downtime.
  - Perform patch management to ensure systems are up to date with security patches and software updates.
  - Manage user accounts and permissions to enforce security policies and access controls.
  - Configure firewalls and network settings to optimize network performance and security.
  - Manage backups and disaster recovery solutions to prevent data loss and ensure business continuity.
  - Troubleshoot issues and provide timely resolutions to minimize disruptions to users.

- Achievements:
  - Improved system uptime from 95% to 99.9% by implementing proactive monitoring and automation tools.
  - Reduced security breaches by 50% by enforcing security policies and implementing two-factor authentication.
  - Optimized network performance by configuring firewalls and network settings, resulting in a 30% reduction in network latency.
  - Automated patch management tasks using Ansible, resulting in a 40% reduction in patching time.
  - Successfully managed backups and disaster recovery solutions, resulting in no data loss and minimal downtime during system failures.
- Education:
  - Bachelor's Degree in Computer Science, XYZ University (2015-2019)
- Certifications:
  - Linux Professional Institute Certification (LPIC-1)
  - Red Hat Certified System Administrator (RHCSA)
- Skills:
  - Linux system administration
  - Patch management
  - Network management
  - Security
  - Automation tools (Ansible, Puppet)
  - Scripting (Python, Bash)
  - Virtualization (VMware, KVM)
  - Cloud computing (AWS, Azure)