1. Linux

GNU/Linux is the result of combining the Linux kernel, which is responsible for managing hardware, with the tools and utilities from the GNU project, which provide essential functionalities to make this operating system usable. For this reason, it is called GNU/Linux. GNU/Linux distributions are complete operating systems based on the Linux kernel. These include GNU tools and additional components such as package managers, graphical environments, and specific applications. These customizations allow each distribution to be optimized for different niches or uses. Among the distributions are:

- Debian: Known for its stability and use on servers.
- Slackware: Geared toward advanced users looking for minimalist environments.
- Arch Linux: Focused on simplicity and customization.
- Red Hat Enterprise Linux (RHEL): Oriented toward enterprise environments and servers.
- Ubuntu: A distribution based on Debian.

Linux Structure

The file system in Linux is a hierarchical structure that organizes the system's data and resources, starting from the root, represented as /. From the root, several directories branch out, housing different types of data and configurations. These files can be contained in partitions, which are essentially sections within the storage unit where the system is installed. These partitions are separated from one another, allowing for better information organization.
Linux supports various file system types in terms of format, such as:

- EXT4 (Extended Filesystem 4): One of the most widely used due to its reliability and performance.
- XFS: Optimized for handling large volumes of data, primarily ideal for enterprise environments.
- BTRFS: Designed for snapshots, high scalability, and advanced features like integrated compression.
  The key elements of a file system are:
- Inodes: Store metadata information (permissions, owner, size, etc.) about a file or directory.
- Blocks: Storage units where file data is kept.
- Superblock: Contains crucial information about the file system. If the superblock is damaged, the system can be difficult to recover.
  In Linux, there are two types of links:

- Hard links: Direct references to the same inode, meaning they point to the same file content. If the file is deleted, the content remains accessible as long as hard links exist.
- Soft or symbolic links: Point to the file path rather than its content. If the original file is deleted, the symbolic link is broken.

What is a Command?

A command is an instruction entered by the user through the terminal for the system to perform a specific task. Commands are interpreted by the shell, which translates these orders into actions executable by the system. Each command can be associated with a set of predefined instructions that, when executed, allow the user to interact with the system and perform operations such as managing files, querying system information, or running programs.

Shell vs. Terminal Emulator

A terminal is simply the software that provides the interface for data input and output, within which a shell can be executed. In the past, the only way to give instructions to the operating system was through a terminal. Nowadays, things have changed — all commonly used operating systems now have graphical interfaces that make them more accessible to the general public. The easier an operating system is to use, the more people can adopt it.
On the other hand, terminal emulators essentially mimic a physical terminal, and within them, a shell can be executed.
A shell is a program whose primary function is command interpretation. Through this command interpretation, it uses system resources to achieve the action desired by the user. The shell can be seen as a bridge between the user and the operating system, enabling the user to issue commands to the system. The most common shells include:

- sh: The basic shell, generally included in most UNIX systems.
- bash: An improved version of sh, offering additional features like command history, autocompletion, and advanced scripting.
- zsh: A modern shell that combines features from bash and other shells like ksh, with enhancements in interactivity, advanced autocompletion, and plugins.

Introduction to the Linux Prompt

It is commonly referred to as the command prompt or the indicator that we are in a shell within the terminal emulator. The basic structure of Bash is as follows:
user@host:path$

- user: The name of the active user.
- @: Separates the username from the hostname.
- host: The name of the system or machine.

- :: Separates the hostname from the path.
- path: The current directory where the user is located.
- $ or #: The final character that indicates the permission level:
  - $ for regular users.
  - # for the superuser (root).

Important Directories in Linux

The Linux file system follows a hierarchical structure with a single root represented by /. From this root, the system directories are organized as follows:

/
├── boot/ - Files necessary for system boot (kernel, GRUB, etc.).
├── etc/ - System and application configuration files.
├── var/ - Variable data files (logs, databases, print queues, etc.).
│ ├── log/ - System log files.
│ ├── spool/ - Print and mail queues.
│ └── tmp/ - Persistent temporary files.
├── usr/ - Shared files for users and applications.
│ ├── bin/ - Executables for common applications.
│ └── lib/ - Shared libraries.
├── home/ - Personal directories for users.
├── root/ - Home directory for the system administrator (root).
├── tmp/ - Temporary files that are deleted upon reboot.
├── dev/ - Special files representing hardware devices.
├── sbin/ - Essential executables for system administration and recovery.
├── proc/ - Virtual file system with information about processes and system status.
├── sys/ - Virtual file system representing system hardware.
├── opt/ - Additional or third-party software.
├── mnt/ - Temporary mount points for file systems.
├── media/ - Mount points for removable devices such as USB drives or CDs/DVDs.
└── lib/ - Essential shared libraries for /bin/ and /sbin/.

Important Files in Linux

The file system not only organizes directories but also contains specific files with critical roles for the proper functioning and configuration of the system.

/
├── etc/
│ ├── passwd - Information about system users.
│ ├── group - Information about system groups.
│ ├── shadow - Encrypted user password storage.

│ ├── fstab - File system mount configuration (usually edited manually).
│ ├── network/
│ │ └── interfaces - Network interface configuration (in older distributions).
│ ├── hostname - System hostname (usually edited manually).
│ ├── resolv.conf - DNS server configuration (usually edited manually).
│ └── crontab - System's scheduled task configuration.
├── var/
│ ├── log/
│ │ ├── syslog - Main system log.
│ │ ├── auth.log - Authentication and access logs.
│ │ └── dmesg - Kernel logs during system boot.
│ └── spool/
│ └── cron/ - Pending scheduled tasks.
├── boot/
│ └── grub/
│ └── grub.cfg - GRUB bootloader configuration.
├── proc/
│ ├── cpuinfo - Information about the system CPU.
│ ├── cmdline - Kernel parameters passed at boot time.
│ ├── meminfo - Information about system memory.
│ ├── modules - List of currently loaded kernel modules.
│ └── uptime - System uptime.
├── home/
│ └── user_name/
│ ├── .bashrc - Bash shell configuration for a user.
│ └── .ssh/
│ ├── id_rsa - User's private SSH key.
│ └── authorized_keys - Public keys authorized for SSH connection.
├── root/
│ └── .bashrc - Bash shell configuration for the root user.
├── usr/
│ ├── bin/
│ │ └── top - `top` command executable for system monitoring.
│ └── share/
│ └── man/ - System manual pages.
└── dev/
├── null - Special device for discarding data.
├── sda - Primary hard drive (block device).
└── tty - Terminal devices.

Environment Variables in Linux

Environment variables are dynamic values that influence how processes run within the system. These variables allow applications to obtain information about the environment or set default configurations for their execution. They can be queried, defined, and manipulated using specific commands:

- Viewing environment variables:
  - env: Displays all available environment variables.
  - echo $VARIABLE_NAME: Shows the value of a specific variable.
- Defining a new local variable:
  - Variable=directory_path:$Variable: Adds a new path to the selected variable.
- Defining a new environment variable:
  - export VARIABLE_NAME=value: Creates or modifies an environment variable in the current session.
- Deleting an environment variable:
  - unset VARIABLE_NAME: Removes the specified environment variable.

  Important environment variables:
- PATH: Defines the directories where the system will search for executables when commands are run.
- HOME: Specifies the current user's home directory.
- USER: Contains the name of the logged-in user.
- SHELL: Indicates the shell being used in the current session.
- TERM: Defines the type of terminal in use, useful for adjusting terminal-based application settings.
- EDITOR: Sets the default text editor for programs like crontab or git.
- LANG: Specifies the system's language and localization settings.

Absolute and Relative Paths

Paths are used to locate files and directories within the filesystem. These are divided into absolute and relative paths:

1. Absolute Paths:
   - Represent the location of a file or directory in relation to the root of the system (/).
   - Always start with the / character.
   - They are independent of the user’s current working directory.
2. Relative Paths:
   - Indicate the location of a file or directory in relation to the current working directory.
   - Never start with the / character.
   - They depend on the directory where the user is currently working.

3. Special Characters for Navigation:
    - . (Dot):
        - Represents the current directory.
        - It is used to refer to the directory the user is currently in.
        - When a file or directory name starts with a dot (.), it is considered hidden.
    - .. (Double Dot):
        - Represents the parent directory, meaning the directory immediately above the current one

Understanding Standard Streams

Data input and output between programs, commands, and users are managed through three main streams, known as Stdin, Stdout, and Stderr. These standards are essential for efficient inter-process communication and user interaction.

- Stdin (Standard Input):
    - The standard stream used to receive data that a program needs to process. By default, it's associated with the keyboard, but it can be redirected from other sources like a file or the output of another command. It is represented by the number 0 in the system.
- Stdout (Standard Output):
    - The standard stream where a program sends the results of its execution. By default, it's associated with the terminal emulator, but it can be redirected to files or to the standard input of another program. It is represented by the number 1 in the system.
- Stderr (Standard Error):
    - The standard stream used for error messages or diagnostics generated by a program. By default, it is also associated with the terminal emulator, though it can be redirected to a separate file or combined with the standard output. It is represented by the number 2 in the system.

What is a file descriptor?

It is a numeric identifier used by the system to represent input and output resources, such as files, devices, or data streams. In UNIX and Linux systems, each running process has three default open file descriptors:

- 0 (Stdin): Standard input.
- 1 (Stdout): Standard output.
- 2 (Stderr): Standard error output.
  Its main functions are:
- Allowing programs to interact efficiently and uniformly with external resources.

- Identifiers can be manipulated to redirect input and output to different destinations, such as files, devices, or even other processes.
  Usage in command control:
- Debugging programs: Separates normal output from errors to facilitate problem identification.
- Automation: Makes it possible to store data for later analysis or send information to other processes.
- Flexibility: Using commands like `grep`, `awk`, or `sed`, file descriptors enhance advanced data processing.

Redirections in Linux

These are tools that allow controlling the flow of input and output of commands, sending or capturing data to files or between commands.
Redirection operators:

1. *(>) Overwriting standard output*: Redirects the standard output of a command to a file, overwriting existing content.
2. *(>>) Appending to a file*: Adds the standard output to the end of the file.
3. *(|) Pipe*: Allows chaining multiple commands, passing the standard output of one command as the standard input of the next.

Differences Between Arguments and Parameters

Commands can be customized or configured using parameters and arguments. Although often used interchangeably, they have specific roles:

1. Parameters:
   - These are options or indicators that modify the behavior of a command.
   - They are generally specified with:
     - Single hyphen (-): Followed by a letter.
     - Double hyphen (--): Followed by a complete word.
   - They allow activating additional functions or changing how the command is executed.
2. Arguments:
   - These are values or data that a command needs to perform a specific action.
   - They indicate the elements on which the command should act.
   - They often appear after the parameters.
3. Key Differences:
   - Parameters alter the behavior of the command.
   - Arguments specify on what data the command should act.

Logical Operators

- && (AND): Allows executing a second command only if the first one executes successfully.
- || (OR): Allows executing a second command only if the first one fails.
- ! (NOT): Inverts the result of the execution of a command.
- ; (Sequence): Allows executing several commands sequentially, regardless of whether the previous ones succeed or fail.

Glob Characters

They are a feature of the shell, not something specific to any particular command. As a result, they can be used with any existing command in Linux.

- Asterisk (*): Matches zero or more characters in a file or directory name.
- Question mark (?): Matches exactly one character in a file name.
- Brackets ([ ]): Matches any character within the specified set.
- Single quotes ('): Prevent the shell from interpreting all special characters. "Protects" a string from being changed by the shell.
- Double quotes ("): Allow variable expansion and command substitution, but protect other special characters such as spaces or globs.
- Backticks (`` ` ``): Allow command substitution, executing the command within the shell and replacing it with its output.
- Backslash (\ ): Escapes a special character to treat it literally instead of interpreting it.

Basic Regular Expressions

A collection of "normal" and "special" characters used to match a simple or complex pattern. Basic regular expressions are as follows:

Regular Expression	Matches
Period (.)	Any single character.
Brackets ([ ])	A list or range of characters that match one character.
Asterisk (*)	The previous character that is repeated zero or more times
Caret (^)	The following text must appear at the beginning of the line.
Dollar ($)	The preceding text must appear at the end of the line

Linux Commands

The syntax for commands in Linux is: [Command (Options) (Arguments)]
Options: Options can be used to modify the behavior of a command. Arguments: An argument can be used to specify something the command should act upon.

Information and Help Commands

- man: The first line of help to understand how other commands work.
  -k: Displays a summary of all manual pages related to the assigned keyword.
  -f: Shows the manual pages that match the specified name.
- info: Similar to the man command but with a different format, sometimes providing more detailed information than man.
- whatis: Provides a very brief description of the command being queried.
- apropos: Helps find commands related to a keyword.
- whereis: Searches for commands, source files, and man pages in specific locations where these files are stored.
- which: Shows the path of the executable file associated with a command.
- type: Determines information about various commands; it can also identify built-in commands in bash or other shells and recognize command aliases.
  -a: Reveals the path of another command.
  -t: Indicates the type of command (executable, function, alias, etc.)
- help: Provides information only on built-in commands for the shell in use.
- alias: Allows viewing the aliases defined in the shell.

File or Directory Search Commands

- find: Searches for files in the filesystem by name or using wildcards if the exact name is unknown. It can also search based on file metadata like type, size, and ownership.
  -name: Searches for a file by name.
  -ls: Displays file details.
  -size (+/- Size to search): Searches for files by size.
  -iname: Case-insensitive file name search.
  -user: Returns files owned by a specific user.
- grep: Filters lines in a file or the output of another command based on pattern matching, using either exact text or regular expressions.
  -i: Makes the search case-insensitive.
  -c: Counts the number of lines containing the search pattern.
  -n: Shows the original line numbers.
- locate: Primarily finds the location of files within the system. To update the locate database, use sudo updatedb.

Terminal Navigation Commands

- cd: Stands for Change Directory; used to switch the current directory, accepting both relative and absolute paths.
  Double dot (..): Represents the parent directory above the current directory.
  Single dot (.): Refers to the current directory.
- pwd: Stands for Print Working Directory; prints the current working directory.
- ls: Stands for List; lists directories and files with important options such as:
  -a: Shows all files, including hidden ones.
  -l: Displays everything in a long format.
  -R: Recursively lists directories and their contents.
  -h: Displays file sizes in a human-readable format (KB, MB, GB).
  -d: Refers to the current directory, not its contents.
  -S: Sorts files by size.
  -t: Lists files based on their modification time.
- history: Stores a history of executed commands. A command from the history can be executed using an exclamation mark (!) followed by the desired command's number.

File Management Commands

- touch: Primarily used to create empty files.
- cp: Stands for copy; used to copy files and directories. Syntax: cp (source) (destination). Important options:
  -i: Prompts for confirmation before copying a file.
  -r: Recursively copies directories.
  -v: Displays the command's output.
- mv: Stands for move; used to move and rename files and directories. Syntax: mv (source) (destination). Important options:
  -i: Prompts for confirmation before moving a file.
  -u: Overwrites only if the destination file is older.
- rm: Stands for remove; used to delete files and directories. Syntax: rm (item to delete). Important options:
  -i: Prompts for confirmation before deleting a file.
  -r: Recursively deletes directories.

File Viewing Commands

- cat: Stands for concatenate. It was originally designed to concatenate and display the content of files, but nowadays it's mostly used to read relatively small files directly in the terminal. Syntax: `cat (file path)`. Its parameter is:
  -n: Numbers the lines while reading a file.
- head: Displays the first 10 lines of a file. Unlike "cat", which would show the entire file. With the parameter:

-n: Specifies the number of lines to read.

- **tail**: Displays a file starting from the last 10 lines. With the parameter:
  -n: Specifies the number of lines to read.
  -f: Follows the content of a file in real-time.
- **less**: Displays a file's content in a paginated manner, allowing you to scroll page by page. Its parameters are:
  -n: Displays line numbers when viewing a file.
  -f: Follows the file's content in real-time.
- **sort**: A command used to sort lines of text in a file, either alphabetically or numerically. It also supports sorting based on specific columns, making it especially useful for processing and organizing large datasets. Common parameters:
  -f: Ignores case sensitivity when sorting.
  -o: Saves the result into a file.
  -n: Sorts numerically.
  -t: Sorts based on a specific field separator.
  -c: Checks if the file is already sorted.
  -k: Specifies the field number for sorting.
  -r: Performs a reverse sort.

Directory Management Commands

- **mkdir**: Stands for make directory, and its function is to create directories within the file system. Syntax: mkdir (directory path). Its parameter is:
  -p: Creates parent directories as needed.
- **rmdir**: Stands for remove directory and only works for deleting empty directories. Its parameter is:
  -r: Removes non-empty directories.
- **ln**: Creates a hard link to a file or directory. Among its parameters:
  -s: Creates symbolic links.

User Management

- **adduser**: This command is used to create new users on the system.
- **deluser**: This command is used to delete a user from the system. Its most common parameter is:
  –remove-home: Deletes the user and also their assigned home directory.
- **usermod**: Allows modifying user properties such as username, home directory, or assigned group. Its main parameters are:
  -l: Changes the user's login name.
  -d: Changes the user's home directory.

-L: Locks the user's account.
-U: Unlocks the user's account.

- **passwd**: Used to change a user's password. It supports certain parameters:
  -l: Locks the user's password.
  -u: Unlocks the user's password.
- **id**: Stands for identification and shows information about the current user or a specified user. Its parameters are:
  -u: Shows only the user's UID (User ID).
  -g: Shows the user's GID (Group ID).
- **who**: Shows who is currently connected to the system. It supports several parameters like:
  -u: Displays connected users along with their connection time.
- **w**: Provides a more detailed list of users currently on the system.
- **su**: Stands for switch user and allows changing the current user. If no username is specified, it defaults to trying to switch to the root user. It has several parameters: *-l* or *--login*: Provides an environment similar to what the user would have if they logged in directly. *-c*: Allows executing a command as the specified user without starting a full session as that user.
- **sudo**: Executes commands with the privileges of another user. It supports several parameters like:
  -u: Specifies the user under which the command should be executed.
  -i: Provides an environment similar to the one the user would have if they logged in.
  -s: Starts a shell as the specified user.
- **whoami**: Displays the username associated with the current user ID in the terminal.
- **finger**: Provides information about a particular user.

Group Management

- **groupadd**: Allows creating groups within the system; it must be executed by the root user. Some of its parameters are:
  -g: Used to specify an ID for the new group.
- **groupdel**: Allows deleting groups; if the group has a user in it, the user must be removed first.
- **groupmod**: Allows modifying the name of a group and has several parameters: *-n*: Allows changing the group name.
  -g: Allows changing the group's GID.
  -aG: Adds a user to a group.
  -G: Sets the user's supplementary group.
- **gpasswd**: Used to manage group passwords and group members. It has several parameters such as:

-a: Adds a user to a group.
-d: Removes a user from a group.

- **groups**: Displays the groups a user belongs to.
- **newgrp**: Changes the primary group the user belongs to.

Network Management

Linux uses two types of naming conventions: traditional names (non-persistent and not very descriptive) and predictive names (persistent and descriptive, based on the NIC's physical location or hardware topology).

- **ifconfig**: Lists all active network interfaces, showing details such as the IP address, netmask, packet statistics, and more. Some of its parameters are:
  up: Turns on a network interface.
  down: Turns off a network interface.
- **ip**: A tool to display and manipulate routes, devices, routing policies, and tunnels. Some of its parameters are:
  a or *addr*: Displays all network interfaces with their IP addresses.
- **ip route**: Displays the kernel's routing table, essentially showing the routing paths for packets. Some parameters include:
  add + *via*: Adds a new route.
- **netstat**: Displays routing tables, interface statistics, masqueraded connections, and multicast members. It has several parameters:
  -t: Shows TCP connections.
  -u: Shows UDP connections.
  -l: Displays only listening ports.
  -p: Shows the PID.
  -e: Displays extended information.
  -i: Shows network traffic statistics.
  -r: Displays routing information.
  -n: Displays numbers instead of service names.
- **ping**: Verifies connectivity with other hosts. It has several parameters:
  -c: Specifies the number of packets to send.
- **traceroute**: Shows the route packets take to reach a destination host. Its parameters include:
  -m: Specifies the number of hops before the command stops tracing.
- **nslookup**: Queries the DNS server. It has several parameters:
  -type: Specifies the type of record to search.
- **hostname**: Displays and configures the system's hostname.
- **nc**: Stands for netcat, a tool mainly used to open and close ports. It has several parameters: *-l*: Puts the tool in listening mode on a port.

-v: Enables verbose mode to display more details.
-p: Specifies a port.
-n: Prevents DNS resolution.
-u: Uses the UDP protocol instead of the default TCP.

- **dig**: Tests the DNS server functionality of the host being used by performing DNS queries to check if the required information is available.

DEB Package Management

- **apt**: It comes natively in Debian-based distributions, so it won't be found natively in other distros. It stands for Advanced Packaging Tool. It simplifies the installation, updating, and removal of packages, as it automatically resolves the necessary dependencies to install a package. For proper operation, this command must be run with sudo privileges. It has several options:
  update: Updates the list of available packages and their versions.
  upgrade: Installs the latest versions of the packages installed on the system.
  full-upgrade: Performs a full system upgrade.
  install: Installs packages.
  remove: Removes installed packages from the system.
  autoremove: Removes packages that were installed as dependencies and are no longer needed.
  --purge remove: Deletes all files of a software package, including configuration files.
- **apt-get**: An older but more stable version of **apt** and shares its parameters. It has additional options like:
  dist-upgrade: Besides performing the functions of the upgrade parameter, it also intelligently handles changes in dependencies with new package-level versions.
- **apt-cache**: Searches and retrieves information about packages stored in the apt cache. For proper operation, this command must be run with sudo privileges. Some of its options include:
  depends: Shows the dependencies of the mentioned package.
  policy: Displays the installed policy of the mentioned package, including versions and which repository the package will be installed from.
  search: Searches the system's repositories for the mentioned package.
- **dpkg**: Stands for Debian Package; it is the low-level package management system for Debian-based distros, which only manages specific **.deb** packages. Its options include: *-i*: Installs a package.
  -l: Lists the packages installed on the system.
  -r: Removes a package but keeps its configuration files on the system.
  -p: Completely removes the package.
  -L: Lists the files that make up a specific package.

- **dpkg-query**: Used to query the dpkg database.
- **dpkg-reconfigure**: Used to reconfigure a package after it has been installed.
- **dpkg-trigger**: Triggers events in the package system.

RPM Package Management

- **rpm**: It is the low-level package management system for Red Hat packages. It queries information through a local database on the machine. Its options include:
  -qa: Lists all currently installed packages on the system.
  -ql: Lists the files that make up a specific package.
  -qi: Queries a package and retrieves information or its status.
- **yum**: Automates the process of resolving dependency issues. If adding, updating, or removing packages, root privileges are required. Its options include:
  search: Searches for packages from the configured repositories.
  install: Installs packages along with their dependencies.
  update: Updates all packages.
  remove: Removes a package, resolving any dependency issues.

Web Package Management

- **wget**: Allows downloading files from the internet, supporting main protocols like HTTP, HTTPS, or FTP. Its parameters include:
  -c: Allows resuming an interrupted download.
  --no-check-certificate: Ignores TLS certificate verification.
- **curl**: Transfers data to or from a server using various protocols and is mainly used for API testing, downloading files, or sending data. Parameters include:
  -X: Specifies the HTTP request method.
  -H: Adds headers to the request.

Compressed File Management

- **tar**: Creates compressed files. Its parameters include:
  -c: Indicates the creation of a file.
  -z: Uses gzip for compression.
  -v: Displays the processed files.
  -f: Specifies the output file name.
  -x: Indicates file extraction.
  -t: Lists the documents in the packed file.
  -j: Decompresses with bzip2 before reading.
- **gzip**: Compresses individual files, adding the .gz extension. Parameters include:
  -d: Decompresses files.

-l: Displays compression information.

- zip: Compresses files with the .zip extension.
- unzip: Decompresses .zip files.

Process Management

Each process in Linux has several key elements. Every process has a unique PID (Process Identifier). There are different types of processes: user processes, initiated and controlled by users, and system processes, initiated and managed by the OS and essential for its functioning. Processes are assigned states like running, sleeping, stopped, or zombie.

- ps: Displays information on active processes. Parameters include:
  a: Shows processes for all users.
  u: Displays detailed user information.
  x: Includes processes without an associated terminal.
  aux or *-ef*: Shows all system processes.
- pstree: Displays processes in a family tree format, showing parent and child relationships.
- top: Shows real-time running processes and their resource consumption.
- free: Displays the current memory usage.
- kill: Sends signals to a process, often used to terminate it.
- killall: Terminates all processes associated with a given name.
- free: Displays system memory status, including physical and swap memory. Parameter: *-b*: Displays information in bits.
- fg: Brings a background process to the foreground.
- bg: Resumes a suspended process, running it in the background.
- jobs: Lists suspended or background processes.
- disown: Removes jobs from the list, allowing a previously launched process to run independently of the terminal.

For Ownership and Permissions Management

- chown: Allows changing the owner and group of a file or directory, and must be used by the root user. A parameter that can be used is:
  -R: Applies changes to an entire directory structure.
- chgrp: Stands for Change Group, and only allows changing the group ownership of a file or directory. It must also be used by the root user. A parameter that can be used is:
  -R: Applies changes to an entire directory structure.
- stat: Displays detailed information about the permissions and ownership of directories.
- chmod: This command allows modifying the permissions of directories, files, and other system assets, and it can do so either in octal or symbolic form. A parameter that can be

used is: *-R*: Applies permissions recursively at the directory level.

Additional Commands

- echo: Prints various types of values to the screen. Its parameters are:
  -n: Does not print a new line at the end of the text.
  -e: Enables the interpretation of escape sequences like \n for a line break and \t for a tab.
- du: Shows the disk space used by files and directories. It can use the following parameter:
  -a: Displays the usage of all files.

File Editors from the Terminal

Vim/NVim

It's an improved version of the "vi" editor, which is one of the most widely used editors in the Linux community. To open a file with vim, you'd use the `vim` command followed by the file path. If the file doesn't exist, it will create one with the assigned path. It offers multiple modes like:

Normal Mode

The default mode when opening a file. You can use keys and commands to navigate the file and perform actions like deleting text, copying, pasting, etc. However, you cannot directly insert text in this mode. Useful keys for normal mode:

1. Basic Navigation:
   - h: move left.
   - j: move down.
   - k: move up.
   - l: move right.
2. Advanced Navigation:
   - w: jump to the start of the next word.
   - e: jump to the end of the current word.
   - b: jump back to the start of the previous word.
3. Text Manipulation:
   - dd: delete a line.
   - yy: copy a line.
   - p: paste what's been copied or deleted.
   - u: undo the last action.
   - Ctrl + r: redo the last undone action.

Insert Mode

Allows you to add text to the file. To enter this mode from normal mode:

- Press i to insert text before the cursor.
- Press a to insert text after the cursor.
- Press o to insert a new line below the cursor.
  To exit this mode, press Esc.

Command Mode

To access this mode, press : (colon). Here, you can execute advanced commands for saving, exiting, searching, and performing other operations.

1. Save and Exit:
   - :w: save changes.
   - :q: quit (only if no changes are pending).
   - :q!: quit without saving.
   - :wq or :x: save and exit.
2. Search:
   - :/word: search "word" forward.
   - :?word: search "word" backward.
   - n: jump to the next result.
   - N: jump to the previous result.

Nano

Nano is a text editor like Vim but much simpler to use. It's mainly used from the terminal and is ideal for quickly editing files without a steep learning curve.

Common options when opening Nano

When using Nano, you can apply certain parameters to modify its behavior. These include: The syntax is: [nano (parameter) file_name]

- -m: Enables mouse support for interacting with text.
- -l: Displays line numbers.
- -v: Opens the file in read-only mode, preventing accidental modifications.
- -B: Creates a backup copy of the file before saving changes.
- -E: Removes unnecessary trailing spaces at the end of lines.
- -i: Automatically indents new lines.

Linux Shortcuts

Command Line Shortcuts

- Ctrl+a: Moves to the beginning of the line.
- Ctrl+e: Moves to the end of the line.
- Ctrl+b: Moves the cursor one character back.
- Ctrl+f: Moves the cursor one character forward.
- Alt+b: Moves the cursor one word back.
- Alt+f: Moves the cursor one word forward.
- Ctrl+xx: Toggles between the beginning of the line and the current cursor position.

Process Management Shortcuts

- Ctrl+c: Terminates the current process.
- Ctrl+z: Suspends the current process.
- Ctrl+d: Exits the terminal if no text input is present.

Command History Shortcuts

- Ctrl+p: Navigates to the previous command in history.
- Ctrl+n: Navigates to the next command in history.
- Ctrl+r: Searches within command history.
- Ctrl+o: Executes a command from the history. Complements Ctrl+r.

Terminal Emulator Shortcuts

- Ctrl+l: Clears the terminal screen.
- Ctrl+s: Pauses terminal output scrolling.
- Ctrl+q: Resumes terminal output scrolling.
- Ctrl+Alt+(F1-F6): Switches between TTY terminals.

Aliases in Linux

This is a way of assigning a keyword to a custom command. In other words, they allow you to execute a command with certain parameters and arguments, but only through a keyword. The alias command displays the commands that have been used; to create an alias, use alias (alias name = command). Keep in mind that aliases are temporary; if the system is rebooted or shut down, the created aliases are deleted. To make them permanent, add them to the .bashrc file. To delete an alias, use unalias + alias name.

File Types

File types can be identified in the first field when listing files with the ls command with the -l parameter.
The file types are:

Symbol	File Type	Description
d	Directory	A file used to contain other files.
-	Ordinary file	Includes readable files, images, binary files, and compressed files.
l	Symbolic links	Points to another file.
s	Socket	Allows interprocess communication.
p	Pipe	Allows interprocess communication.
b	Block file	Used for hardware communications.
c	Character file	Used for hardware communications.

Permissions

There are several types of permissions that can be assigned to both files and directories, namely read (read), write (write), and execute (execute). Permissions can be assigned in two ways: octal or symbolic.

Octal Permission Assignment

Each permission type has a numerical value:

- r = 4, w = 2, x = 1
  They are assigned by combining the values for each user category:
- Owner (u): User who created the file/directory.
- Group (g): Users who belong to the file/directory's group.
- Other (o): All other users.
  Example:
  chmod 754 filename
- 7 (owner): r + w + x = 4 + 2 + 1 = 7
- 5 (group): r + x = 4 + 1 = 5
- 4 (others): r = 4

Symbolic Permission Assignment

Letters are used to define permissions and operators are used to assign them:

1. Permissions:

- r = read, w = write, x = execute.

2. Categories:

- u = owner, g = group, o = other, a = everyone.

3. Operators:

- - (add), - (remove), = (set exact)
    Examples:
    chmod g+w filename -> Add write permissions to the group.
    chmod g-w filename -> Remove write permissions from the group.

Other Permissions

SUID or Set User ID

- Function:
- Allows a file to be executed as if it were the owner, regardless of who executes it.
- Can only be assigned to executable files.
- Symbol: (s) replaces the (x) in group permissions.
- Octal: Represented by a 4 *Example*: chmod 4755 filename

SGID or Set Group ID

- Function:
- On files: They are executed as if they belonged to the file's group.
- On directories: Created files inherit the directory's group.
- Symbol: (s) replaces the (x) in owner permissions.
- Octal: Represented by a 2 *Example*: chmod 2755 filename

Stickt Bit

- Function:
- Restricts manipulation of files in shared directories, allowing only the owner or administrator to delete or modify them.
- Only applies to directories.
- Symbol: (t) in the permissions of others.
- Octal: Represented by a 1 *Example*: chmod 1755 filename.

Permissions Summary

Permission	Octal	Symbol
Read (r)	4	r
Write (w)	2	w
Execute (x)	1	x
SUID (owner)	4	s
SGID (group)	2	s
Sticky Bit (others)	1	t