

Introduction

Welcome to the beginning of our journey into the Linux operating system. In this section, we aim to introduce the fundamental concepts, history, significance of Linux in the modern computing landscape, and the core principles underpinning this powerful operating system.

What is Linux?

Linux is a free and open-source operating system (OS) developed by Linus Torvalds in 1991 and continues to evolve. Since then, Linux has become a global phenomenon, powering everything from supercomputers to servers, mobile phones to personal computers. Known for its stability, security, and flexibility, Linux is a popular choice for both personal and professional use.

The Philosophy of Linux

Linux embodies the spirit of collaboration and freedom. Its development reflects the following principles:

- **Freedom:** Linux gives users the freedom to control, modify, and redistribute their software. This fosters an environment of innovation and security.
- **Collaboration:** Thousands of developers worldwide contribute to Linux, keeping it sharp, secure, and responsive to user needs.
- **Transparency:** The open-source nature of Linux provides a level of transparency unmatched by proprietary systems.

How Does Linux Work?

Linux is an operating system similar to Windows or macOS but differs in how it works and its free, open-source nature. At its core lies the Linux kernel, which is the central component of the system. The kernel is responsible for managing the computer's hardware, such as the CPU, memory, and peripherals, and allows all software applications to interact with the physical hardware.

The kernel acts as a bridge between software applications and the computer hardware. When a software application wants to do something hardware-related, like save a file or display something on the screen, it sends a request to the kernel. The kernel translates this request into instructions that the hardware can understand.

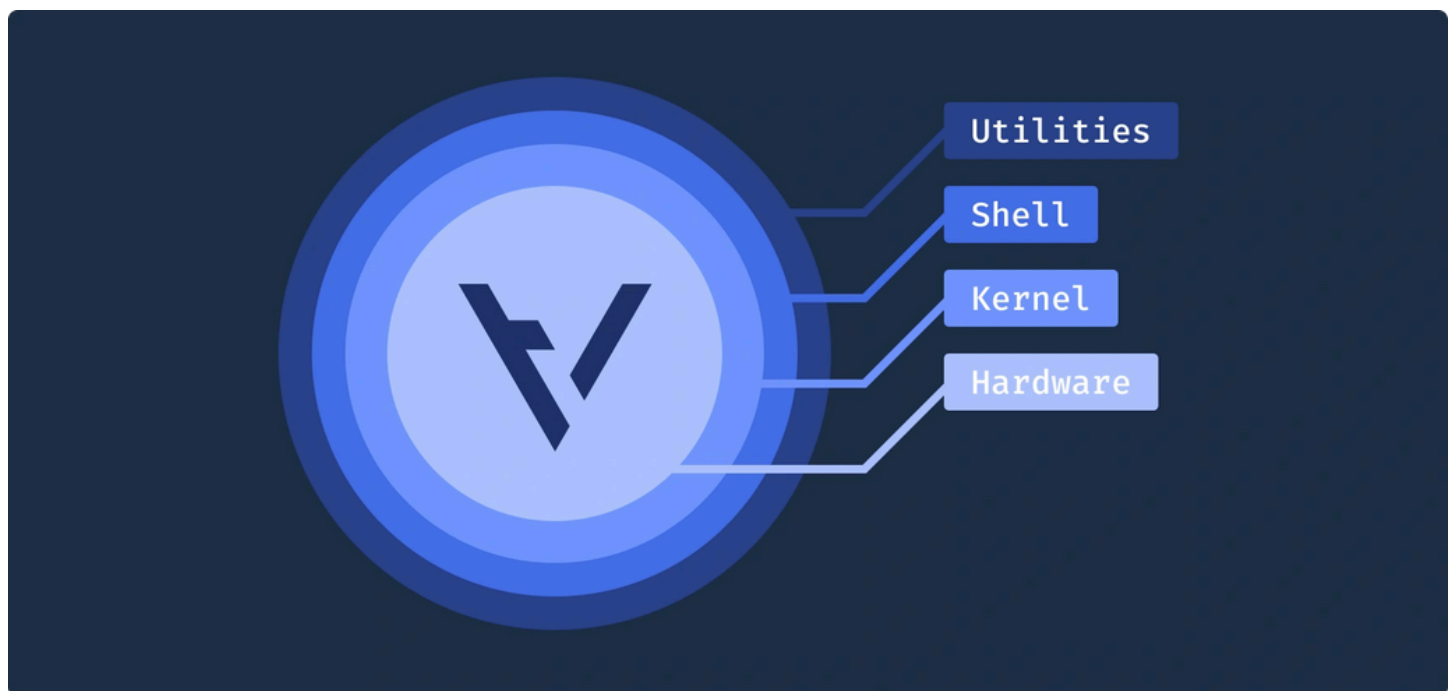
Linux supports multi-tasking and multi-user environments. This means multiple users can use the system simultaneously, and each can run multiple programs at the same time. This is especially useful in server environments where many people need to access the same system for different tasks.

The Linux file system is hierarchically organized, starting from the root directory (denoted as "/") and expanding into sub-directories. This organization makes it easy to manage and locate files.

Around the kernel, there are many software tools and libraries that add extra functionality. These can include graphical user interfaces (GUIs), system utilities, and application software. Users can mix and match these components to create a Linux distribution that meets their specific needs. Linux distributions package the kernel with a selection of software to provide a complete operating system ready for use. Popular Linux distributions include Ubuntu, Fedora, and CentOS.

Linux Architecture

The Linux operating system is composed of several layers that manage the computer's resources and facilitate user interaction:



1. **Hardware Layer:** This is the physical foundation of the system, composed of the computer's processor (CPU), memory (RAM), storage (hard disks), and peripherals such as keyboards, mice, and printers.
2. **Kernel Layer:** The kernel is the heart of Linux and is vital to the operating system. It acts as an intermediary between software and hardware. The kernel manages tasks such as memory allocation, process scheduling (deciding when and what tasks the CPU should execute), and handling input/output requests from software. This layer ensures that different programs and users running on the system do not interfere with each other and have the necessary resources to operate.
3. **Shell Layer:** The shell is the user interface for accessing the services of the kernel. It is commonly a command-line interface (CLI) where users type commands, but graphical shells also exist. The shell allows users to interact with the kernel to run programs,

manage files, and request other services by typing commands or using a graphical interface.

4. **System Utility Layer:** This layer contains various tools and applications necessary for performing tasks on the computer. System utilities can range from file management tools to software installers, network configuration tools, and more. They serve as a bridge between the user's commands (entered in the shell or through a graphical interface) and the kernel's handling of those commands.

In summary, Linux architecture organizes the interaction between the computer's hardware and user activities from physical components to software applications through a well-defined management and control layer.

Linux Distributions

Linux is not a single operating system but a family of open-source operating systems built around the Linux kernel. These variants are known as "distributions" or "distros." Each distribution offers a different flavor of Linux tailored for various types of users, devices, and purposes. This section explores the concept of Linux distributions, their importance, and examples of popular distros.

What are Linux Distributions?

A Linux distribution is a complete operating system built around the Linux kernel. It includes the kernel, a wide array of software applications, libraries, and optionally a graphical user interface (GUI). Distros are developed by various organizations, communities, and even individuals, focusing on different needs such as usability, stability, security, or customization.

Why Are There Different Distributions?

The diversity of Linux distributions stems from the philosophy of freedom and flexibility. Different users have different needs, ranging from personal computers to servers, from beginners to experts, and from lightweight systems for older hardware to feature-rich environments for modern machines. Distributions meet these varied requirements by offering specialized software packages, user interfaces, and management tools.

Core Components of a Distribution

While each distribution offers unique features and software, they all share several core components:

- **Linux Kernel:** The heart of the operating system, managing hardware resources.
- **System Libraries:** Essential for running applications, providing standard ways to interact with the kernel.
- **Software Applications:** Ranging from system utilities to end-user applications.

- **Package Manager:** A tool for installing, updating, and managing software applications and components.
- **Bootloader:** Software that manages the boot process of the computer after power is turned on.
- **Graphical User Interface (GUI):** Optional, but most distros offer a GUI for ease of use, such as GNOME, KDE, or XFCE.

Popular Linux Distributions

Ubuntu

- **Target Audience:** Beginner and regular desktop users.
- **Features:** User-friendly, strong focus on ease of installation and usability. Extensive documentation and community support.
- **Use Cases:** Desktops, servers, and cloud environments.

Fedora

- **Target Audience:** Developers and system administrators.
- **Features:** Cutting-edge technology, close alignment with the Linux community, strong emphasis on free software.
- **Use Cases:** Development, servers, and workstations.

CentOS

- **Target Audience:** Servers and enterprise users.
- **Features:** Stability and long-term support, binary compatibility with RHEL (Red Hat Enterprise Linux).
- **Use Cases:** Servers, enterprise environments.

Debian

- **Target Audience:** Advanced users and environments focusing on stability.
- **Features:** Large package repository, strict free software guidelines, stability.
- **Use Cases:** Servers, desktops, embedded systems.

Arch Linux

- **Target Audience:** Experienced users and enthusiasts.
- **Features:** Rolling release model, user-centric approach focusing on customization and minimalism.
- **Use Cases:** Desktops, development, personal servers.

Kali Linux

- **Target Audience:** Security professionals and ethical hackers.
- **Features:** Preloaded with a wide range of tools for penetration testing, security research, computer forensics, and reverse engineering.
- **Use Cases:** Security research, penetration testing, and ethical hacking.

Choosing the Right Distribution

Choosing the right distribution depends on your specific needs, level of expertise, and preferences. Consider the following factors:

- **Purpose:** Desktop use, server, development, etc.
- **Ease of Use:** Is it beginner-friendly or requires Linux expertise?
- **Support and Community:** Access to documentation, forums, and community support.
- **Hardware Compatibility:** Supports your hardware, especially for older or specialized equipment.
- **Software Availability:** Availability of the software packages you need.

In conclusion, Linux distributions offer a wide range of choices for users of all levels and purposes. By understanding the unique features and target audiences of each distro, you can select the one that best meets your needs.

Linux Shell

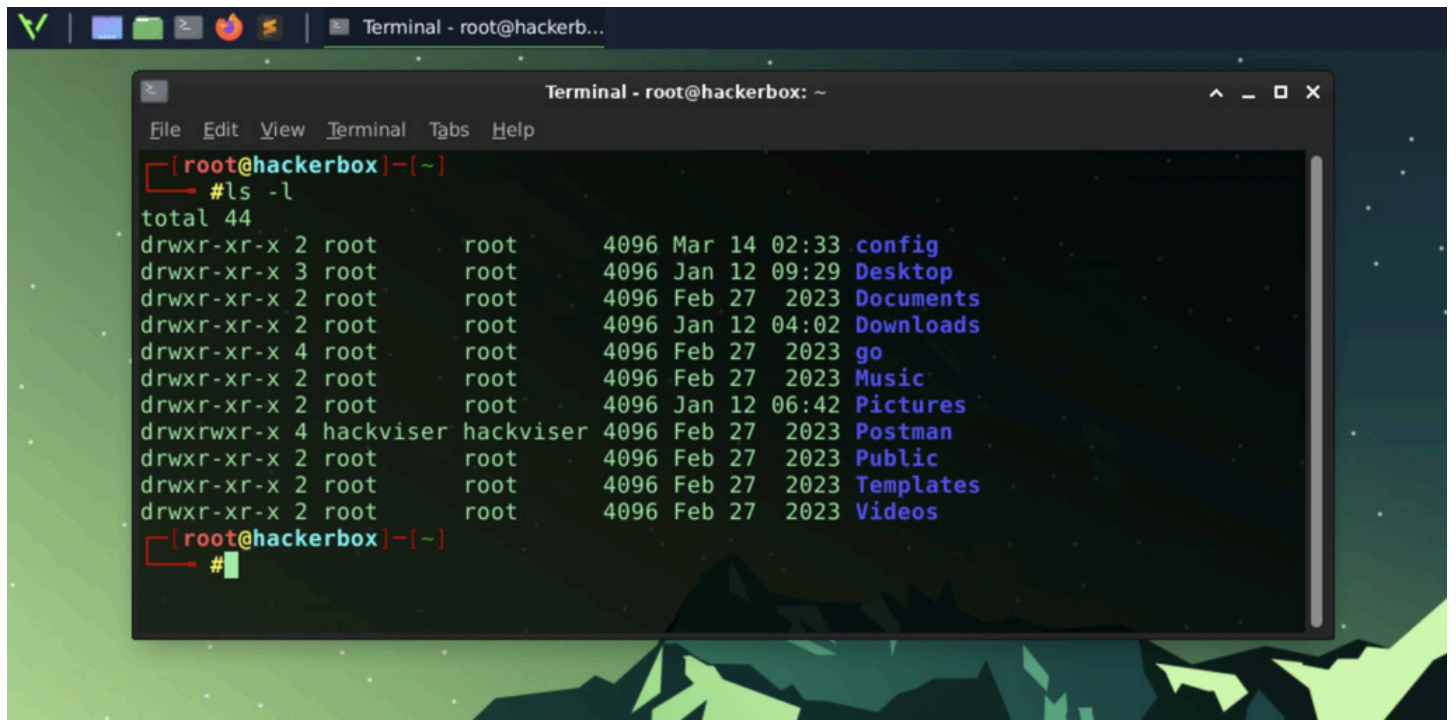
The Linux shell is a unique program that acts as a bridge between the user and the services of the operating system. It takes commands entered in a human-readable format, such as text typed on a keyboard or stored in a file, and translates them into a form that the operating system's kernel can process. The shell functions essentially as a command-language interpreter, executing these instructions. It becomes active automatically when users log into their accounts or start a terminal session.

In Linux, there are two main types of shells: **Command Line** and **Graphical Interface**.

Command Line

The command-line shell is accessible to users through a text-based interface. Users can enter commands into a specific application known in Linux as the Terminal. Commands like `cat`, `ls` and others are written into this interface, processed, and executed. The results of these commands are displayed directly in the terminal window.

A terminal on Linux looks like this:



```
Terminal - root@hackerbox: ~
File Edit View Terminal Tabs Help
[root@hackerbox]~
#ls -l
total 44
drwxr-xr-x 2 root    root    4096 Mar 14 02:33 config
drwxr-xr-x 3 root    root    4096 Jan 12 09:29 Desktop
drwxr-xr-x 2 root    root    4096 Feb 27 2023 Documents
drwxr-xr-x 2 root    root    4096 Jan 12 04:02 Downloads
drwxr-xr-x 4 root    root    4096 Feb 27 2023 go
drwxr-xr-x 2 root    root    4096 Feb 27 2023 Music
drwxr-xr-x 2 root    root    4096 Jan 12 06:42 Pictures
drwxrwxr-x 4 hackviser hackviser 4096 Feb 27 2023 Postman
drwxr-xr-x 2 root    root    4096 Feb 27 2023 Public
drwxr-xr-x 2 root    root    4096 Feb 27 2023 Templates
drwxr-xr-x 2 root    root    4096 Feb 27 2023 Videos
[root@hackerbox]~
#
```

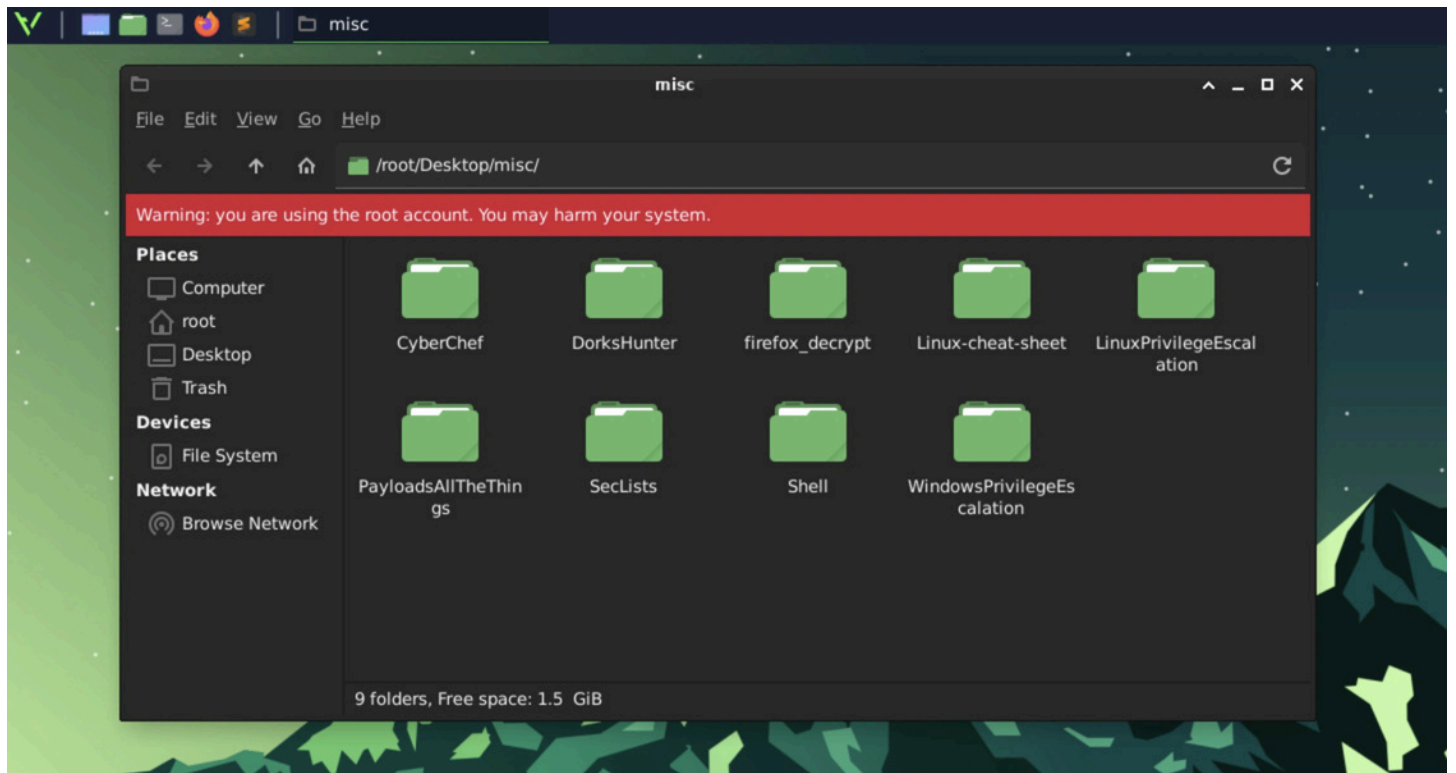
In the given screenshot, the `ls` command is executed with the `-l` option. This command lists all files in the current directory in a long format.

Using the command line can be challenging at first because remembering commands might be difficult. However, its power lies in its flexibility and strength; users can compile scripts and execute them simultaneously, making it easy to automate repetitive tasks. In the Linux context, these scripts are often referred to as shell scripts.

Graphical Interface

The graphical interface offers a user-friendly way to interact with programs. It allows users to perform actions like opening, closing, moving, and resizing windows without needing to type commands. Operating systems like Windows or Ubuntu provide a graphical interface to simplify user interaction with the system.

A graphical interface on a typical Linux system looks like this:



Linux operating systems offer various shells, each with unique features and command syntax:

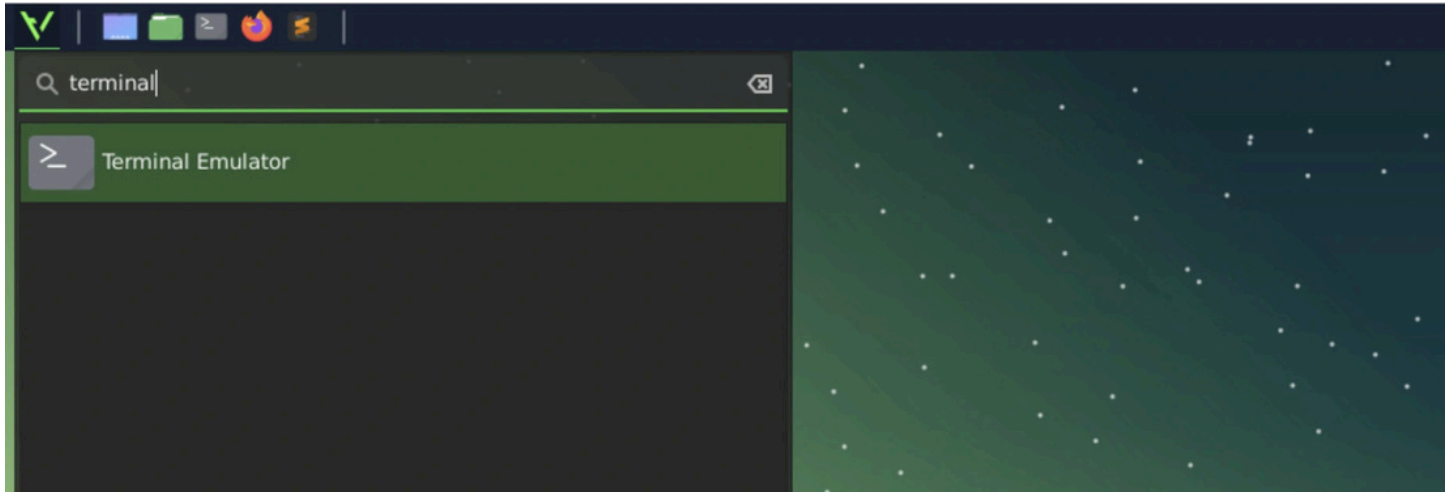
- **BASH (Bourne Again SHell)**: The most widely used Linux shell, serving as the default shell on Linux and macOS. BASH is known for its comprehensive scripting capabilities and extensive support.
- **CSH (C Shell)**: The syntax and usage of C Shell are similar to the C programming language, making it appealing for users familiar with C.
- **KSH (Korn Shell)**: The Korn Shell, which forms the basis for the POSIX Shell standard features, combines elements from BASH and CSH, offering powerful scripting abilities and command-line editing features.
- **ZSH (Z Shell)**: ZSH incorporates features from BASH, KSH, and TCSH, offering robust auto-completion functions, customizable prompts, and numerous plugins and themes through frameworks like Oh My Zsh.
- **Fish (Friendly Interactive SHell)**: Known for its user-friendly interface, Fish provides features like syntax highlighting, auto-suggestions, and tab completions without requiring additional configuration.

While all these shells perform the basic task of command interpretation, they vary in the specific commands they support, built-in functionalities, and scripting capabilities, allowing users to choose the shell that best fits their preferences and needs.

What is a Terminal?

The terminal application serves as a gateway for users to interact with the shell, providing a text-based interface where commands can be entered and their outputs viewed. It is a critical tool for executing commands directly and is also used to run larger scripts designed to automate and perform repetitive and complex tasks.

To open the terminal, you can usually find it by typing 'terminal' into the search box in the graphical interface and launching it by double-clicking the result.



Navigating Directories

In a typical graphical interface (desktop environment), you can browse directories using a mouse, but on the Linux terminal, you need to use various commands to navigate, interact with files, and directories.

This section is designed to teach the basics of Linux navigation, including navigating directories, listing files, managing files, and using shortcuts for efficiency.

Linux File System Structure

The Linux file system is hierarchically organized, starting with the root directory `/`. Understanding this structure is key to effectively navigating in Linux.

- `/`: The root directory, the base of the file system.
- `/bin`, `/sbin`: Contains essential user and system binaries.
- `/etc`: Contains system configuration files.
- `/home`: Contains personal directories for users.
- `/var`: Contains variable files like logs and databases.
- `/usr`: Secondary hierarchy for user data; contains most user programs and utilities.

Getting Started with Linux Navigation

First, it's important to know which directory/location you're in. The `pwd` (print working directory) command will show you your current location in the file system.

```
user@hackerbox:~$ pwd
```

```
/home/user
```


In the example above, the user is in the `/home/user` directory. The home folder contains individual home directories for users. Each user typically has a home directory, and the terminal starts in the user's home directory when it is first opened.

To list the files and folders in your current directory, you can use the `ls` command.

```
user@hackerbox:~$ ls
```

```
Desktop Documents Downloads Music Pictures Videos
```

The above example shows the output of the `ls` command. As seen, there are 6 folders in the current location. You can detail the output of the `ls` command using the `-l` option, which shows details such as permissions, ownership, and modification times.

```
user@hackerbox:~$ ls -l
```

```
total 8
```

```
drwxr-xr-x 2 user users 4096 Jul 29 08:24 Desktop
```

```
drwxr-xr-x 2 user users 4096 Jul 29 08:24 Documents
```

```
drwxr-xr-x 2 user users 4096 Jul 29 08:24 Downloads
```

```
drwxr-xr-x 2 user users 4096 Jul 29 08:24 Pictures
```

```
drwxr-xr-x 2 user users 4096 Jul 29 08:24 Videos
```

The output with the `-l` option includes columns with the following structure:

Column Content	Description
drwxr-xr-x	File type and permissions
2	Number of hard links to the file/directory
user	Owner of the file/directory
users	Group owner of the file/directory
4096	Size of the file or blocks used to store the directory information
Jul 29 08:24	Creation or last modification date of the file/directory
Desktop	Name of the file/directory

We have listed the contents of the directory, but there might be hidden files/directories we don't see. In Linux, files and directories starting with `.` are considered **hidden files** (e.g., `.bashrc` file). These won't be listed by default with the `ls` command. To include hidden files in the list, use the `-a` option with `ls`.

```
user@hackerbox:~$ ls -a
```

```
total 12
```

```
Desktop Documents Downloads Music Pictures Videos      .bashrc
```

As seen, the previously hidden

```
.bashrc
```

file is now listed in the output of the `ls` command. You can combine the `-l` and `-a` options with `ls -la`.

```
user@hackerbox:~$ ls -la
```

```
total 12
```

```
drwxr-xr-x 3 user user 4096 Aug  1 10:00 .
```

```
drwxr-xr-x 4 user user 4096 Aug  1 09:58 ..
```

```
drwxr-xr-x 2 user users 4096 Jul 29 08:24 Desktop
drwxr-xr-x 2 user users 4096 Jul 29 08:24 Documents
drwxr-xr-x 2 user users 4096 Jul 29 08:24 Downloads
drwxr-xr-x 2 user users 4096 Jul 29 08:24 Pictures
drwxr-xr-x 2 user users 4096 Jul 29 08:24 Videos
-rw----- 1 user users 220 Jul 29 09:58 .bashrc
```

As seen, combining the `-l` and `-a` options allows for both detailed listing and inclusion of hidden files.

Navigating to Other Directories

You don't need to be in a directory to list its contents. You can provide the path to a directory as a parameter to the `ls` command:

```
user@hackerbox:~$ ls -l /var/log
```

```
total 2097152
```

```
drwxr-x--- 2 root root      4096 Aug  1 06:25 apache2
```

To change your current directory, use the `cd` (change directory) command:

```
user@hackerbox:~$ cd /tmp
```

```
user@hackerbox:/tmp$
```

If you wish to go back to the previous directory, simply type `cd -`.

```
user@hackerbox:/tmp$ cd -
```

```
/home/user
```

Another feature you should know about while navigating directories is auto-completion. It speeds up your navigation and prevents typos.

Type `cd /usr/s` and press the `Tab` key twice, which will suggest directories starting with "s" in the `/usr/` location, allowing you to effortlessly write the path you want to navigate to.

File and Directory Operations

In Linux's powerful command-line interface, there are numerous commands to work with files and directories. In this section, you will learn how to create, copy, move, delete, and

search for files and directories. Additionally, we will look at some basic commands used to change file and directory permissions.

Creating Files and Directories

In Linux, the `touch` command is used to create files. This command can also be used to update the access and modification times of existing files, but if the file specified does not exist, it will create a new empty file.

```
user@hackerbox:~$ touch readme.txt
```

The command above creates an empty file named `readme.txt` in your current directory.

To create directories, use the `mkdir` (make directory) command.

```
user@hackerbox:~$ mkdir documents
```

This command creates a new directory named `documents`.

Copying Files and Directories

To copy files, the `cp` command is used. The general usage of the command is `cp source_file target_file`.

```
user@hackerbox:~$ cp readme.txt readme_copy.txt
```

This command creates a copy of `readme.txt` named `readme_copy.txt`. When copying directories, you must use the `-r` (recursive) option, which ensures that all subdirectories and files within the directory are also copied.

```
user@hackerbox:~$ cp -r documents documents_copy
```

This command copies the `documents` directory and all its contents to a new directory named `documents_copy`.

Moving or Renaming Files and Directories

To move or rename files and directories, use the `mv` command. This command can move files and directories to a new location or rename them.

```
user@hackerbox:~$ mv readme_copy.txt readme_moved.txt
```

This command renames the file `readme_copy.txt` to `readme_moved.txt`.

```
user@hackerbox:~$ mv readme_moved.txt documents/
```

This command moves `readme_moved.txt` to the `documents` directory.

Deleting Files and Directories

To delete files, use the `rm` (remove) command.

```
user@hackerbox:~$ rm readme_moved.txt
```

This command deletes the file `readme_moved.txt`. To delete directories, you can use the

`rm -r` command. This command deletes the directory along with its contents.

```
user@hackerbox:~$ rm -r documents
```

This command deletes the `documents` directory and its contents.

Methods of Finding Files and Directories in Linux

File and directory search operations in Linux enable users to navigate their file systems effectively. In this section, we'll cover how to find files and directories using the `find`, `locate`, and `which` commands. These commands offer suitable solutions for different scenarios and needs.

Searching with the `find` Command

The `find` command is used to search for files and directories based on specific criteria. With its ability to perform in-depth searches, it produces effective results in large and complex file systems.

Basic Usage

```
find [search path] [search criteria] [action]
```

- **Search by Name:** Use the `-name` option to search by file name.

```
find / -name "notes.txt"
```

- **Search by Type:** Use the `-type` option to search by file type (e.g., regular files `f`, directories `d`).

```
find /home/user -type d -name "Project*"
```

- **Search by Size:** Use the `-size` option to search for files by their size.

```
find / -size +50M
```

- **Search by Modification Time:** Use `-mtime`, `-atime`, or `-ctime` to filter files by their time attributes.

```
find / -mtime -7
```


Searching with the locate Command

The locate command is used to find files on the system. locate uses a database called updatedb, which contains an index of files on your system, and the locate command quickly searches this database.

- **Basic Usage:**

```
locate notes.txt
```

The speed of the locate command's results depends on the periodic updates of the file system by updatedb. Therefore, it may not find very recently created files.

Finding Executable Files with the which Command

The which command is used to obtain information about the path of a specific command. This is especially useful for understanding which version of a program is being used when multiple versions are installed.

- **Basic Usage:**

```
which python
```

This command shows the path of the python executable. It is commonly used to find out the location of the default python version on the system.

Summary

File and directory search operations in Linux can be easily performed using the find, locate, and which commands. The find command is ideal for in-depth, criteria-based searches. The locate command is useful for quickly obtaining results but may not provide the most current information. The which command is used to locate executables on the system. These tools allow Linux users to navigate their file systems effectively and find the files and programs they need.

Text Editing

Linux provides a range of powerful tools for creating, editing, and viewing text files. In this section, we will cover one of the most popular text editors and tools accessible from the command line: nano.

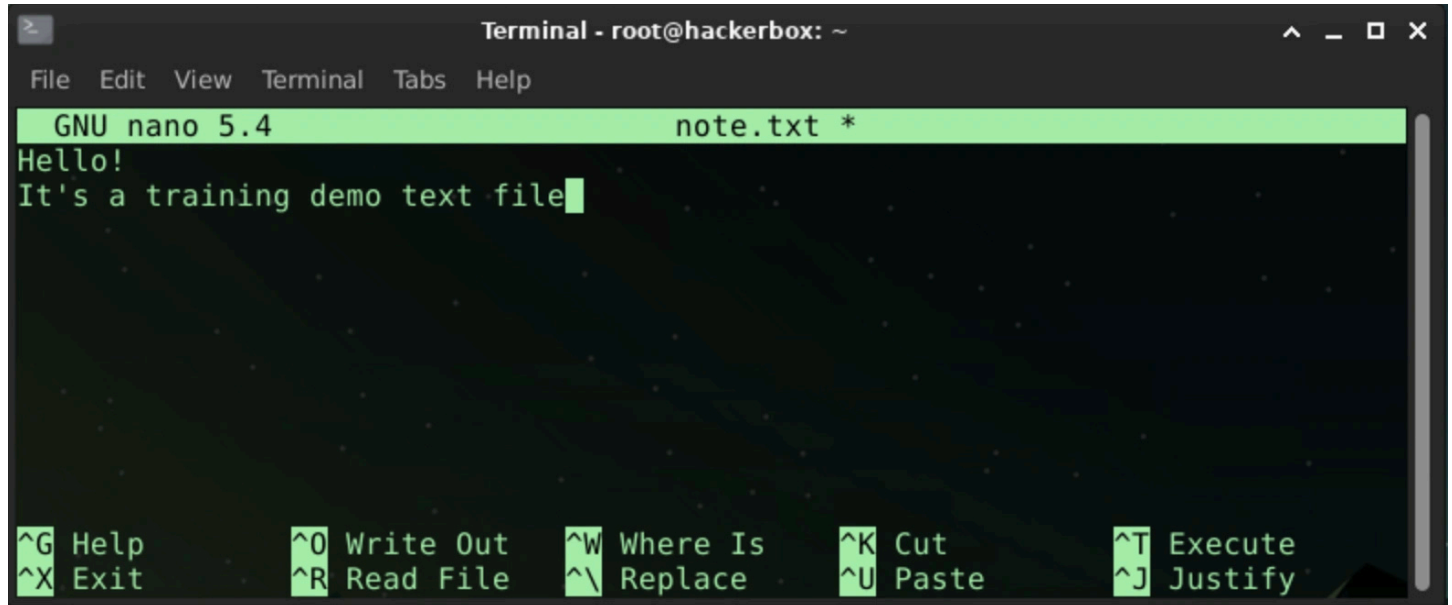
Nano Text Editor

Nano is one of the most widely used editors on Linux-based systems. It is a simple yet effective text editor that comes pre-installed with many Linux distributions. No prior knowledge about the nano editor is required to use it. You don't need to use commands to perform operations on a file in nano; all basic operations are displayed at the bottom of the

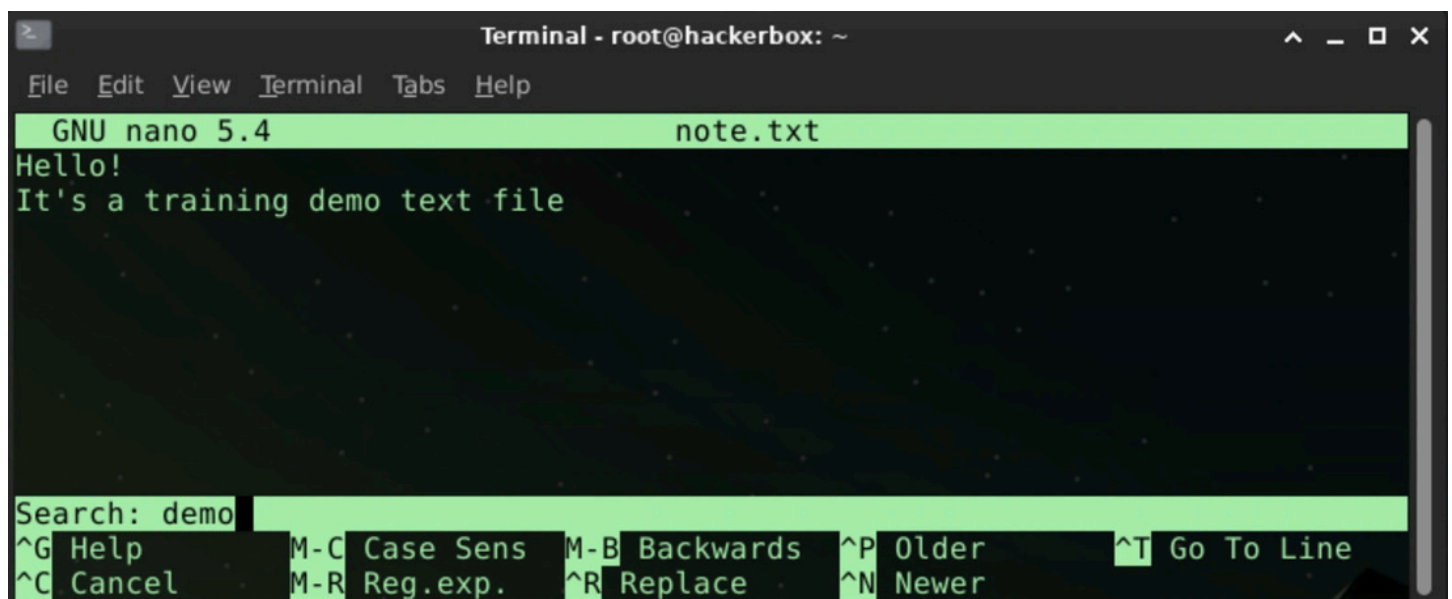
editor, which can be triggered using the CTRL key. For example, to save a file, you press CTRL+O, and to exit the editor, you press CTRL+X. To edit a file with the nano editor, run the following command:

```
root@hackerbox:~$ nano note.txt
```

The above command opens the file note.txt with the nano editor. You can move the cursor and enter the desired text to edit the file.

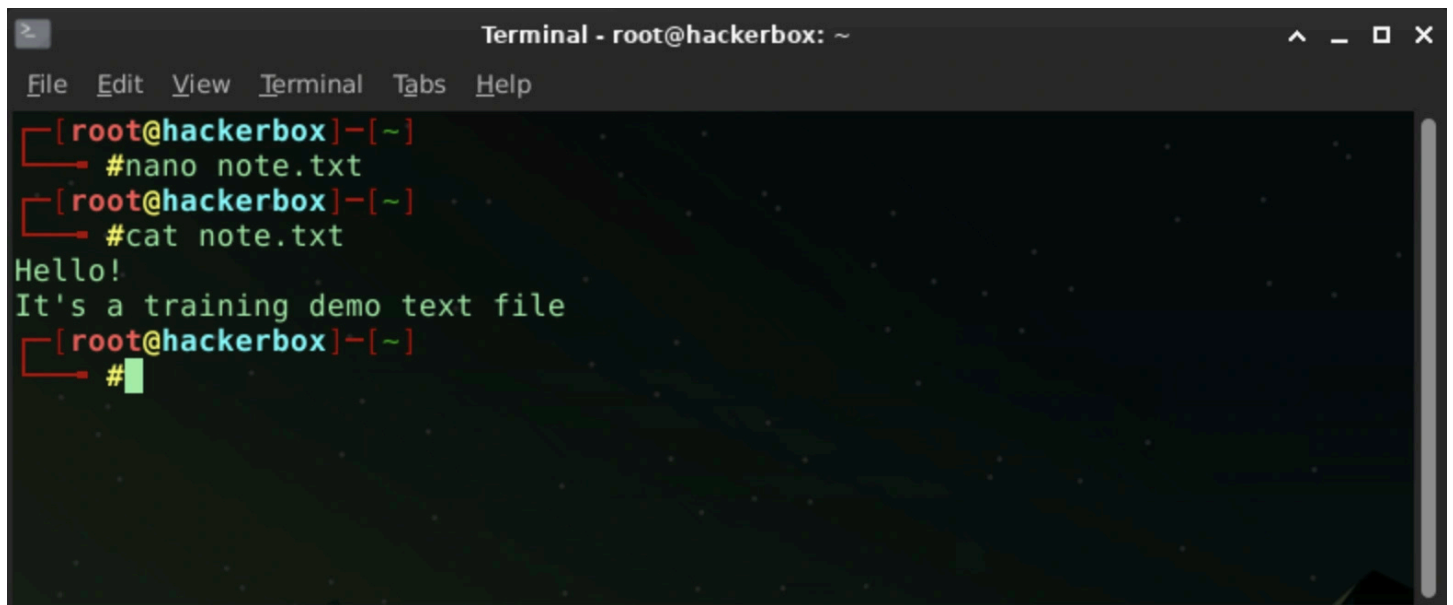
A screenshot of a terminal window titled "Terminal - root@hackerbox: ~". The terminal shows the nano 5.4 text editor editing a file named "note.txt". The editor's status bar at the top indicates "GNU nano 5.4" and "note.txt *". The main editing area contains the text "Hello!" followed by "It's a training demo text file" with a cursor at the end. At the bottom, a menu of keyboard shortcuts is displayed, including ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, and ^J Justify.

At the bottom of the screen, you'll see two lines with short descriptions. The ^ symbol represents the CTRL key on your keyboard. For example, pressing [CTRL + W] brings up the Search: line at the bottom of the editor, where you can enter the word or phrase you are looking for. If you now enter demo and press [ENTER], the cursor will move to the first instance of the word.

A screenshot of the same terminal window showing the nano editor. The search function has been activated, and a "Search: demo" prompt is visible at the bottom. The status bar at the top remains the same. The bottom menu now includes additional search-related shortcuts: ^G Help, ^C Cancel, M-C Case Sens, M-R Reg.exp., M-B Backwards, ^R Replace, ^P Older, ^N Newer, and ^T Go To Line.

In this way, you can search within the text. To save the changes made in the note.txt file, use the shortcut CTRL + O. Then, to exit the text editor, use the shortcut CTRL + X.

After exiting the text editor, you can verify that your text has been saved by printing the contents of the file with the `cat` command. The `cat` command is used to print the contents of files to the terminal.

A terminal window titled "Terminal - root@hackerbox: ~" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows a sequence of commands and their output: `#nano note.txt`, `#cat note.txt`, and the output "Hello!" followed by "It's a training demo text file". The prompt `[root@hackerbox]~#` is visible at the end of the output.

```
Terminal - root@hackerbox: ~
File Edit View Terminal Tabs Help
[ root@hackerbox ]-[ ~]
#nano note.txt
[ root@hackerbox ]-[ ~]
#cat note.txt
Hello!
It's a training demo text file
[ root@hackerbox ]-[ ~]
#
```

Filters

Filters are programs that take plain text (stored in a file or produced by another program) as standard input, transform it into a meaningful format, and then return it as standard output. Linux has a multitude of filters.

Cat

The primary purpose of the `cat` command is to display the contents of one or more text files on the terminal. This command allows quick viewing of file contents.

```
root@hackerbox:~$ cat /etc/ssh/sshd_config
```

```
# Port 22
```

```
# AddressFamily any
```

```
# ListenAddress 0.0.0.0
```

```
# ListenAddress ::
```

```
PermitRootLogin no
```

```
PasswordAuthentication yes
```

```
PermitEmptyPasswords no
```

```
ChallengeResponseAuthentication no
```

UsePAM yes

In the example above, the configuration file for the SSH service located at `/etc/ssh/sshd_config` is displayed on the terminal using the `cat` command.

Head

The `head` command is used to display the first few lines of a specified file. By default, the `head` command shows the first 10 lines, but this number can be changed using the `-n` parameter. This command is handy when you want to quickly review the beginning of a file without displaying its entire content.

```
root@hackerbox:~$ head -n 3 /var/log/apache2/access.log
```

```
192.168.1.1 - - [15/Mar/2024:10:00:00 +0000] "GET /index.html HTTP/1.1" 200 612 "-"  
"Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
```

```
192.168.1.2 - - [15/Mar/2024:10:00:02 +0000] "POST /login.php HTTP/1.1" 200 452  
"http://example.com/login" "Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N)"
```

```
192.168.1.3 - - [15/Mar/2024:10:00:03 +0000] "GET /wp-admin HTTP/1.1" 403 497 "-"  
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6)"
```

In the example above, the first 3 lines of the log file located at `/var/log/apache2/access.log` for the Apache2 Web Server service are displayed on the terminal.

Tail

The `tail` command is used to display the last few lines of a specified file. By default, the `tail` command shows the last 10 lines, but this number can be changed using the `-n` parameter. This command is extremely useful for monitoring the most recently added content to continuously growing files, such as log files.

```
root@hackerbox:~$ tail -n 3 /var/log/auth.log
```

```
Mar 15 12:00:00 servername sshd[23456]: Failed password for invalid user admin from  
192.168.1.1 port 54321 ssh2
```

```
Mar 15 12:01:00 servername sshd[23457]: Accepted password for user1 from 192.168.1.2  
port 65432 ssh2
```

```
Mar 15 12:02:00 servername sshd[23458]: Failed password for user2 from 192.168.1.3 port  
76543 ssh2
```

In the example above, the last three lines of the `auth.log` file are displayed. The `auth.log` file logs events related to user authentication on a Linux system, including user logins and logouts, `sudo` command usage, SSH sessions, and other authentication-related events.

Sort

The `sort` command sorts the contents of a given file alphabetically.

```
root@hackerbox:~$ cat names.txt
```

Bob

Charlie

Alice

```
root@hackerbox:~$ sort names.txt
```

Alice

Bob

Charlie

In the example above, the contents of "`names.txt`" are sorted alphabetically.

Uniq

The `uniq` command filters out consecutive duplicate lines from a file and shows the unique lines. It is often used in conjunction with the `sort` command because, when used alone, it only detects consecutive duplicate lines. To address duplicates throughout the file, it's recommended to first sort the data.

```
root@hackerbox:~$ cat names.txt
```

Alice

Charlie

Alice

Bob

```
root@hackerbox:~$ uniq names.txt
```

Alice

Charlie

Alice

Bob

In the example above, although "Alice" appears twice, the `uniq` command does not remove non-consecutive duplicates. Therefore, to eliminate non-consecutive duplicates, first sort the file and then apply the `uniq` command.

```
root@hackerbox:~$ sort names.txt | uniq
```

Alice

Bob

Charlie

Grep

The `grep` command searches files for specific text strings, filters lines, and displays matching results. `grep` is a powerful tool commonly used to search log files, configuration files, or any text file.

```
root@hackerbox:~$ grep '192.168.1.1' /var/log/apache2/access.log
```

This command will display all records in the Apache 2 web server access logs located at `/var/log/apache2/access.log` that contain the IP address 192.168.1.1.

```
root@hackerbox:~$ grep '192.168.1.1' /var/log/apache2/access.log
```

```
192.168.1.1 - - [15/Mar/2024:10:00:00 +0000] "GET /index.html HTTP/1.1" 200 612 "-"  
"Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
```

```
192.168.1.1 - - [15/Mar/2024:10:00:02 +0000] "POST /login.php HTTP/1.1" 200 452  
"http://example.com/login" "Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N)"
```

```
192.168.1.1 - - [15/Mar/2024:10:00:03 +0000] "GET /wp-admin HTTP/1.1" 403 497 "-"  
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6)"
```

Wc

The `wc` (word count) command quickly determines how large a file is or how much data it contains.

```
root@hackerbox:~$ wc /etc/passwd
```

```
46  67 2544 /etc/passwd
```

In the example above, the `wc` command returns the number of lines, words, and characters respectively in the `/etc/passwd` file, which contains the list of registered users on a Linux system.

Column Value	Description
46	Number of lines
67	Number of words
2544	Number of characters
/etc/passwd	File path

The **wc** command has various parameters:

- -l: Displays only the number of lines.
- -w: Displays only the number of words.
- -c: Displays only the number of bytes.
- -m: Displays only the number of characters (useful for multi-byte character sets).

For example, to see the total number of log entries recorded by the Apache 2 web server to date, you can use the -l parameter with the following command:

```
root@hackerbox:~$ wc -l /var/log/apache2/access.log
```

```
54230 /var/log/apache2/access.log
```

The output indicates that there are a total of **54,230** log entries.

Sed

The sed (stream editor) command is a tool capable of performing various text edits such as processing, modifying, adding, deleting, or replacing texts between files. The sed command is commonly used to filter and transform texts.

```
root@hackerbox:~$ cat names.txt
```

```
Alice
```

```
Charlie
```

```
Bob
```

```
root@hackerbox:~$ sed 's/Alice/George/' names.txt
```

```
George
```

```
Charlie
```

Bob

In the example above, the name **Alice** in names.txt is replaced with **George** using the sed command. However, note that the sed command only prints the change to the screen and does not save it to the file.

Awk

The awk command is designed for text and data processing tasks, and it is especially effective when working with column-based data. It reads files line by line, splits each line into fields (columns), and processes them based on specified conditions. awk offers numerous functions and control structures for complex text processing.

```
root@hackerbox:~$ cat names.txt
```

John Doe

Emily Clark

Alex Turner

```
root@hackerbox:~$ awk '{print $1}' names.txt
```

John

Emily

Alex

In this example, the file names.txt contains three name-surname pairs: John Doe, Emily Clark, and Alex Turner. The command `awk '{print $1}' names.txt` processes the content of this file using the awk program. awk reads the text files line by line, splitting each line into fields separated by spaces or tabs. In this case, the expression `{print $1}` instructs awk to print only the first field (the first name) of each line. Using these useful Linux filters, you can process, search, and transform text files quickly and efficiently according to your specific needs.

Package Management

In the daily use and system administration of Linux operating systems, managing software packages is critically important. Linux distributions provide various package managers to facilitate tasks such as software installation, updating, and removal. These package managers allow you to manage software in package formats, along with their dependencies.

Each Linux distribution uses specific package management systems and package formats. In this section, we will explore package management tools commonly used in Debian-based Linux distributions and their basic usages.

Key Features of Package Managers

Linux package managers offer various features for managing software packages:

- **Package Downloading:** Automatically download software packages and dependencies from the internet.
- **Dependency Resolution:** Identify and automatically install the dependencies required by a package.
- **Package Installation and Removal:** Install software packages to the system and remove them when necessary.
- **Updates and Upgrades:** Check for new versions of installed packages and update them.
- **Configuration Files and Directories:** Provide standards for the configuration files and directories of software packages.

Advanced Package Manager (APT)

APT (Advanced Package Tool) is a powerful tool used to manage software packages in Debian-based Linux distributions such as Ubuntu. This tool allows users to easily install new software, update existing software, remove unnecessary ones, and automatically resolve dependencies between software packages. With simple command-line commands, users can keep their systems up-to-date and secure, while also quickly accessing the software they need. The convenience and efficiency offered by APT make it a popular choice among Linux users.

Updating Package Lists

APT keeps package lists in a database on your local system. It doesn't connect to servers each time you search, thus providing faster results, but the results might not be up-to-date. To get the most current results, you need to update your lists.

To update your package list with apt, use the **update** command.

```
user@hackerbox:~$ sudo apt update
```

```
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
```

```
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic InRelease
```

```
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
```

```
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
```

```
Reading package lists... Done
```

Building dependency tree

Reading state information... Done

All packages are up to date.

Searching in Package Lists

Now that we have updated the package lists in the database, we can search within those lists. For example, to search for a package with the name **htop**, you can perform the following search.

```
user@hackerbox:~$ sudo apt search htop
```

```
aha - ANSI color to HTML converter
```

```
htop - interactive processes viewer
```

```
libauthen-oath-perl - Perl module for OATH One Time Passwords
```

As you can see in the results, in addition to the **htop** package, there are other unrelated packages. This is because the `apt search` command also searches within the package descriptions. For example, if we look at the description of the **aha** package, we can see where

htop is mentioned. The `apt search` command supports regular expressions. For example, to search for packages starting with **htop**, you could do:

```
user@hackerbox:~$ sudo apt search ^htop
```

```
htop - interactive processes viewer
```

Or, you could search for packages whose names contain the term **htop**.

```
user@hackerbox:~$ sudo apt search --names-only htop
```

```
htop - interactive processes viewer
```

Installing and Updating Packages

To install a package found, use the `apt` command.

```
user@hackerbox:~$ sudo apt install htop
```

```
Reading package lists... Done
```

```
Building dependency tree
```

```
Reading state information... Done
```

The following NEW packages will be installed:

htop

0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.

Need to get 80.0 kB of archives.

After this operation, 201 kB of additional disk space will be used.

Get:1 <http://archive.ubuntu.com/ubuntu> bionic/universe amd64 htop amd64 2.1.0-3 [80.0 kB]

Fetched 80.0 kB in 1s (110 kB/s)

Selecting previously unselected package htop.

(Reading database ... 32507 files and directories currently installed.)

Preparing to unpack .../htop_2.1.0-3_amd64.deb ...

Unpacking htop (2.1.0-3) ...

Setting up htop (2.1.0-3) ...

Processing triggers for man-db (2.8.3-2ubuntu0.1) ...

This command also updates an existing installed package to the latest version if it's already installed.

To update all packages on your system, you can use the **upgrade** command.

```
user@hackerbox:~$ sudo apt upgrade
```

This method will not install any new packages or remove any old ones. If this isn't a concern, you can use the **dist-upgrade** command.

```
user@hackerbox:~$ sudo apt dist-upgrade
```

Removing Packages

To remove a package, use the **remove** command.

```
user@hackerbox:~$ sudo apt remove htop
```

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages will be REMOVED:

htop

0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.

After this operation, 201 kB disk space will be freed.

(Reading database ... 32558 files and directories currently installed.)

Removing htop (2.1.0-3) ...

Processing triggers for man-db (2.8.3-2ubuntu0.1) ...

The output above shows that the htop package has been successfully removed from the system, freeing up 201 kB of disk space.

The **remove** command does not delete the configuration files and deb files for **htop**. To remove those as well, you would use the **purge** command.

```
user@hackerbox:~$ sudo apt purge htop
```

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages will be REMOVED:

htop*

0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.

After this operation, 201 kB disk space will be freed.

(Reading database ... 32558 files and directories currently installed.)

Removing htop (2.1.0-3) ...

Processing triggers for man-db (2.8.3-2ubuntu0.1) ...

Purging configuration files for htop (2.1.0-3) ...

User Management

User management in Linux operating systems is of vital importance for system security and efficient resource sharing. This section focuses on how to create, manage, and delete users in Linux.

What is a User in Linux?

In Linux systems, users are defined as individuals or entities performing various tasks by logging into the system. User management is crucial for controlled access, resource allocation, and overall system administration.

In Linux, a user is associated with a user account that has several attributes defining their identity and privileges within the system. These attributes include the username, UID (User ID), GID (Group ID), home directory, default shell, and password.

Each user account possesses unique attributes listed above.

Types of Users

Linux supports two types of users: system users and regular users.

System users are created by the system during installation and are used to run system services and applications.

Regular users are created by an administrator and can access the system and resources based on their permissions.

Creating a User

To create a user, use the `useradd` command. For example, to create a user named "John," use the following command:

```
root@hackerbox:~$ useradd -u 1002 -d /home/john -s /bin/bash john
```

This command creates a user account for John with a user ID (UID) of 1002, a home directory set as /home/john, and a default shell of /bin/bash. You can verify the newly created user account by running the `id john` command. This command shows the ID and group memberships for the john user.

```
root@hackerbox:~$ id john
```

```
uid=1002(john) gid=1002(john) groups=1002(john)
```

User Attributes

In Linux systems, user accounts have various attributes that define their properties and access privileges.

- **Username:** A unique identifier for the user within the Linux system. For instance, John's username is john.
- **UID (User ID) and GID (Group ID):** Each user account is associated with a UID and a GID. The UID is a numeric value assigned to the user, while the GID represents their

primary group. For example, John's UID is 1002, and his primary group's GID could also be 1002.

- **Home Directory:** A designated directory where the user's personal files and settings are stored. John's home directory is `/home/john`.
- **Default Shell:** The default shell specifies the command interpreter used when the user logs in. This defines the user's interactive environment. John's default shell is set to `/bin/bash`, a popular shell in Linux.
- **Password:** User accounts require passwords for access and authentication.
- **Group:** Group membership determines which system resources the user can access and which other users can access the user's files.

In Linux systems, registered users are stored in the `/etc/passwd` file. You can display the contents of this file to see the list of users on the system.

```
root@hackerbox:~$ cat /etc/passwd
```

```
root:x:0:0:System Administrator:/root:/bin/bash
```

```
john:x:1002:1002:John Doe:/home/johndoe:/bin/bash
```

The user list within the `/etc/passwd` file follows this format:

Column Value	Description
john	Username
x	Contains the hashed password of the user. For security reasons, the password is stored in the /etc/shadow file, so this field is replaced with the character x .
1002	UID (User ID) of the user account, a unique numeric identifier assigned to the user by the system.
1002	GID (Group ID) of the user account, representing their primary group membership.
"	GECOS field, which stands for "General Electric Comprehensive Operating System." This field is used to store additional information about the user, such as the full name or contact information. In this case, the field is empty because no additional information was provided during account creation.
/home/john	Home directory of the user account where the user's files and personal data are stored.
/bin/bash	Default shell of the user account, used to interpret commands entered by the user in the terminal. In this case, the default shell is Bash, the most commonly used shell in Linux.

Changing User Passwords

User passwords can be easily changed using the `passwd` command. For example, to set a new password for the `john` user, use the following command:

```
root@hackerbox:~$ sudo passwd john
```

This command prompts you to enter a new password interactively. Note that nothing will appear on the screen as you type for security reasons. Simply type the new password and press `ENTER`.

Deleting a User

To remove a user named `John` and their associated files, use the `userdel` command.

```
root@hackerbox:~$ sudo userdel john
```

This command deletes the `john` user's account, including their home directory and all files owned by the user.

Group Management

In Linux operating systems, group management is as important as user management. Groups are used to collectively assign the same access rights and permissions to multiple users. This enables system administrators to easily control resources and access. This section focuses on how to create, manage, and delete groups in Linux.

What is a Group in Linux?

In Linux systems, a group is a collection of users who have specific permissions and access rights. Groups are used to simplify access control in file systems and make user management more efficient.

Creating a Group

To create a new group, use the `groupadd` command. For example, to create a group named `development`, use the following command:

```
root@hackerbox:~$ sudo groupadd development
```

You can view the groups we have created in the `/etc/group` file.

```
root@hackerbox:~$ cat /etc/group
```

```
root:x:0:
```

```
daemon:x:1:
```

bin:x:2:

sys:x:3:

adm:x:4:

tty:x:5:

disk:x:6:

...

development:x:1004:

The `/etc/group` file may contain many groups, making it difficult to find the group you are looking for. To simplify your task, you can use the command below with `grep` to display only a specific group:

```
root@hackerbox:~$ cat /etc/group | grep development
```

development:x:1004:

Adding Users to a Group

To add users to the created group, use the `usermod` command with the `-aG` option. The `-aG` option adds the user to the specified group while preserving existing group memberships. For example, to add the user "john" to the "development" group, use the command below:

```
root@hackerbox:~$ sudo usermod -aG development john
```

Deleting a Group

To remove a group that is no longer needed, use the `groupdel` command. For example, to delete the "development" group, use the following command:

```
root@hackerbox:~$ sudo groupdel development
```

This command deletes the "development" group from the system.

Permissions

Just like in other operating systems, multiple user accounts can be created on Linux, and these users can share the same system.

However, when different users share the same system, privacy issues can easily arise. For instance, one user may not want others to view, edit, or delete their files.

We can address this issue with permissions that can be defined at the file and directory level.

To view the permissions for a file or directory, we can use the `-l` parameter of the `ls` command, as discussed in previous sections.

```
root@hackerbox:~$ ls -l
```

```
drwxr--r-- 2 john development 4096 Jul 29 12:34 notes.txt
```

The columns in the output obtained with the `-l` parameter of the `ls` command are as follows:

Column Content	Description
d	File type. If it's a directory, it's shown as d, if it's a file, it's shown as -. In this example, it's d, so it's a directory .
rwxr--r--	File permissions
2	Number of hard links to the file/directory
john	Owner of the file/directory
development	Group owner of the file/directory
4096	Size of the file or the block count used to store directory information
Jul 29 12:34	Creation or last modification date of the file/directory
notes.txt	Name of the file/directory

Understanding Permissions

The file permissions (`rwxr--r--`) given in the example above can be thought of as three different sets of permissions consisting of 9 characters in total. Each set of three characters represents the user, group, and others permission sets.

```
---      ---      ---  
  
rwx      rwx      rwx
```

user group others

r, w, x, and - Characters

The r character represents read permission, i.e., the permission to read the contents of the file. The w character represents write permission, i.e., the permission to write or modify the contents of the file. The x character represents execute permission, i.e., the permission to execute the file. The x permission is given only to executable programs. If any of the rwx characters are replaced with -, it means that permission is not granted.

User, Group, and Others

- user - User permissions concern only the owner of the file or directory.
- group - Group permissions concern only the users who belong to the group assigned to the file or directory.
- others - Other permissions concern all other users and groups on the system.

Reading Permissions

First, let's divide the given permissions (rwxr--r--) into three distinct groups.

rwx r-- r--

user group others

It is seen that all permissions (read, write, and execute) are granted for the owner user. In other words, the owner of the file (the user named john) can read, modify, and execute this file. However, since this file is a text file, as indicated by its name, it will not execute even though it has execute permission.

For group permissions, only read permission is granted to the group assigned to the file. Write and execute permissions are not granted, as indicated by the -character. Members of the development group, to which the file is assigned, have only read permission for this file.

As for the permissions of other users and groups, it is also seen that only read permission is granted. Again, write and execute permissions are not granted, as indicated by the - character. This means that all other users and groups on the system have read permission for this file.

Changing File and Directory Permissions

To change file and directory permissions, use the chmod command. The first argument given to the chmod command indicates which permission set you want to change. You can specify the permission set with the u, g, or o options.

- u (user) - Owner user permissions
- g (group) - Group permissions
- o (others) - Other permissions

To change permissions for all sets, you can use u,g,o. After specifying the first argument, you need to indicate whether you want to add or remove a permission. You can use the + or - options.

- + - Adds permission
- - - Removes permission

Lastly, you need to specify which permission you want to change (r, w, or x).

- r (read) - Read permission
- w (write) - Write permission
- x (execute) - Execute permission
- You can also use the combination rwx.

Let's do an example to understand it better

For instance, if we want to grant write permission to others for the file notes.txt, we start the command by indicating the permission set (o for others):

```
root@hackerbox:~$ chmod o
```

Then, we indicate whether we want to add or remove the permission. Since we want to add the permission, we use the + character.

```
root@hackerbox:~$ chmod o+
```

Lastly, we specify the permission (w for write).

```
root@hackerbox:~$ chmod o+w
```

Finally, we specify the file we want to modify, notes.txt, and execute the command. We then verify the changes with ls -l.

```
root@hackerbox:~$ chmod o+w notes.txt
```

```
root@hackerbox:~$ ls -l
```

```
drwxr--rw- 2 john development 4096 Jul 29 12:34 notes.txt
```

As we can see, the permission set for others has been changed to rw-. Now, other users can read and write to the file.

Another example

You can also update multiple permissions and groups in a single command. For example, to grant all permissions to all sets for the file notes.txt, run the following command:

```
root@hackerbox:~$ chmod ugo+rwx notes.txt
```

```
root@hackerbox:~$ ls -l
```

```
drwxrwxrwx 2 john development 4096 Jul 29 12:34 notes.txt
```

Process Management

In Linux, the term "process" is defined as a program loaded into memory and executing in the processor (CPU).

Processes can range from short commands run from the command line to network services running for the duration of the operating system's uptime.

Types of Processes

Foreground Processes

By default, all processes run in the foreground. They take input from the keyboard and display output on the screen. Running the `pwd` command is a good example of a foreground process.

```
root@hackerbox:~$ pwd
```

```
/root
```

In this example, the process executed by the `pwd` command ran in the foreground, displayed the output, and exited after completing its task. Foreground processes lock the terminal until they finish their tasks and do not allow other operations to be performed in the meantime.

Background Processes

Background processes do not lock the terminal when run; they start running in the background. When background processes are running, multiple processes can run in parallel.

To run a command in the background, append the `&` character to it. For example, let's run the `pwd` command in the background:

```
root@hackerbox:~$ pwd &
```

```
[1] + Done          pwd
```

Unlike the first example, the `pwd` command did not print the path of the current directory as it was run as a background process. Instead, it provided information about the background process it started. `[1]` indicates that the process job number is 1. `Done` indicates that the process successfully completed its task.

Let's do a different example

The `pwd` command, by nature, completes its task immediately and exits. However, some commands run continuously until you send a stop signal. The `ping` command is a good example of this. Let's send a ping to 127.0.0.1 with the `ping` command and run it in the background. This will allow us to send ping requests to 127.0.0.1 for as long as our terminal or the running process remains open, while still allowing us to use our terminal for other tasks.

```
root@hackerbox:~$ ping 127.0.0.1 &
```

```
[1] 54017
```

```
root@hackerbox:~$
```

```
64 bytes from 127.0.0.1: icmp_seq=100 ttl=64 time=0.081 ms
```

```
64 bytes from 127.0.0.1: icmp_seq=101 ttl=64 time=0.164 ms
```

```
64 bytes from 127.0.0.1: icmp_seq=102 ttl=64 time=0.075 ms
```

[1] indicates that the background job number for the process is 1. 54017 is the Process ID (PID) for the background process. The PID is a unique number identifying the process. Some commands, like `ping`, may provide output to the terminal even when running in the background. As seen in the example, although running in the background, the `ping` command prints output to the terminal every second. However, this output does not prevent us from performing other tasks and does not lock the terminal.

Managing Processes in the Current Terminal Session

We can easily list the background processes started in our current terminal session using the `jobs` command.

```
root@hackerbox:~$ jobs
```

```
[1]  running  ping 127.0.0.1
```

According to the output, we have a `ping 127.0.0.1` command running in the background in our current terminal session, with a job number of 1. To bring this background command back to the foreground, we can use the `fg` (foreground) command. We type the job number of the process with a `%` sign next to the `fg` command.

```
root@hackerbox:~$ fg %1
```

```
[1] - running  ping 127.0.0.1
```

```
64 bytes from 127.0.0.1: icmp_seq=645 ttl=64 time=1.448 ms
```

```
64 bytes from 127.0.0.1: icmp_seq=646 ttl=64 time=0.161 ms
```



```
64 bytes from 127.0.0.1: icmp_seq=647 ttl=64 time=0.219 ms
```

```
64 bytes from 127.0.0.1: icmp_seq=648 ttl=64 time=0.130 ms
```

As we can see, the process is running in the foreground again. How do we stop it? We can easily stop a program running in the terminal by pressing the CTRL+C shortcut on the keyboard.

```
64 bytes from 127.0.0.1: icmp_seq=645 ttl=64 time=1.448 ms
```

```
64 bytes from 127.0.0.1: icmp_seq=646 ttl=64 time=0.161 ms
```

```
64 bytes from 127.0.0.1: icmp_seq=647 ttl=64 time=0.219 ms
```

```
64 bytes from 127.0.0.1: icmp_seq=648 ttl=64 time=0.130 ms
```

```
^C
```

```
--- 127.0.0.1 ping statistics ---
```

```
706 packets transmitted, 706 packets received, 0.0% packet loss round-trip  
min/avg/max/stddev = 0.055/0.180/1.668/0.105 ms
```

```
root@hackerbox:~$
```

Managing System-Wide Processes

In addition to processes in our current terminal session, there are processes running system-wide. These can be processes used by the operating system, the terminal, service providers, or programs that continuously run in the background. We can view system-wide processes using the `ps` command.

```
root@hackerbox:~$ ps
```

PID	TTY	TIME	CMD
19	pts/1	00:00:00	sh
24	pts/1	00:00:00	ps

As seen in the example, there are two processes running system-wide. One is the terminal running the `sh` process, and the other is the `ps` command we just executed. For a more detailed output, we can use the `-f` parameter of the `ps` command.

```
root@hackerbox:~$ ps -f
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
1001	19	1	O	07:20	pts/1	00:00:00f	sh

```
1001    25    19 0 08:04   pts/1 00:00:00    ps -f
```

The detailed explanations of the columns in the command output are as follows:

Column Name	Description
UID	User ID - The ID number of the user running the process
PID	Process ID - The unique identifier for the process
PPID	Parent Process ID - The unique identifier for the parent process that started the process
C	CPU - The CPU usage of the process
STIME	Start Time - The start time of the process
TTY	Terminal Type - The type of terminal where the process is running
TIME	The total time the process has been running
CMD	Command - The command that started the process

Stopping a Running Process

To stop running processes, if the process runs in the foreground, we can easily press the CTRL+C shortcut on the keyboard. If it's a background process, we need to know the Process ID (PID) number to stop it.

```
root@hackerbox:~$ ps
```

```
UID    PID  PPID  C  STIME   TTY      TIME CMD
```

```
1001   19    1 0 07:20   pts/1 00:00:00f  sh
```

```
1001    25    19 0 08:04    pts/1 00:00:00    ps -f
```

In the output above, we have the PID for the process `sh` (19). To stop this process, use the `kill` command.

```
root@hackerbox:~$ kill 19
```

If the process does not terminate with the command above, we can forcefully stop it with the interrupt signal `-9`.

```
root@hackerbox:~$ kill -9 19
```

Network Management

Network management in Linux is accomplished through commands and configuration files. There are many commands available for network configuration and troubleshooting. This section will teach you how to configure the network on a Linux operating system.

Note: This section applies to Debian and derivative distributions.

Network Interface Configuration

Many GNU/Linux system administrators still prefer the traditional `ifconfig` command to configure network interface cards (NIC). It is a traditional command used to configure and manage network interfaces in Linux and Unix-based operating systems. Preferred by system administrators and network professionals for many years, this tool has been used to perform various network configuration tasks such as assigning IP addresses, setting netmasks, and activating or deactivating network interfaces.

Listing Available Devices

When the `ifconfig` command is called without parameters, it lists the available network devices (NIC, Network Interface Controller).

```
root@hackerbox:~$ ifconfig
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

```
    inet 172.20.1.109  netmask 255.255.255.0  broadcast 172.20.1.255
```

```
    inet6 fe80::5054:ff:fe10:72c3  prefixlen 64  scopeid 0x20<link>
```

```
    ether 52:54:00:10:72:c3  txqueuelen 1000  (Ethernet)
```

```
    RX packets 4542  bytes 352144 (343.8 KiB)
```

RX errors 2 dropped 0 overruns 0 frame 2

TX packets 1475 bytes 6213607 (5.9 MiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

device interrupt 11 memory 0xfc840000-fc860000

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 16 bytes 1888 (1.8 KiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 16 bytes 1888 (1.8 KiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

In the output above, there are 2 network interfaces.

eth0: This is the Ethernet card interface. The UP flag indicates it is active. The IP address is 172.20.1.109. The MAC address is 52:54:00:10:72:c3.

lo: This is the Loopback interface. It is a virtual interface created to allow local networking, pointing to the 127.0.0.1 IP address.

To view a specific interface, provide the interface name as a parameter:

```
root@hackerbox:~$ ifconfig eth0
```

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 172.20.1.109 netmask 255.255.255.0 broadcast 172.20.1.255

inet6 fe80::5054:ff:fe10:72c3 prefixlen 64 scopeid 0x20<link>

ether 52:54:00:10:72:c3 txqueuelen 1000 (Ethernet)

RX packets 531168 bytes 41026391 (39.1 MiB)

RX errors 2 dropped 0 overruns 0 frame 2

TX packets 4130 bytes 499172576 (476.0 MiB)

```
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
device interrupt 11 memory 0xfc840000-fc860000
```

To view interfaces that are DOWN (i.e., inactive), use the `-a` parameter.

```
root@hackerbox:~$ ifconfig -a
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
```

```
inet 172.20.1.109 netmask 255.255.255.0 broadcast 172.20.1.255
```

```
inet6 fe80::5054:ff:fe10:72c3 prefixlen 64 scopeid 0x20<link>
```

```
ether 52:54:00:10:72:c3 txqueuelen 1000 (Ethernet)
```

```
RX packets 4542 bytes 352144 (343.8 KiB)
```

```
RX errors 2 dropped 0 overruns 0 frame 2
```

```
TX packets 1475 bytes 6213607 (5.9 MiB)
```

```
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
device interrupt 11 memory 0xfc840000-fc860000
```

```
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 16 bytes 1888 (1.8 KiB)
```

```
RX errors 0 dropped 0 overruns 0 frame 0
```

```
TX packets 16 bytes 1888 (1.8 KiB)
```

```
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Since we don't have any inactive network interfaces, the output of the `ifconfig -a` command remains the same as the previous one.

Activating and Deactivating Interfaces

To bring an interface (e.g., `eth0`) up, use the `ifconfig` command as follows:

```
root@hackerbox:~$ ifconfig eth0 up
```

To take an interface down, use the following command:

```
root@hackerbox:~$ ifconfig eth0 down
```

Note: Performing these actions on the interface connected to your internet may affect your internet connection.

Assigning an IP Address

To assign an IP address to a network interface or update an existing IP address using the `ifconfig` command, directly write the interface name and the desired IP address:

```
root@hackerbox:~$ ifconfig eth0 172.20.1.110
```

```
root@hackerbox:~$ ifconfig -a
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
```

```
    inet 172.20.1.110 netmask 255.255.255.0 broadcast 172.20.1.255
```

```
    inet6 fe80::5054:ff:fe10:72c3 prefixlen 64 scopeid 0x20<link>
```

```
    ether 52:54:00:10:72:c3 txqueuelen 1000 (Ethernet)
```

```
    RX packets 4542 bytes 352144 (343.8 KiB)
```

```
    RX errors 2 dropped 0 overruns 0 frame 2
```

```
    TX packets 1475 bytes 6213607 (5.9 MiB)
```

```
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
    device interrupt 11 memory 0xfc840000-fc860000
```

```
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 16 bytes 1888 (1.8 KiB)
```

```
    RX errors 0 dropped 0 overruns 0 frame 0
```

```
    TX packets 16 bytes 1888 (1.8 KiB)
```

```
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

In the example above, the IP address of the eth0 interface was changed to 172.20.1.110.

Assigning a Netmask

To set the netmask of a network interface, use the following command:

```
root@hackerbox:~$ ifconfig eth0 netmask 255.255.255.0
```

Promiscuous Mode

If your Ethernet card supports it, you can enable promiscuous mode to process packets intended for other devices on the same network.

To enable promiscuous mode:

```
root@hackerbox:~$ ifconfig eth0 promisc
```

To disable promiscuous mode:

```
root@hackerbox:~$ ifconfig eth0 -promisc
```

Changing the MAC Address

You can change the MAC address of your device. Be aware that this might cause confusion in the network's ARP tables, so use it carefully.

```
root@hackerbox:~$ ifconfig eth0 hw ether AA:BB:CC:DD:EE:FF
```

DNS Settings

In Linux, DNS settings are located in the `/etc/resolv.conf` file. You can update the DNS settings within this file using a text editor like nano.

```
root@hackerbox:~$ nano /etc/resolv.conf
```

The file content will look something like this:

```
nameserver 172.20.1.1
```

You can add the DNS servers you want to use, line by line, in this format. For example, to use Cloudflare's DNS servers system-wide, update the file as shown:

```
nameserver 1.1.1.1
```

```
nameserver 1.0.0.1
```

SSH (Secure Shell)

SSH is a protocol used to securely connect to another computer over a network and execute commands. SSH is widely used, especially for accessing and managing remote computers. The `ssh` command is used to create an SSH connection.

Installing and Starting the SSH Service

First, you may need to install the SSH service. On a Debian-based system, you can install the `openssh-server` package using the following command:

```
sudo apt-get update
```

```
sudo apt-get install openssh-server
```

Once the installation is complete, you can start the service:

```
sudo systemctl start ssh
```

To ensure the SSH service starts automatically when the system boots:

```
sudo systemctl enable ssh
```

Connecting to a Remote Server with SSH

You can use the `ssh` command to connect to a remote server:

```
ssh user@ip_address
```

For example, if your username is `root` and the server address is `192.168.1.100`:

```
ssh root@192.168.1.100
```

After running this command, you will be prompted to enter the password of the remote server.

Creating an SSH Key Pair

In addition to password-based login, you can connect without a password (and more securely) by using an SSH key pair. You can create an SSH key by using the `ssh-keygen`

command:

```
ssh-keygen
```

After running this command, you will need to copy the generated public key to the remote server:

```
ssh-copy-id user@ip_address
```

For example:

```
ssh-copy-id root@192.168.1.100
```

Once this process is complete, you can connect using SSH without entering a password.

SSH Configuration File

The SSH configuration settings are usually found in the `/etc/ssh/sshd_config` file. Various SSH settings can be configured in this file, such as changing the SSH port or disabling root logins:

```
sudo nano /etc/ssh/sshd_config
```

In the file content, you can find and edit the Port setting to change the port number:

Port 2222

After making changes, you will need to restart the SSH service:

```
sudo systemctl restart ssh
```

In this section, we learned the basics of using SSH. Now, you can establish secure connections over the network and manage remote servers using SSH.

These changes have comprehensively updated your network management section.
