Basics of Linux Operating System

Introduction to the Linux Operating System

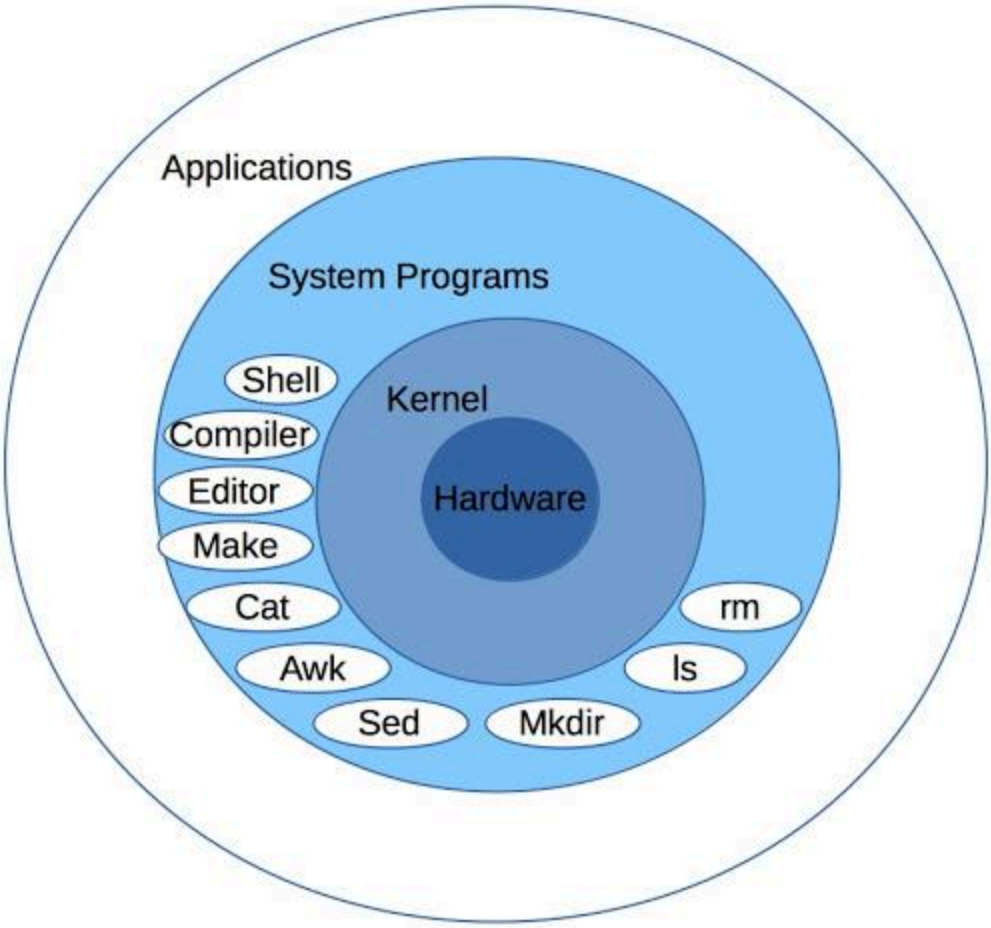
Linux is a powerful and versatile operating system that was developed by Linus Torvalds in 1991. It offers a wide range of features and capabilities that make it a popular choice for various environments, including servers, desktops, smartphones, and IoT devices. Let us explore some key aspects of Linux and its historical connection with Unix.

Key Features of Linux

Operating System (Kernel): The core of the Linux operating system is called the kernel, which manages various hardware and software resources.
Command Line Interface: Linux provides a command line interface (CLI) that allows users to interact with the system using text-based commands, offering powerful control and flexibility.
Open Source: Linux is an open-source operating system, meaning its source code is freely available for anyone to view, modify, and distribute.
Hierarchical File System: Linux uses a hierarchical file system, where directories (folders) and files are organized in a tree-like structure.
Supports Multiple Users: Linux supports multiple users to share a single computer system simultaneously, making it suitable for multi-user environments.
Stable: Linux is known for its stability and reliability, making it a preferred choice for critical systems and applications.
Diverse Environments: Linux is versatile and finds application in various environments, including servers, desktops, smartphones, and IoT devices.
Wide Range of Programming Languages: Linux supports a vast array of programming languages, making it an excellent platform for software development.

Linux Kernel

The Linux kernel is the core component of the Linux operating system. It serves as the bridge between hardware and software, managing system resources and enabling various applications to interact with the computer's hardware. The Linux kernel is open source, allowing developers worldwide to contribute to its continuous improvement.



Source: http://nathanielgmartin.com/unix/Wk01/r1-intro.html

Key features of the Linux kernel include:

Resource Management: The kernel allocates and manages resources like CPU, memory, and peripherals, ensuring efficient utilization and fair distribution among processes.
Device Drivers: It provides device drivers to facilitate communication between hardware components and software applications, enabling seamless access to peripherals.
Process Management: The kernel oversees the creation, execution, and termination of processes, allowing multitasking and proper sharing of CPU time.
Memory Management: It organizes and optimizes the system's memory, handling virtual memory, page allocation, and memory protection.
Inter-Process Communication (IPC): The kernel facilitates communication between different processes, allowing them to exchange data and collaborate.
Security: The kernel enforces security measures, controlling user access rights, protecting system resources, and ensuring data integrity.
File System Support: It supports various file systems, enabling data storage, retrieval, and organization.

Networking: The kernel includes networking protocols and drivers, allowing network connectivity and communication.

Background Processing in Linux

In Linux, background processing refers to the execution of tasks or commands that run independently of the active terminal or user session. These processes are detached from the terminal and continue running even if the user logs out or closes the terminal window.
There are several ways to run commands in the background in Linux:
Using the & symbol: To run a command in the background, simply append an ampersand (&) at the end of the command. For example:

```
$ command &
```

Using nohup: The nohup command allows running a command immune to hangups (i.e., it keeps running even if the user logs out). It is often used for long-running processes that need to continue running independently. Example:

```
$ nohup command &
```

Background processing is commonly used for tasks like system updates, file transfers, backups, data processing, and software compilations.

Unix and Linux Historical Connection

Unix and Linux share a strong historical connection, with Linux being influenced by Unix's design principles. Linux's open-source nature, extensive community support, and compatibility with Unix have made it a dominant force in modern computing, serving a wide range of users and devices across the globe.
Unix was developed in the late 1960s at Bell Labs by Ken Thompson, Dennis Ritchie, and others. It became the foundation for many commercial Unix variants, such as Solaris, AIX, and HP-UX.
Both Unix and Linux share a similar design philosophy of simplicity, modularity, and the use of small, specialized tools to perform tasks. Both Unix and Linux offer powerful CLIs with similar commands and utilities, allowing users to interact with the system efficiently.
Unix and Linux both follow a hierarchical file system structure, with directories organized in a tree-like format, starting from the root directory.

Popular Linux Distributions

Linux serves as the foundation for numerous distributions, each tailored to specific needs and preferences. Some well-known Linux distributions include:

- Ubuntu
- Rocky Linux
- Red Hat Enterprise Linux
- Debian
- Kali
- SUSE Linux

- Arch Linux
- Raspbian
- MX Linux
- Fedora

Package Management

Linux distributions use package managers to install, update, and remove software packages. Two common package managers are:
APT (Advanced Package Tool): primarily used in Debian-based distributions like Ubuntu and Linux Mint. It uses .deb packages.
yum (Yellowdog Updater, Modified): used in Red Hat–based distributions such as Red Hat Enterprise Linux (RHEL), CentOS, and Fedora.

Linux Command Line Interface (CLI)

The Linux terminal provides a command line interface (CLI) that allows users to perform various tasks, manage files, and processes efficiently. Different shells, such as C Shell, Bourne Shell, and Korn Shell, interpret and execute commands.

Advantages of Linux

Vast Array of Development Tools: Linux provides an extensive collection of development tools and programming languages for software development.
Automation with Shell Scripting: Developers can automate repetitive tasks using shell scripting in the Linux CLI.
Cost-Effective: Linux is freely available, making it an attractive option for individual developers and startups.

Multi-User Support

One of the key strengths of Linux is its robust multi-user support, which allows multiple users to access and utilize the same system simultaneously. This feature is essential in shared environments, such as servers, educational institutions, and workplaces, where multiple users need to interact with the system concurrently. The following are the main aspects of Linux's multi-user feature:

1. **User accounts:** Each user on a Linux system has a unique user account. A user account is associated with a username and a unique user ID (UID). The system uses the UID internally to differentiate between users.
2. **User privileges:** Linux differentiates users based on their privileges. The root user (superuser) has administrative access and can perform system-wide changes. Regular users have limited access and can only modify files and settings they are permitted to.

3. **User authentication:** When users log in, they need to provide a valid username and password for authentication. The system verifies the credentials against the user account database to grant access.
4. **Concurrent sessions:** Linux allows multiple users to log in and run processes simultaneously. Each user gets their own session, and their processes are isolated from other users' processes for security and privacy.
5. **Process isolation:** Processes run by different users are isolated from each other, preventing unauthorized access to each other's data and resources. This isolation is crucial for system stability and security.
6. **File permissions:** Linux uses file permissions to control access to files and directories. Each file and directory has permission settings that determine which users can read, write, or execute them.
7. **Home directories:** Each user typically has a home directory where they can store their files and personal settings. Home directories are usually named after the users' usernames (e.g., /home/username).
8. **Superuser (root) privileges:** The root user has unrestricted access to the entire system. Root privileges are required to perform critical system-level tasks, such as installing software, modifying system configurations, and managing user accounts.

su and sudo commands*:* The su (switch user) and sudo (superuser do) commands allow users to switch to another user's account or execute specific commands with root privileges, respectively. This provides a controlled and audited way to perform administrative tasks.

9. Linux's multi-user feature is fundamental to its versatility and popularity as a server operating system.

Basic Commands

1. Logging into Linux System

Life Cycle of a User Session in Linux

In a Linux environment, a user session goes through a sequence of steps from login to logout. Here's an overview of the process:

Login Attempt Initiation:

A user initiates a login attempt either through a graphical login manager or a text-based terminal.

For remote connections, users can log in via SSH (Secure Shell).

Username and Password Entry:

Users enter their username and password during the login attempt.

The system uses this information to authenticate the user's identity.

Authentication:

The system compares the entered password with the encrypted password stored in the system's user account database (usually in the /etc/shadow file).

If the authentication is successful, the user gains access to the system.

Session Initialization:

Upon successful authentication, the system initializes the user's session.

This involves setting up the user's environment, loading necessary configurations, and executing startup scripts.

Assigning a Shell:

After session initialization, the system assigns the user a shell (C Shell, Bourne Shell, Bourne Again Shell – BASH, Korn Shell, etc.), which is a command-line interpreter.

The default shell in most Linux distributions is Bash (Bourne Again SHell).

Session Start:

With the session initialized and the shell assigned, the user can start using the system.

In a graphical environment, this means accessing the desktop environment.

In a text-based environment, the user gains access to the shell.

Logging Out:

When the user finishes their tasks, they can log out to end the session.

Logging out involves cleaning up processes or resources associated with the session.

2. Manual Help & Linux Basic Linux Commands

Here are some essential Linux commands and their usage:

Command: passwd

The passwd command in Linux is used to change a user's password. It allows users to set a new password for their account or, if you have administrative privileges, change the password for another user. The passwd command can be run from the terminal.

Syntax:

passwd [options] [username]

Options:

-l: Lock the password of the specified account.
-u: Unlock the password of the specified account.
-d: Delete the password of the specified account.
-e: Expire the password of the specified account.

Example 1: Changing Your Own Password:

Suppose you are logged in as a regular user "john" and want to change your own password. You can run the passwd command without specifying a username:

$ passwd
Changing password for john.
(current) UNIX password: [enter your current password]
New password: [enter your new password]
Retype new password: [retype your new password]
passwd: password updated successfully

Example 2: Changing Another User's Password (Requires Admin Privileges):

If you have administrative privileges (e.g., you are the root user or a user with sudo access), you can change the password of another user. In this example, let's change the password for the user "alice":

$ sudo passwd alice
[sudo] password for your_username: [enter your sudo password]
Changing password for user alice.
New password: [enter the new password for alice]
Retype new password: [retype the new password for alice]
passwd: password updated successfully

Command: date

The date command in Linux is used to display or set the current system date and time. When executed without any options, it simply prints the current date and time in the default format, which typically includes the day of the week, month, day, time, timezone, and year.

However, you can customize the output format using various options available with the date command. It is also possible to use the date command to set the system's date and time if you have the appropriate permissions.

Displaying the date and time in a specific format:

%a: Abbreviated weekday name (e.g., Sun).
%A: Full weekday name (e.g., Sunday).
%b: Abbreviated month name (e.g., Jan).
%B: Full month name (e.g., January).
%d: Day of the month (01 to 31).
%H: Hour in 24-hour format (00 to 23).
%I: Hour in 12-hour format (01 to 12).

%M: Minute (00 to 59).
%S: Second (00 to 60).
%Y: Year with century (e.g., 2023).
%y: Year without century (e.g., 23).
%Z: Timezone name (e.g., UTC).

Example: Display date in given format

$ date +"%A, %d %B %Y"
Sunday, 31 July 2023

Example: Setting the system date and time

To set the date and time, you need superuser (root) privileges. The format to set the date is usually "MMDDhhmm[[CC]YY][.ss]" where MM is the month, DD is the day, hh is the hour, mm is the minute, CC is the first two digits of the year (century), YY is the last two digits of the year, and ss is the seconds.

$ sudo date MMDDhhmm[[CC]YY][.ss]
$ sudo date 073110302023.30

Sets the date to July 31, 2023, 10:30 AM and 30 seconds.

Example: Displaying a date relative to the current date

+n days: Display the date n days in the future.
−n days: Display the date n days in the past.

Add 2 days to the current date

$ date "+%A, %d %B %Y" -d "+2 days"
Tuesday, 02 August 2023

Check the new date

$ date
Sun Jul 31 10:30:00 UTC 2023

Command: who

The who command provides information about currently logged-in users on the system. It displays details such as the username, terminal, date, and time of the login. If used without any options, it shows a list of all logged-in users on the system. This can be useful for system administrators to monitor active sessions and check who is currently using the system.

Example:

$ who
john pts/0 2023-07-31 09:45 (:0)
alice pts/1 2023-07-31 10:00 (:0)

Command: whoami

The whoami command is used to print the username of the current user who is executing the command. It is helpful when you need to know the current user's identity, especially in shell scripts or when managing file permissions and access control.

Example:

$ whoami
John

Options

-q or --count Gives number of users currently logged in
-H or --head Displays the header line

who -q or who --count:

```
$ who -q
# users=2
```

who -H or who --heading:

The -H option displays a header line, indicating the meaning of each column in the output.

```
$ who -H
NAME     LINE         TIME             COMMENT
john     tty1         2023-07-31 09:45 (:0)
alice    pts/0        2023-07-31 10:00 (:0)
```

Command: write [username]

The write command in Linux allows users to send messages to other logged-in users on the same system. It is a simple and efficient way to communicate with other users who are currently using the terminal.

The basic syntax of the write command is as follows:

```
write [username] [terminal]
```

Here, username is the username of the user to whom you want to send the message, and terminal is the terminal or device number where the recipient is logged in. If terminal is not specified, the message is sent to the user's terminal where the write command was executed.

Example:

```
$ write john
```

After entering this command, you can type your message, and the recipient (in this case, the user "john") will receive your message on their terminal.

Options for the write command:

write -h or write --help:

Displays the help message, which includes a brief description of the write command and its usage.

write -V or write --version:

Shows the version information of the write command.

write -t timeout or write --timeout=timeout:

Specifies the duration (in seconds) after which the write command times out if the recipient does not respond. If the timeout is reached, the message will not be sent.

Command: man

The man command in Linux is used to display the manual pages (documentation) for other commands or programs installed on your system. It provides comprehensive information about the usage, options, and functionality of various commands.

man [command]: This is the basic syntax for using the man command. Replace [command] with the name of the command you want to know more about. For example, man ls will display the manual page for the ls command.

man -k [keyword] or man -f [keyword]: These options allow you to search for commands by a specific keyword. For instance, man -k network will list all commands related to "network."

man -a [command]: When multiple manual pages exist for the same command (e.g., from different software packages), using this option will display all the available manual pages for the specified command.

man -w [command]: This option prints the path to the manual page file for the given command, allowing you to locate where the documentation is stored on your system.
man --help or man -h: These options display a short summary of the man command itself, including the available options and how to use them.
man --sections [section_number] [command]: Manual pages are categorized into different sections (e.g., 1 for commands, 2 for system calls, 3 for library functions). Using this option, you can specify a section number to view the manual page from a particular section. For example, man --sections 3 printf will show the manual page for the printf library function.
Example:

man ls

After pressing Enter, the man command will display the manual page for the ls command. You can scroll through the page using the arrow keys, and to exit the manual page, press q.
The manual page for ls will provide you with detailed information about how to use the command, including a description of its functionality, available options, and usage examples.

3. Linux Basic Commands: ln, pwd, and ls with options

Command: ln [options] source_file target_file
The ln command in Linux is used to create links between files or directories. Links are essentially pointers to the original file or directory, and they allow multiple filenames to refer to the same data on disk. The ln command has two primary types of links: hard links and symbolic links (also known as soft links). Here are some short notes on the ln command with its options:
Basic Syntax:

ln [OPTION]... [-T] TARGET LINK_NAME

Creating Hard Links:
Hard links are direct references to the same inode (data) on disk as the original file. If you modify the hard link's content, it affects the original file and vice versa. Hard links can't link to directories and cannot span across different filesystems (partitions).
By default, ln creates hard links.
Creating Symbolic Links (Soft Links):
Symbolic links are references to the path of the original file or directory. They act as shortcuts, and if you modify the symbolic link, it doesn't affect the original file. Symbolic links can link to directories and can span across different filesystems.
To create a symbolic link, use the -s option.
Usage Examples: Creating a hard link:

ln file.txt hardlink.txt

Usage Examples: Creating a symbolic link:

ln -s file.txt symlink.txt

Usage Examples: Overwriting existing link without prompting:

ln -sf newfile.txt existinglink

Usage Examples: Creating a symbolic link to a directory:

ln -s /path/to/source_directory link_to_directory

Remember that when you delete a link (hard link or symbolic link), it does not affect the original file or directory.

Command: pwd

The pwd command in Linux stands for "print working directory." It is used to display the current working directory, which is the directory in the file system where the user is currently located. The pwd command is a simple and useful tool for quickly identifying the directory path, which can be handy while navigating through the file system or working with relative paths.

Basic Syntax:

pwd [OPTION]

Common Options:

pwd (no options):

When pwd is used without any options, it prints the absolute path of the current working directory, starting from the root of the filesystem.

Example:

$ pwd
/home/user/documents

pwd -L or pwd --logical:

This option prints the logical path of the current directory. It does not resolve symbolic links in the path. If the current directory is a symbolic link to another directory, pwd -L will show the path of the symbolic link itself.

Example:

$ pwd -L
/home/user/documents

pwd -P or pwd --physical:

This option prints the physical path of the current directory, resolving symbolic links along the path. If the current directory is a symbolic link, pwd -P will show the actual path of the target directory.

Example:

$ pwd -P
/mnt/storage/docs

Using the appropriate option depends on whether you want to see the physical path (resolved symbolic links) or the logical path (without resolving symbolic links) of the current directory. By default, the pwd command behaves as pwd -P, showing the physical path.

Command: ls [options] [directory]

The ls command in Linux is used to list the files and directories in a specified directory. It is one of the most commonly used commands in the Linux command-line interface for exploring the filesystem. The ls command provides various options to customize the output and display additional information about files and directories.

Basic Syntax:

ls [OPTION]... [FILE]...

Common Options:

ls (no options):

When ls is used without any options, it lists the files and directories in the current working directory (usually the directory where the user is currently located).

Example:

```
$ ls
file1.txt file2.txt directory1 directory2
```

ls -l or ls --long:

This option displays the files and directories in a detailed, long listing format. It shows additional information such as file permissions, owner, group, file size, modification date, and the name of each entry.

Example:

```
 $ ls -l
-rw-r--r-- 1 user group  1024 Jul 31 10:00 file1.txt
-rw-r--r-- 1 user group  2048 Jul 31 10:15 file2.txt
drwxr-xr-x 2 user group  4096 Jul 31 10:30 directory1
drwxr-xr-x 2 user group  4096 Jul 31 10:45 directory2
```

ls -a or ls --all:

This option includes hidden files and directories in the listing. Hidden entries in Linux start with a dot (e.g., .hidden_file).

Example:

```
$ ls -a
.  ..  file1.txt  file2.txt  .hidden_file  directory1  directory2
```

ls -h or ls --human-readable:

This option makes file sizes human-readable by displaying them in a more understandable format, using units such as KB, MB, GB, etc.

Example:

```
$ ls -lh
-rw-r--r-- 1 user group 1.0K Jul 31 10:00 file1.txt
-rw-r--r-- 1 user group 2.0K Jul 31 10:15 file2.txt
drwxr-xr-x 2 user group 4.0K Jul 31 10:30 directory1
drwxr-xr-x 2 user group 4.0K Jul 31 10:45 directory2
```

These are just a few of the options available for the ls command. To see the complete list of options and their descriptions, you can check the manual page for the ls command by running man ls in the terminal.

4. Linux Basic Commands: touch, tr, and pr

Command: touch [filename]

The touch command in Linux is used to create an empty file or update the access and modification times of an existing file. It is a versatile command that allows users to create new files quickly or modify the timestamps of existing ones.

Basic Syntax:

touch [OPTION]... FILE...

Common Options:

touch FILENAME:

When touch is used without any options, it creates an empty file with the specified filename. If the file already exists, it updates the access and modification times to the current time without modifying the file's content.

Example:

$ touch myfile.txt

touch -a or touch --time=access:

This option changes only the access time of the file, leaving the modification time unchanged.

Example:

$ touch -a myfile.txt

touch -m or touch --time=modification:

This option changes only the modification time of the file, leaving the access time unchanged.

Example:

$ touch -m myfile.txt

The touch command is a powerful tool for managing file timestamps and creating new files in a straightforward manner.

Command: tr [options] [set1] [set2]

The tr command in Linux is a versatile utility used for translating, deleting, or squeezing characters in a given input stream. It is primarily used for text manipulation and can be quite handy for simple string transformations. The name "tr" stands for "translate."

Basic Syntax:

tr [OPTION]... SET1 [SET2]

Common Options:

tr [SET1] [SET2]:

The most basic form of the tr command requires two sets of characters, SET1 and SET2. It reads the input from the standard input (stdin) and replaces characters from SET1 with the corresponding characters from SET2.

Example:

$ echo "hello" | tr 'aeiou' 'AEIOU'

hEllO

tr -d [SET] or tr --delete [SET]:

The -d option deletes all characters from the input that are listed in the SET.

Example:

$ echo "hello" | tr -d 'l'

heo

tr -s or tr --squeeze-repeats:

The -s option squeezes repeated occurrences of characters in the input to a single character. It removes consecutive duplicates.

Example:

$ echo "aaabbbccc" | tr -s 'a'

abbbccc

tr -c [SET] or tr --complement [SET]:

The -c option complements the SET specified. It replaces all characters not listed in the SET.

Example:

```
$ echo "hello" | tr -c 'a-z' 'X'
XXXXX
```

The tr command is frequently used for basic character conversions, removing unwanted characters, and replacing characters in text streams. It is particularly useful in shell scripts and one-liners for text manipulation tasks. For more advanced text processing or complex pattern matching, other tools like sed or awk may be more appropriate.

Command: pr [options] [filename]

The pr command in Linux is used for formatting and paginating text files before printing. It enables users to adjust the layout of text and control the appearance of text on printed pages. While its primary purpose is to format files for printing, the pr command can also be useful for viewing text in a more organized manner on the terminal.

Basic Syntax:

```
pr [OPTION]... [FILE]...
```

Common Options:

pr [FILE]:

When pr is used without any options, it paginates the text file and prints it on the standard output (usually the terminal) with default formatting.

Example:

```
$ pr myfile.txt
```

pr -n [NUMBER] or pr --columns=[NUMBER]:

The -n option specifies the number of columns in the output. By default, pr uses two columns. The output will be formatted to fit the specified number of columns, wrapping text appropriately.

Example:

```
$ pr -n 4 myfile.txt
```

pr -l [NUMBER] or pr --length=[NUMBER]:

The -l option sets the page length to the specified number of lines. pr breaks the output into pages based on the line length.

Example:

```
$ pr -l 30 myfile.txt
```

pr -h [HEADER] or pr --header=[HEADER]:

The -h option allows you to add a header to each page of the output.

Example:

```
$ pr -h "Report" myfile.txt
```

pr -o [NUMBER] or pr --omit-pagination=[NUMBER]:

The -o option specifies the number of initial lines to omit from pagination. The omitted lines are not included in the page count.

Example:

```
$ pr -o 5 myfile.txt
```

The pr command is useful when you want to format text files for printing or create structured and readable output on the terminal. It is often used in combination with other commands in shell scripts to produce well-organized reports or display text content with custom formatting.

Remember to explore the manual pages for these commands to discover more options and their usage.

Network-Related Commands

Linux Commands: hostname, ping, traceroute, and nmap

hostname command:

The hostname command in Linux is used to display or set the system's hostname. The hostname is a label assigned to the computer on a network, and it helps identify the machine within the network.

hostname: When used without any options, the hostname command displays the current hostname of the system.

hostname -f: The -f option stands for "fully qualified domain name." It displays the complete domain name associated with the hostname.

hostname -i: The -i option displays the IP addresses associated with the system's hostname.

hostname -s: The -s option stands for "short" or "simple." It displays only the short name (the hostname without the domain name).

Examples:

Display the current hostname:

```
hostname
```

Display the fully qualified domain name:

```
hostname -f
```

Display the IP addresses associated with the hostname:

```
hostname -i
```

Display only the short name (hostname without the domain name):

```
hostname -s
```

ping command:

The ping command in Linux is used to test network connectivity between a source and a destination. It sends Internet Control Message Protocol (ICMP) Echo Request packets to the destination and waits for ICMP Echo Reply packets to measure the round-trip time and packet loss.

ping host: The basic syntax of the ping command requires you to specify the hostname or IP address of the destination you want to test connectivity with.

ping -c count host: The -c option specifies the number of packets to send (count). It stops pinging after sending the specified number of packets.

ping -i interval host: The -i option sets the interval (in seconds) between successive packet transmissions.

ping -W timeout host: The -W option sets the timeout (in seconds) for waiting for a response before considering the packet lost.

ping -R host: The -R option records the route taken by packets in the output.
ping -6 host: The -6 option forces the use of IPv6 when pinging a host.
Examples:
Basic ping test to a host:

```
ping google.com
```

Ping a host four times:

```
ping -c 4 example.com
```

Set the interval between pings to 2 seconds:

```
ping -i 2 example.com
```

The ping command is a simple yet powerful tool for testing network connectivity. It is commonly used for diagnosing network-related issues, verifying if a host is reachable, and assessing network performance.

traceroute command:

The traceroute command in Linux is used to trace the route that packets take from the source to a destination over a network. It provides valuable information about the network path, including the IP addresses of intermediate routers and the round-trip time (latency) for each hop.
traceroute host: The basic syntax of the traceroute command requires you to specify the hostname or IP address of the destination you want to trace the route to.
traceroute -I host: The -I option forces the use of ICMP Echo Request packets instead of UDP. Some routers may block UDP traceroute, making ICMP traceroute a more reliable option.
traceroute -n host: The -n option disables hostname resolution, showing IP addresses instead of hostnames in the output.
traceroute -w timeout host: The -w option sets the timeout (in seconds) for each probe.
Examples:
Basic traceroute to a host:

```
traceroute example.com
```

Traceroute using ICMP Echo Request (ICMP-based traceroute):

```
traceroute -I example.com
```

The traceroute command is a valuable network diagnostic tool that helps identify network delays, packet loss, and routing issues. It provides insights into the path taken by packets across networks, making it useful for network troubleshooting and analysis.

nmap command:

The nmap command in Linux is a powerful network scanning tool used to discover hosts and services on a computer network. It provides a wide range of options to scan hosts for open ports, detect operating systems, and gather information about network services.
nmap target: The basic syntax of the nmap command requires you to specify the target IP address or hostname you want to scan.
nmap -p ports target: The -p option specifies the ports to scan. You can specify individual ports or port ranges separated by commas.

nmap -F target: The -F option (Fast Scan) scans only the most common 100 ports, providing a quicker scan result.
nmap -A target: The -A option enables aggressive scanning, which includes OS detection, version detection, script scanning, and traceroute.
nmap -O target: The -O option attempts to detect the remote operating system.
Examples:
Basic scan of a single target:

```
nmap 192.168.1.1
```

Scan specific ports on a target:

```
nmap -p 80,443 example.com
```

Aggressive scan with OS detection and version detection:

```
nmap -A 192.168.0.1
```

The nmap command is an essential tool for network administrators and security professionals to discover and assess the security of their networks. It helps identify open ports, services, and potential vulnerabilities that can aid in securing the network infrastructure.

Linux Basic Commands: ssh, scp and /etc/ssh

SSH (Secure Shell)

SSH (Secure Shell) is a cryptographic network protocol used to securely access and manage remote systems over an unsecured network. It provides encrypted communication between the client and the server, ensuring data confidentiality and integrity.
Syntax:

```
ssh [options] [user@]hostname [command]
```

Common Options:

```
-p port: Specify a custom SSH port (default is 22).
-i identity_file: Use a specific private key for authentication.
```

Example:

```
ssh user@example.com
Connects to the example.com server using SSH.
```

SCP (Secure Copy)

SCP (Secure Copy) is a command-line utility for securely copying files between local and remote systems using SSH for data transfer.
Syntax:

```
scp [options] [source] [destination]
```

Common Options:

```
-P port: Specify a custom SSH port (default is 22).
```

-i identity_file: Use a specific private key for authentication.
-r: Recursively copy directories and their contents.

Example:

scp file.txt user@example.com:/path/to/destination

Copies file.txt from the local system to the remote host example.com at /path/to/destination.

/etc/ssh (SSH Configuration Directory)

The /etc/ssh directory contains configuration files for the OpenSSH server and client. It allows administrators to customize SSH behavior and enhance security.

Important Files:

sshd_config: Configuration file for the SSH server (sshd).
ssh_config: Configuration file for the SSH client (ssh).
ssh_host_*: Files containing host keys used for server authentication.
ssh_known_hosts: System-wide file containing known host keys.

Note:

Modifying SSH configuration files requires administrative privileges. Always create backups before making changes.