Shell Scripting

1. Introduction to Shell Scripting
    - Definition and purpose of shell scripting
    - Common shells (e.g., Bash, C Shell)
    - Advantages of shell scripting
    - GUI
    - CLI
    - Kernel
    - Distros
2. Root Level Dir
    - Home
    - Lib
    - Bin
    - Sbin
    - Dev
    - Etc
    - Boot
    - Root
    - Usr
    - Var
3. Overview of pseudo file dir
    - Proc
    - Dev
    - Sys

4. Help Feature

- **Man**
- **Info**

5. Environment

- **Locale**
- **pwd**
- **echo**
- **Env**
- **Type**
- **Timedate**
- **Timedate ctl**

6. Working with File System

- **cd**
- **ls**
- **mkdir**
- **vi**
- **nano**
- **touch**
- **cp**
- **mv**
- **rm**
- **tar**
- **clear**
- **history**

- **chmod**
- **cat**
- **gzip**
- **Head**
- **Tail**
- **Find**
- **Diff**

7. Useful and basic commands

- **Kill**
- **grep**
- **ps**
- **wc**
- **Ipconfig**
- **Yum**
- **Wget**
- **Apt-get**
- **Curl**
- **Ping**
- **Ssh**
- **telnet**
- **Crontab**

8. Scripts

- **.SH files**
- **Variables**

- **Flow control**

1. Introduction to Shell Scripting

- **Definition and purpose of shell scripting**
- **Common shells (e.g., Bash, C Shell)**
- **Advantages of shell scripting**

- **Definition and purpose of shell scripting:** Shell scripting refers to the process of writing and executing scripts (sequences of commands) using a command-line interpreter, or shell. The shell acts as an interface between the user and the operating system, allowing users to interact with the system by executing commands. Shell scripting allows for automating repetitive tasks, performing system administration tasks, and executing a series of commands in a specific sequence.

- **Common shells (e.g., Bash, C Shell): There are several shells available, each with its own syntax and features. Two commonly used shells are:**
    - **Bash (Bourne Again SHell):** Bash is the default shell for most Linux distributions and is also available on other Unix-like systems. It is a versatile and widely-used shell with a rich set of features, including variables, loops, conditional statements, functions, and more.
    - **C Shell (csh):** The C Shell is another popular shell that provides a C-like syntax and additional features like command-line editing and history. It is commonly used in BSD-based systems.
- **Advantages of shell scripting:**
    - **Automation:** Shell scripting allows for automating repetitive tasks, saving time and effort. By writing scripts, you can create a set of commands that can be executed with a single command, reducing manual work.
    - **Flexibility:** Shell scripts are versatile and can be used for various purposes, such as system administration, file manipulation, data processing, and more. They can incorporate conditionals, loops, and other control structures to handle complex tasks.
    - **Portability:** Shell scripts are generally portable across different Unix-like systems since most systems provide compatible shells. This allows scripts to be easily executed on different platforms without significant modifications.

  - **Integration:** Shell scripts can integrate with other command-line utilities and tools, making it easy to combine and use various system commands, utilities, and programs within a script.
  - **Customization:** Shell scripts can be customized and parameterized by accepting command-line arguments or reading input from users, allowing for dynamic behavior and adaptability.

These aspects make shell scripting a powerful tool for automating tasks, performing system administration, and streamlining workflows in Unix-like environments.

- **GUI (Graphical User Interface):** GUI refers to a visual interface with buttons, menus, and windows that allows users to interact with programs or operating systems using a mouse and keyboard.
- **CLI (Command-Line Interface):** CLI is a text-based interface where users interact with the computer by typing commands in a terminal or shell.
- **Kernel:** In shell scripting, the kernel is the core part of the operating system. It manages system resources, acts as a bridge between software and hardware, and coordinates tasks. Shell scripts interact with the kernel through system calls and commands provided by the operating system. These commands allow shell scripts to perform operations like file manipulation, process management, and networking.
- **Distros (Distributions):** Distributions, often referred to as distros, are different versions or variations of operating systems based on the Linux kernel. Examples include Ubuntu, Fedora, and Debian.

Root Level Dir

1. **Home**: This directory contains personal user directories, each representing a specific user on the system. For example, "/home/john" represents the home directory of the user "john".

2. **Lib:** The "lib" directory stores shared libraries that are required by various programs on the system. For example, "/lib/libc.so.6" is a shared library used by many programs for basic functionality.

3. **Bin:** The "bin" directory contains essential executable files and commands that are available to all users. For example, "/bin/ls" is the command to list files and directories.

4. **Sbin:** The "sbin" directory holds system binaries, which are executable files that perform administrative or system-related tasks. For example, "/sbin/reboot" is a command to reboot the system.

5. **Dev:** The "dev" directory contains device files that represent various hardware devices connected to the system. For example, "/dev/sda" represents the first physical hard disk.

6. **Etc:** The "etc" directory stores system configuration files. For example, "/etc/passwd" contains user account information.

7. **Boot:** The "boot" directory contains files related to the system's boot process, such as boot loaders and kernel images. For example, "/boot/vmlinuz-5.10.0-1234" is a Linux kernel image.

8. **Root:** The "root" directory represents the root of the filesystem hierarchy. It is often denoted as "/". All other directories and files stem from this directory.

9. **Usr:** The "usr" directory contains user-related programs, libraries, documentation, and shared resources that are not essential for basic system functionality. For example, "/usr/bin/python3" is the path to the Python interpreter.

10. **var:** The "var" directory holds variable data that changes frequently, such as logs, spool files, and temporary files. For example, "/var/log/syslog" stores system log messages**.**

11. **/proc:** The /proc directory provides a view into the running processes on the system. It contains a collection of directories and files that represent each running process. Each process is assigned a numerical directory (e.g., /proc/1234/) with files containing information about the process, such as its status, memory usage, open files, command line arguments, and more. The /proc directory also includes system-related information, such as CPU information, kernel configuration, and hardware details.

Example: You can view the process ID and status of a running process by accessing the corresponding file in /proc. For example, /proc/1234/status contains information about process 1234.

12. **/sys:** The /sys directory provides an interface to the kernel's device and system information. It exposes information about hardware devices, device drivers, bus topology, and system configuration. The /sys directory is structured hierarchically, representing different devices, buses, and system components. It provides files that allow querying and configuring device-specific parameters and status.

Example: You can view information about the CPU by accessing files in /sys/devices/system/cpu. For example, /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_max_freq contains the maximum frequency of the first CPU core.

9. Help Feature

- **Man**
- **Info**

Man: Both Man and Info are almost same. These commands will allow us to access detailed documentation of various commands and topics.

Man command is used to get help and detailed documentation of various commands.

Lets say we want to learn more on ls command then if we do man ls it will give the detailed documentation of that command

Whereas,

Info command is used for accessing documentation, providing more extensive and structured information compared to man. Hyper

Environments: lets see few commands in the context of the linux environment:

1. **locale:** The locale command shows information about the language, character encoding, and cultural conventions used by the system**.**

Example: Running locale will display the current locale settings, such as the language code (e.g., en_US for English, US variant) and character encoding (e.g., UTF-8).

2. **pwd:** The pwd command prints the current working directory, which is the directory you are currently in.

Example: If you are in the directory /home/user/documents, running pwd will display /home/user/documents.

3. **echo:** The echo command displays text or variable values in the terminal.

Example: Running echo "Hello, World!" will output "Hello, World!" in the terminal.

4. **env:** The env command displays the current environment variables, which are dynamic values that control the behaviour of programs and processes.

Example: Running env will list environment variables such as PATH, which defines the directories to search for executables.

5. **type:** The type command determines the type and location of a command or executable.

Example: Running type ls will indicate that ls is a command to list directory contents and show its location, such as /bin/ls.

6. **timedatectl:** The timedatectl command manages the system's time and date settings.

Example: Using timedatectl, you can view the current system time and date settings, change the time zone, or enable automatic time synchronization.

In summary, these commands are used to access information about the system's locale settings, display the current working directory, output text, view environment variables, determine command types, and manage time and date settings.

cd: This command is used to change the current working directory.

Example: cd /home/user changes the current directory to /home/user.

ls: It lists the files and directories in the current directory.

Example: ls lists the files and directories in the current directory.

mkdir: This command is used to create a new directory.

Example: mkdir new_directory creates a directory named new_directory.

vi: It is a text editor used to create and edit files.

Example: vi myfile.txt opens the file myfile.txt in the vi editor.

nano: It is a lightweight text editor used to create and edit files.

Example: nano myfile.txt opens the file myfile.txt in the nano editor.

touch: It creates an empty file or updates the timestamp of an existing file.

Example: touch myfile.txt creates an empty file named myfile.txt.

cp: It is used to copy files and directories.

Example: cp file1.txt file2.txt copies file1.txt and creates a new file named file2.txt with the same content.

mv: This command is used to move or rename files and directories.

Example: mv file.txt new_location/ moves file.txt to the new_location directory**.**

rm: It is used to remove files and directories.

Example: rm file.txt deletes the file file.txt.

tar: It is used for archiving multiple files into a single file.

Example: tar -czvf archive.tar.gz directory/ creates a compressed tar archive of the directory named archive.tar.gz.

clear: It clears the terminal screen.

Example: clear clears the terminal screen.

history: It displays a list of previously executed commands.

Example: history shows a list of previously executed commands.

chmod: It is used to change the permissions of files and directories.

Example: chmod 755 myfile.txt sets read, write, and execute permissions for the owner and read and execute permissions for others on myfile.txt.

The first digit (7) represents the permissions for the owner of the file.

The second digit (5) represents the permissions for the gro

The third digit (5) represents the permissions for others (users not in the owner or group).

- `0` (no permission)
- `1` (execute)
- `2` (write)
- `3` (write and execute)
- `4` (read)
- `5` (read and execute)
- `6` (read and write)
- `7` (read, write, and execute)

cat: It displays the contents of a file.

Example: cat file.txt displays the contents of file.txt in the terminal.

gzip: It compresses files.

Example: gzip file.txt compresses file.txt and creates a compressed file named file.txt.gz.

head: It displays the first few lines of a file.

Example: head file.txt displays the first 10 lines of file.txt.

tail: It displays the last few lines of a file.

Example: tail file.txt displays the last 10 lines of file.txt.

find: It is used to search for files and directories.

Example: find /path/to/directory -name "*.txt" searches for all files ending with .txt in the specified directory.

diff: It compares and displays the differences between files.

Example: diff file1.txt file2.txt compares file1.txt and file2.txt and displays the differences.

Kill: The kill command is used to terminate a process by its process ID (PID). It sends a signal to the specified process, allowing you to stop or control its execution. For example, to terminate a process with PID 1234, you can use:

- **kill 123**

grep: The grep command is used to search for specific patterns within files or command output. It allows you to filter and display lines that match a given pattern.

- **grep "example" text.txt**

ps: The ps command is used to display information about running processes. It shows a snapshot of the current processes on your system. For example, to display all running processes, you can use:

- **ps -ef**

wc: The wc command is used to count the number of words, lines, and characters in a file or command output. It's often used to analyze text data.

for example, to count the number of lines in a file named data.txt, you can use:

- **wc -l data.txt**

Ipconfig: The ipconfig command (Windows) or ifconfig command (Linux/Unix) is used to display network interface information, such as IP addresses, subnet masks, and network configurations.

- **ifconfig**

Yum: The yum command is a package management tool used in Red Hat-based Linux distributions. It is used to install, update, and manage software packages and dependencies.

- **yum install nginx**

wget: The wget command is used to download files from the web using HTTP, HTTPS, or FTP protocols.

- **wget <URL>**

Apt-get: The apt-get command is a package management tool used in Debian-based Linux distributions. It is used to **install, update, and manage software packages and dependencies.**

- **apt-get install nginx**

Curl: The curl command is used to transfer data to or from a server using various protocols like HTTP, FTP, etc. It supports making requests, downloading/uploading files, and more.

- **curl <URL>**

Ping: The ping command is used to check the reachability and response time of a network host or IP address. It sends ICMP Echo Request packets to the target and measures the round-trip time.

- **ping 192.168.0.1**

Ssh: The ssh command is used to establish a secure shell connection to a remote server. It allows you to log in to and execute commands on a remote machine securely. For example, to connect to a remote server with the IP

- **ssh john@192.168.0.1**

Telnet: The telnet command is used to establish a Telnet connection to a remote server. It allows you to communicate with other devices over a network.

- **telnet 192.168.0.1 80**

Crontab: The crontab command is used to create, edit, and manage cron jobs. Cron jobs are scheduled tasks that run at specified intervals or times. To edit cron job of the current user

- **crontab -e**

.sh Files:

A .sh file is a script file written in the Bash scripting language, which is commonly used in Unix-like operating systems (e.g., Linux, macOS).

The .sh extension indicates that the file contains shell commands and is intended to be executed by a shell interpreter, such as Bash.

.sh files are used to automate tasks, execute commands, and perform various operations in a scripting environment.

You can create and edit .sh files using a text editor and run them using the command line.

Variables:

Variables in programming are used to store and manipulate data. They act as containers to hold values that can be accessed and modified during the execution of a script.

In Bash scripting, variable names are case-sensitive, conventionally written in uppercase, and assigned values using the syntax: variable_name=value.

To access the value of a variable, you can use the $ symbol followed by the variable name: $variable_name.

Variables can store various types of data, including strings, numbers, and arrays.

Example:

```
# Assigning a value to a variable
name="John"

# Accessing the value of the variable
echo "Hello, $name!"
```

Flow Control:

Flow control refers to the mechanisms used to control the execution order of commands or statements in a script.

In Bash scripting, common flow control structures include:

if statement: It allows you to perform conditional execution of code based on certain conditions.

for loop: It iterates over a sequence of values and executes a block of code for each iteration.

while loop: It repeatedly executes a block of code as long as a specified condition is true.

until loop: It repeatedly executes a block of code until a specified condition becomes true.

case statement: It enables you to perform different actions based on the value of a variable or expression.

These flow control structures provide flexibility and control over the execution flow of your script, allowing you to make decisions and repeat actions based on certain conditions.

Example .SH file

```
#!/bin/bash

# Example .sh file

# Variables
name="John"
age=25

# Flow control (if statement)
if [ $age -ge 18 ]; then
  echo "Hello, $name! You are an adult."
else
  echo "Hello, $name! You are a minor."
fi
```

Explanation:

It starts with the shebang (#!/bin/bash) to specify the interpreter.

The variables name and age are defined with corresponding values.

An if statement checks if the age is greater than or equal to 18 and displays an appropriate message based on the condition.

Command to Execute the Shell Scripting

```
Bash ./file.sh

Chmod +x file.sh
```