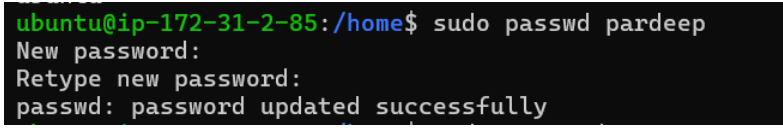
**SHELL SCRIPTING**

* netstat -a : Displays all connections and listening ports in a server.
* Ipconfig : You need to verify the IP address of your network interface.
* iwconfig: Checking Wireless Network Status i.e., Troubleshooting a slow Wi-Fi connection.
* top: Monitoring server performance in real-time, : Provides a dynamic view of system performance, including CPU and memory usage.
* du: Shows disk usage for each directory, helping you find large directories.eg: du -sh /var/log/\*
* apt: Used by Debian-based distributions like Ubuntu.
* yum and dnf: Used by RPM-based distributions like Fedora and CentOS/RHEL (dnf is the newer version).
* pacman: Used by Arch Linux and its derivatives.
* portage: Used by Gentoo Linux.
* passwd: This command is used to change a user's password.



**File permission commands:**

* umask**:** The umask command sets the default file permissions for newly created files and directories.

Syntax: umask [mode]

Eg: umask 022 sets the default permission for newly created files to 644 and for directories to 755.

* chgrp: The chgrp command is used to change the group ownership of files and directories.

Syntax: chgrp [options] group file

Eg: chgrp group file.txt changes the group owner of the file file.txt to group.

**File transfer commands:**

* scp (Secure Copy): scp is a command-line tool used to securely transfer files between a local and a remote host or between two remote hosts over SSH (Secure Shell).
* Syntax for copying from local to remote: scp [options] local\_file remote\_user@remote\_host:remote\_path
* Syntax for copying from remote to local: scp [options] remote\_user@remote\_host:remote\_path local\_path
* Example: scp file.txt user@example.com:/remote/directory/ copies the file "file.txt" from the local system to the remote host "example.com" into the directory "/remote/directory/".
* rsync: rsync is a powerful command-line utility used for file synchronization and transfer. It is particularly useful for synchronizing files between local and remote systems or between directories on the same system.
* Syntax: rsync [options] source destination.
* Example: rsync -avz /local/directory/ user@example.com:/remote/directory/ synchronizes the contents of the local directory "/local/directory/" with the directory "/remote/directory/" on the remote host "example.com".

**Linux built in variables:**

#!/bin/bash

# This script demonstrates the use of built-in variables in bash.

# Display the name of the script

echo "Script name: $0"

# Display the number of arguments passed to the script

echo "Number of arguments: $#"

# Display the current process ID

echo "Current process ID: $$"

# Display the user executing the script

echo "User executing the script: $USER"

# Display the current working directory

echo "Current working directory: $PWD"

**Wild Cards:**

#!/bin/bash

# This script lists all files with a specific extension in a directory.

# Specify the directory to search (you can change this to any directory)

DIRECTORY="/path/to/directory" # Replace with the actual directory path

# Specify the file extension to look for

EXTENSION="txt" # Change this to the desired file extension

# Use wildcards to list all files with the specified extension in the directory

echo "Listing all .$EXTENSION files in the directory $DIRECTORY:"

ls "$DIRECTORY"/\*.$EXTENSION

# Alternatively, you can use a loop to list the files

echo "Using a loop to list the files:"

for file in "$DIRECTORY"/\*.$EXTENSION;

do

echo "$file"

done

**Log Analyzer and Report Generator using shell script**

Write a Bash script that automates the process of analyzing log files and generating a daily summary report. The script should perform the following steps:

Input: The script should take the path to the log file as a command-line argument.

Error Count: Analyze the log file and count the number of error messages. An error message can be identified by a specific keyword (e.g., "ERROR" or "Failed"). Print the total error count.

Critical Events: Search for lines containing the keyword "CRITICAL" and print those lines along with the line number.

Top Error Messages: Identify the top 5 most common error messages and display them along with their occurrence count.

Summary Report: Generate a summary report in a separate text file. The report should include:

* Date of analysis
* Log file name
* Total lines processed
* Total error count
* Top 5 error messages with their occurrence count
* List of critical events with line numbers

**Script:**

#!/bin/bash

# Here we are showing how to use log file

if [ -z "$1" ]; then

echo "Usage: $0 <path\_to\_log\_file>"

exit 1

fi

# Declare variable for first argument

log\_file=$1

report\_file="summary\_report\_$(date +%Y%m%d).txt" # Create file using custom timestamp

# Check if log file is exist or not

if [ ! -f "$log\_file" ]; then

echo "Log file not found!"

exit 1

fi

# Counting lines in log file

total\_lines=$(wc -l < "$log\_file")

# Counting number of lines containing words ERROR & Failed

error\_count=$(grep -cE "ERROR|Failed" "$log\_file")

# Finds line include CRITICAL and line number using -n

critical\_events=$(grep -n "CRITICAL" "$log\_file")

# Declare an associative array to store error\_messages

declare -A error\_messages

# Read log file line by line

while IFS= read -r line; do

if [[ "$line" =~ ERROR|Failed ]]; then

error\_message=$(echo "$line" | awk -F']' '{print $NF}')

((error\_messages["$error\_message"]++))

fi

done < "$log\_file"

# Sorting of errors

top\_errors=$(for message in "${!error\_messages[@]}"; do

echo "${error\_messages[$message]} $message"

done | sort -rn | head -n 5)

# Printing final outout data

echo "Date of analysis: $(date)" > "$report\_file"

echo "Log file name: $log\_file" >> "$report\_file"

echo "Total lines processed: $total\_lines" >> "$report\_file"

echo "Total error count: $error\_count" >> "$report\_file"

echo "Top 5 error messages with their occurrence count:" >> "$report\_file"

echo "$top\_errors" >> "$report\_file"

echo "List of critical events with line numbers:" >> "$report\_file"

echo "$critical\_events" >> "$report\_file"

echo "Analysis complete. Report generated: $report\_file".

**Error Handling in Shell Scripting**

**Task 1 : Checking Exit Status:**

#!/bin/bash

# User input to create directory

read -p "Enter directory that you want to create: " dir

# Create directory

mkdir $dir

# Check if previous command executed or not

if [ $? -ne '0' ]; then

echo "Sorry :( Failed to create directory $dir"

exit 1

else

echo "Directory '$dir' created successfully:)"

fi