**Linux Commands**

1. **A) Provide the Linux command for identifying and terminating a specific process using its Process ID (PID).**

**Kill 1234**

**B) Write Script to see current date, time, username, and current directory.**

**#!/bin/bash**

**# Display current date**

**echo "Current Date: $(date '+%Y-%m-%d')"**

**# Display current time**

**echo "Current Time: $(date '+%H:%M:%S')"**

**# Display username**

**echo "Username: $USER"**

**# Display current directory**

**echo "Current Directory: $(pwd)"**

2. **List the essential Linux utility commands**

* **Find**
* **Date**
* **Mount**
* **Time**
* **Cal**

Certainly! Here's a brief description of each of the listed essential Linux utility commands:

1. **find**: The **find** command is used to search for files and directories in a directory hierarchy based on various criteria such as file name, size, modification time, and more.

Example:

bashCopy code

find /path/to/search -name "\*.txt"

1. **date**: The **date** command is used to display or set the current date and time. It is also often used in shell scripts to generate timestamps.

Example:

bashCopy code

date

1. **mount**: The **mount** command is used to mount filesystems or storage devices to specified mount points in the Linux file system.

Example:

bashCopy code

mount /dev/sdb1 /mnt/data

1. **time**: The **time** command is used to measure the execution time of a command or script. It provides information about the real, user, and system time taken by a process.

Example:

bashCopy code

time some\_command

1. **cal**: The **cal** command is used to display a calendar for a specific month or year in the terminal.

Example:

bashCopy code

cal

These commands are fundamental utilities in Linux and are commonly used for various tasks such as file management, time-related operations, filesystem handling, and calendar display.

1. **Given the output of the command ls -l, analyze and provide information about the files and directories listed, including permissions, owner, size, and modification time.**
2. **A) Write the linux command to show the use of AWK command.**

**B) Using appropriate commands, create a file named "report.txt," add some text to it, and then copy it to a subdirectory named "backup."**

**5. Write the functionalities of Linux file content commands such as tac, tail, head, rename, and more.**

**6. Write the functionalities of the rename, mv, cp, rm, cat, and touch commands within the realm of Linux file operations.**

1. **Implement the features for Linux directory commands, encompassing functions such as pwd, mkdir, rmdir, ls, and cd.**

**8.Write Linux commands for the following.**

1. **To create a directory in your home directory having 2 subdirectories.**
2. **In the first subdirectory, create 3 different files with different content in each of them.**
3. **Copy the first file from the first subdirectory to the second subdirectory.**
4. **Create one more file in the second subdirectory, which has the output of the number of users and number of files.**
5. **To list all the files which starts with either a or A.**
6. **Execute the following Linux commands:**
7. **Create a new file named "confidential.txt" within the "docs" directory.**
8. **Modify the permissions of "confidential.txt" to allow read and write access only to the owner.**
9. **Present the current date and time of the system.**
10. **Examine the active processes on the system.**
11. **Identify and terminate the process with the highest CPU usage.**
12. **Compose Linux commands for the following operations:**

* Utilize **uniq.**
* Implement **comm.**
* Apply **egrep.**
* Utilize **sed.**
* Make use of **zcat.**

**Shell Scripts**

1. **Write a shell script to evaluate arithmetic operations.**
2. **Develop a shell script to ascertain whether a specified year is a leap year or not.**
3. **Develop a shell script that calculates the factorial (n!) of a given number.**
4. **Create a shell script to compute the simple interest for a given principal amount, interest rate, and time period.**
5. **Write a Shell Script to generate Fibonacci Series.**
6. **Write a shell script to add the digits of a number.**
7. **Write a shell script to check whether a number entered is prime or not.**
8. **Create a shell script to determine whether a provided number is an Armstrong number or not.**
9. **Craft a shell script to produce a multiplication table.**
10. **Create a shell script to demonstrate the use of a switch-case construct.**

**4A. Example of AWK Command:**

The **awk** command is a powerful text processing tool in Unix and Linux. Here's a simple example demonstrating the use of **awk** to print the second and third columns of a file:

bashCopy code

awk '{print $2, $3}' filename.txt

Replace **filename.txt** with the actual name of your file. This command will print the values in the second and third columns of each line in the specified file.

**4B. File Manipulation Commands:**

**tac**

* Displays the content of a file in reverse order (lines are reversed).

**tail**

* Outputs the last part of a file.

bashCopy code

tail filename.txt

**head**

* Outputs the first part of a file.

bashCopy code

head filename.txt

**rename**

* Renames files according to specified rules.

bashCopy code

rename 's/old\_pattern/new\_pattern/' \*.txt

**mv**

* Moves or renames files and directories.

bashCopy code

mv file.txt new\_location/

**cp**

* Copies files or directories.

bashCopy code

cp file.txt backup/

**rm**

* Removes (deletes) files or directories.

bashCopy code

rm file.txt

**cat**

* Concatenates and displays the content of files.

bashCopy code

cat file1.txt file2.txt

**touch**

* Updates the access and modification times of a file or creates an empty file.

bashCopy code

touch new\_file.txt

**7. Linux Directory Commands:**

**pwd**

* Displays the current working directory.

**mkdir**

* Creates a new directory.

bashCopy code

mkdir new\_directory

**rmdir**

* Removes an empty directory.

bashCopy code

rmdir empty\_directory

**ls**

* Lists the contents of a directory.

bashCopy code

ls

**cd**

* Changes the current directory.

bashCopy code

cd /path/to/directory

**8. Create Subdirectories and Files:**

bashCopy code

# i. mkdir ~/main\_directory cd ~/main\_directory mkdir subdirectory1 subdirectory2 # ii. cd subdirectory1 echo "Content for file1" > file1.txt echo "Content for file2" > file2.txt echo "Content for file3" > file3.txt # iii. cp file1.txt ../subdirectory2/ # iv. cd ../subdirectory2 echo "Number of users: $(who | wc -l)" > stats.txt echo "Number of files: $(ls -l | grep '^-' | wc -l)" >> stats.txt # v. ls -l | grep -iE '^[aA]'

**9. Execute Linux Commands:**

bashCopy code

# i. touch ~/docs/confidential.txt # ii. chmod 600 ~/docs/confidential.txt # iii. date # iv. ps aux # v. kill -9 $(ps aux --sort=-%cpu | awk 'NR==2{print $2}')

**10. Linux Commands for Operations:**

* **uniq**: Removes or displays duplicate lines.
* **comm**: Compares sorted files line by line.
* **egrep**: Extended grep for pattern matching.
* **sed**: Stream editor for filtering and transforming text.
* **zcat**: Displays the contents of compressed files.

**Shell Scripts:**

1. Arithmetic Operations:

bashCopy code

#!/bin/bash echo "Enter two numbers:" read num1 read num2 sum=$((num1 + num2)) echo "Sum: $sum"

2. Leap Year Check:

bashCopy code

#!/bin/bash echo "Enter a year:" read year if [ $((year % 4)) -eq 0 -a $((year % 100)) -ne 0 -o $((year % 400)) -eq 0 ]; then echo "$year is a leap year." else echo "$year is not a leap year." fi

3. Factorial Calculation:

bashCopy code

#!/bin/bash echo "Enter a number:" read num factorial=1 for ((i=1; i<=num; i++)); do factorial=$((factorial \* i)) done echo "Factorial of $num is: $factorial"

4. Simple Interest Calculation:

bashCopy code

#!/bin/bash echo "Enter principal amount:" read principal echo "Enter interest rate:" read rate echo "Enter time period (in years):" read time interest=$((principal \* rate \* time / 100)) echo "Simple Interest: $interest"

5. Fibonacci Series:

bashCopy code

#!/bin/bash echo "Enter the number of terms in the Fibonacci series:" read n a=0 b=1 echo "Fibonacci Series:" for ((i=0; i<n; i++)); do echo -n "$a " temp=$((a + b)) a=$b b=$temp done echo

6. Add Digits of a Number:

bashCopy code

#!/bin/bash echo "Enter a number:" read num sum=0 while [ $num -gt 0 ]; do digit=$((num % 10)) sum=$((sum + digit)) num=$((num / 10)) done echo "Sum of digits: $sum"

7. Prime Number Check:

bashCopy code

#!/bin/bash echo "Enter a number:" read num is\_prime=true for ((i=2; i<=num/2; i++)); do if [ $((num % i)) -eq 0 ]; then is\_prime=false break fi done if [ $is\_prime = true ]; then echo "$num is a prime number." else echo "$num is not a prime number." fi

8. Armstrong Number Check:

bashCopy code

#!/bin/bash echo "Enter a number:" read num temp=$num num\_digits=${#num} sum=0 while [ $temp -gt 0 ]; do digit=$((temp % 10)) sum=$((sum + digit\*\*num\_digits)) temp=$((temp / 10)) done if [ $sum -eq $num ]; then echo "$num is an Armstrong number." else echo "$num is not an Armstrong number." fi

9. Multiplication Table:

bashCopy code

#!/bin/bash echo "Enter a number to generate its multiplication table:" read num echo "Multiplication Table for $num:" for ((i=1; i<=10; i++)); do echo "$num x $i = $((num \* i))" done

10. Switch-Case Construct:

bashCopy code

#!/bin/bash echo "Choose an option:" echo "1. Option 1" echo "2. Option 2" echo "3.

User

**You**

do the last qn

ChatGPT

**ChatGPT**

Certainly! Here's the completion of the last shell script for the switch-case construct:

bashCopy code

#!/bin/bash echo "Choose an option:" echo "1. Option 1" echo "2. Option 2" echo "3. Option 3" echo "4. Quit" read choice case $choice in 1) echo "You chose Option 1." # Add your commands for Option 1 here ;; 2) echo "You chose Option 2." # Add your commands for Option 2 here ;; 3) echo "You chose Option 3." # Add your commands for Option 3 here ;; 4) echo "Quitting the script." exit 0 ;; \*) echo "Invalid choice. Please choose a valid option." ;; esac

Replace the comments with the specific commands you want to execute for each option. This script provides a menu with four options and executes the corresponding code block based on the user's choice. The script quits if the user chooses option 4.