|  |  |
| --- | --- |
| 1 | Write a shell script to show various system configuration like currently logged userand his logname, your current shell, home directory, operating system type, current path setting, current working directory, show currently logged number of users, show memory information, Hard disk information  like size of hard-disk, cache memory, model etc, and file system mounted.  # Display currently logged in user and logname  echo "Currently logged in user: $USER"  echo "Logname: $LOGNAME"  # Display current shell  echo "Current shell: $SHELL"  # Display home directory  echo "Home directory: $HOME"  # Display operating system type  echo "Operating system type: $(uname -o)"  # Display current path setting  echo "Current path setting: $PATH"  # Display current working directory  echo "Current working directory: $PWD"  # Display currently logged in users  echo "Currently logged in users: $(who | wc -l)"  # Display memory information  echo "Memory information:"  free -m  # Display hard disk information  echo "Hard disk information:"  df -h  # Display file system mounted  echo "File system mounted:"  mount |
| 2 | Write a shell script to add user and password on Linux system.  # Prompt for username and password  read -p "Enter username: " username  read -s -p "Enter password: " password  echo  # Add user  sudo useradd -m $username  # Set password for user  echo "$username:$password" | sudo chpasswd |
| 3 | Write a shell script to print last login details.  echo "Last login details:"  last |
| 4 | Write a shell script to upgrade and cleans the system automatically instead of doing it manually.  # Update package lists  sudo apt-get update  # Upgrade packages  sudo apt-get upgrade -y  # Remove unused packages and dependencies  sudo apt-get autoremove -y  # Clean up package cache  sudo apt-get clean |
| 5 | Write a shell script to delete all log files present inside your var/log directory.  # Delete log files in /var/log directory  sudo rm /var/log/\*.log |
| 6 | Write a script that accepts the hostname and IP address as command-line arguments and adds them to  the /etc/hosts file.  if [ $# -ne 2 ]; then  echo "Usage: $0 <hostname> <ip\_address>"  exit 1  fi  # Get hostname and IP address arguments  hostname=$1  ip\_address=$2  # Append hostname and IP address to /etc/hosts file  echo "$ip\_address $hostname" | sudo tee -a /etc/hosts |
| 7 | Write a awk script to find the number of characters, words and lines in a file?  BEGIN {  # Initialize counts to zero  chars = 0  words = 0  lines = 0  }  {  # Increment line count  lines++  # Count characters and words in line  chars += length  words += NF  }  END {  # Print counts  printf "Number of characters: %d\n", chars  printf "Number of words: %d\n", words  printf "Number of lines: %d\n", lines  } |
| 8 | Write a shell script that delete all lines containing a specified word  if [ $# -ne 2 ]; then  echo "Usage: $0 <filename> <word>"  exit 1  fi  filename=$1  word=$2  # Use grep to find lines containing the word, and use sed to delete them  grep -v "$word" "$filename" | sudo tee "$filename" >/dev/null |
| 9 | write a shell script to find the factorial of given integer  if [ $# -ne 1 ]; then  echo "Usage: $0 <integer>"  exit 1  fi  # Read in the integer argument  n=$1  # Initialize the factorial to 1  factorial=1  # Calculate the factorial  for (( i=1; i<=$n; i++ ))  do  factorial=$((factorial \* i))  done  # Print the factorial  echo "Factorial of $n is $factorial" |
| 10 | **Configuration of DNS Server with Domain Name.** |
| 11 | **Configuration of NFS File server and transfer files to a windows client.** |
| 12 | **Configuration of Web Server.** |
| 13 | **Configuration of FTP server and transfer files to demonstrate the working of the same.** |
| 14 | **Using Sed Editor Perform the Following (Any 5)**   1. Replacing or substituting string 2. Replacing the nth occurrence of a pattern in a line 3. Replacing all the occurrence of the pattern in a line 4. Replacing from nth occurrence to all occurrences in a line 5. Parenthesize first character of each word 6. Replacing string on a specific line number 7. Duplicating the replaced line with /p flag 8. Printing only the replaced lines 9. Replacing string on a range of lines 10. Deleting lines from a particular file   # 1. Replacing or substituting string  sed 's/string/replace/g' file.txt > new\_file.txt  # 2. Replacing the nth occurrence of a pattern in a line  sed 's/pattern/replace/2' file.txt > new\_file.txt  # 3. Replacing all the occurrence of the pattern in a line  sed 's/pattern/replace/g' file.txt > new\_file.txt  # 4. Replacing from nth occurrence to all occurrences in a line  sed 's/pattern/replace/3g' file.txt > new\_file.txt  # 5. Parenthesize first character of each word  sed 's/\b\(.\)/(\1)/g' file.txt > new\_file.txt  # 6. Replacing string on a specific line number  sed '3s/string/replace/g' file.txt > new\_file.txt  # 7. Duplicating the replaced line with /p flag  sed 's/string/replace/gp' file.txt > new\_file.txt  # 8. Printing only the replaced lines  sed -n 's/string/replace/pg' file.txt > new\_file.txt  # 9. Replacing string on a range of lines  sed '2,5s/string/replace/g' file.txt > new\_file.txt  # 10. Deleting lines from a particular file  sed '/pattern/d' file.txt > new\_file.txt |
| 15 | **Using awk perform the following**   1. To print the first item ($1) and then the second last item $(NF-1) from each line in sample.txt. 2. To print non-empty line from a file. 3. To print the length of the longest input line. 4. To print seven random numbers from zero to 100, inclusive. 5. To count the lines in a file   # 1. To print the first item ($1) and then the second last item $(NF-1) from each line in sample.txt.  awk '{ print $1, $(NF-1) }' sample.txt  # 2. To print non-empty line from a file.  awk 'NF' file.txt  # 3. To print the length of the longest input line.  awk '{ if (length > max\_length) max\_length = length } END { print max\_length }' file.txt  # 4. To print seven random numbers from zero to 100, inclusive.  awk 'BEGIN { srand(); for (i=1; i<=7; i++) print int(rand() \* 101) }'  # 5. To count the lines in a file.  awk 'END { print NR }' file.txt |
| 16 | **Using awk Script Preform the following (any 5)**   1. To print the first item along with the row number(NR) separated with ” – “ from each line in Tsecextc.txt: 2. To return the second row/item from Tsecextc.txt: 3. To print any non empty line if present 4. To find the length of the longest line present in the file: 5. To count the lines in a file: 6. Printing lines with more than 10 characters: 7. To find/check for any string in any specific column: 8. To print the squares of first numbers from 1 to n say 6:   # 1. To print the first item along with the row number(NR) separated with ” – “ from each line in Tsecextc.txt:  awk '{ print $1" - "NR }' Tsecextc.txt  # 2. To return the second row/item from Tsecextc.txt:  awk 'NR==2 { print $2 }' Tsecextc.txt  # 3. To print any non empty line if present:  awk 'NF' file.txt  # 4. To find the length of the longest line present in the file:  awk '{ if (length > max\_length) max\_length = length } END { print max\_length }' Tsecextc.txt  # 5. To count the lines in a file:  awk 'END { print NR }' Tsecextc.txt  # 6. Printing lines with more than 10 characters:  awk 'length > 10' file.txt  # 7. To find/check for any string in any specific column:  awk '$3 ~ /string\_to\_find/ { print }' file.txt  # 8. To print the squares of first numbers from 1 to n say 6:  awk 'BEGIN { for (i=1; i<=6; i++) print i, i\*i }' |
| 17 | Write awk Script to find square and cube of the numbers between 1 to 10  Print number , Square and Cube of the numbers  # set the field separator to a space  BEGIN { FS=" " }  # loop through the numbers from 1 to 10  for (i=1; i<=10; i++) {  # calculate the square and cube of the number  square = i \* i  cube = i \* i \* i  # print the number, square and cube  printf("%d %d %d\n", i, square, cube)  } |
| 18 | Write Shell Script to find square and cube of the numbers between 1 to 10  Print number , Square and Cube of the numbers  echo "Number Square Cube"  echo "------ ------ ----"  for (( i=1; i<=10; i++ ))  do  square=$(( i\*i ))  cube=$(( i\*i\*i ))  printf "%-7d %-7d %-7d\n" "$i" "$square" "$cube"  done |
| 19 | Write Bash script to display name and a greeting message with some information  name="John"  age=30  occupation="Software Developer"  echo "Hello, my name is $name."  echo "I am $age years old."  echo "I work as a $occupation."  echo "Nice to meet you!" |
| 20 | Write shell Script to display Following information using environment variables  like SHELL,USER,PWD,PATH,LANG,HOME,BASH\_VERSION,BASH\_VERSINFO, HOSTNAME,UID etc  echo "SHELL: $SHELL"  echo "USER: $USER"  echo "PWD: $PWD"  echo "PATH: $PATH"  echo "LANG: $LANG"  echo "HOME: $HOME"  echo "BASH\_VERSION: $BASH\_VERSION"  echo "BASH\_VERSINFO: ${BASH\_VERSINFO[@]}"  echo "HOSTNAME: $HOSTNAME"  echo "UID: $UID" |
| 21 | Write Menu Drive program using Shell script to Perform Following operation like Addition,  Subtraction , multiplication, Division, remainder  echo "Choose an operation:"  echo "1. Addition"  echo "2. Subtraction"  echo "3. Multiplication"  echo "4. Division"  echo "5. Remainder"  echo "6. Exit"  read choice  case $choice in  1)  echo "Enter first number:"  read num1  echo "Enter second number:"  read num2  sum=$((num1+num2))  echo "Result: $sum"  ;;  2)  echo "Enter first number:"  read num1  echo "Enter second number:"  read num2  diff=$((num1-num2))  echo "Result: $diff"  ;;  3)  echo "Enter first number:"  read num1  echo "Enter second number:"  read num2  prod=$((num1\*num2))  echo "Result: $prod"  ;;  4)  echo "Enter dividend:"  read dividend  echo "Enter divisor:"  read divisor  if [ $divisor -eq 0 ]; then  echo "Cannot divide by zero"  else  quotient=$((dividend/divisor))  echo "Result: $quotient"  fi  ;;  5)  echo "Enter dividend:"  read dividend  echo "Enter divisor:"  read divisor  if [ $divisor -eq 0 ]; then  echo "Cannot divide by zero"  else  remainder=$((dividend%divisor))  echo "Result: $remainder"  fi  ;;  6)  exit  ;;  \*)  echo "Invalid choice"  ;;  esac |
| 22 | Write Shell Script to find maximum of three numbers, read number from user  echo "Enter three numbers: "  read num1  read num2  read num3  if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]  then  echo "$num1 is the maximum."  elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]  then  echo "$num2 is the maximum."  else  echo "$num3 is the maximum."  fi |
| 23 | Write Shell script using for loop to generate fibonacci Series till the limit specified by user  echo "Enter the limit: "  read limit  num1=0  num2=1  echo "Fibonacci Series up to $limit: "  for (( i=0; i<=limit; i++ ))  do  echo -n "$num1 "  temp=$num1  num1=$num2  num2=$((temp+num2))  done |
| 24 | Write shell script to perform Following String Operations   1. Checks if the given string operand size is non-zero 2. Checks if the given string operand size is zero 3. Checks if the value of two operands are equal 4. Checks if the value of two operands are not equal 5. Checks if str is not the empty string; if it is empty   # Check if string operand size is non-zero  echo "Enter a string:"  read str  if [[ -n $str ]]; then  echo "String operand size is non-zero."  else  echo "String operand size is zero."  fi  # Check if string operand size is zero  if [[ -z $str ]]; then  echo "String operand size is zero."  else  echo "String operand size is non-zero."  fi  # Check if the value of two operands are equal  echo "Enter two strings:"  read str1  read str2  if [[ $str1 == $str2 ]]; then  echo "The values of the two operands are equal."  else  echo "The values of the two operands are not equal."  fi  # Check if the value of two operands are not equal  if [[ $str1 != $str2 ]]; then  echo "The values of the two operands are not equal."  else  echo "The values of the two operands are equal."  fi  # Check if str is not the empty string  if [[ -n $str1 ]]; then  echo "The string is not empty."  else  echo "The string is empty."  fi |
| 25 | Write Shell Script to perform Following Operations(Any 10)   1. Checks if file is a block special file 2. Checks if file is a character special file 3. Checks if file is a directory 4. Checks if file is an ordinary file as opposed to a directory or special file 5. Checks if file has its set group ID (SGID) bit set 6. Checks if file has its sticky bit set 7. Checks if file is a named pipe 8. Checks if file descriptor is open and associated with a terminal 9. Checks if file has its Set User ID (SUID) bit set 10. Checks if file is readable 11. Checks if file is writable 12. Checks if file is executable 13. Checks if file has size greater than 0 14. Checks if file exists   # Prompt the user to enter a file name  read -p "Enter a file name: " file  # Check if file exists  if [ -e "$file" ]  then  echo "File exists."  # Check if file is a block special file  if [ -b "$file" ]  then  echo "File is a block special file."  fi  # Check if file is a character special file  if [ -c "$file" ]  then  echo "File is a character special file."  fi  # Check if file is a directory  if [ -d "$file" ]  then  echo "File is a directory."  fi  # Check if file is an ordinary file  if [ -f "$file" ]  then  echo "File is an ordinary file."  fi  # Check if file has its set group ID (SGID) bit set  if [ -g "$file" ]  then  echo "File has its SGID bit set."  fi  # Check if file has its sticky bit set  if [ -k "$file" ]  then  echo "File has its sticky bit set."  fi  # Check if file is a named pipe  if [ -p "$file" ]  then  echo "File is a named pipe."  fi  # Check if file descriptor is open and associated with a terminal  if [ -t "$file" ]  then  echo "File descriptor is open and associated with a terminal."  fi  # Check if file has its Set User ID (SUID) bit set  if [ -u "$file" ]  then  echo "File has its SUID bit set."  fi  # Check if file is readable  if [ -r "$file" ]  then  echo "File is readable."  fi  # Check if file is writable  if [ -w "$file" ]  then  echo "File is writable."  fi  # Check if file is executable  if [ -x "$file" ]  then  echo "File is executable."  fi  # Check if file has size greater than 0  if [ -s "$file" ]  then  echo "File has size greater than 0."  fi |

|  |  |
| --- | --- |
| 26 | Write Bash script find factorial of all the number using Loop, number is to be read from user  echo "Enter numbers to find factorial (separated by space):"  read -a numbers  for number in "${numbers[@]}"  do  factorial=1  for (( i=1; i<=$number; i++ ))  do  factorial=$((factorial\*i))  done  echo "Factorial of $number is $factorial"  done |
| 27 | Write bash script to find sum of square of n numbers, read n from user  echo "Enter the value of n:"  read n  sum=0  for ((i=1;i<=n;i++))  do  square=$((i\*i))  sum=$((sum+square))  done  echo "The sum of the square of $n numbers is $sum" |
| 28 | Write Bash script to find whether character is vowel , consonant, Special Character or Digit use switch  Case  read -p "Enter a single character: " char  case "$char" in  [AEIOUaeiou]) echo "Vowel" ;;  [BCDFGHJKLMNPQRSTVWXYZbcdfghjklmnpqrstvwxyz]) echo "Consonant" ;;  [0-9]) echo "Digit" ;;  \*) echo "Special Character" ;;  esac |
| 29 | Write Bash script to find whether character is vowel , consonant, Special Character or Digit use if else  echo "Enter a character:"  read ch  if [[ $ch == [aeiouAEIOU] ]]; then  echo "$ch is a vowel."  elif [[ $ch == [[:alpha:]] ]]; then  echo "$ch is a consonant."  elif [[ $ch == [[:digit:]] ]]; then  echo "$ch is a digit."  else  echo "$ch is a special character."  fi |
| 30 | Write a shell program to check if a given string is a palindrome or not.  read -p "Enter a string: " input  # Remove all non-alphanumeric characters and convert to lowercase  input=$(echo "$input" | tr -d '[:punct:]' | tr '[:upper:]' '[:lower:]')  # Reverse the string  reverse=$(echo "$input" | rev)  # Check if the input string and its reverse are the same  if [ "$input" = "$reverse" ]; then  echo "The string is a palindrome."  else  echo "The string is not a palindrome."  fi |
| 31 | Write shell script to demonstrate command line arguments  $ ./script.sh 5 10  The script name is: ./script.sh  The first argument is: 5  The second argument is: 10  All arguments passed are: 5 10  The sum of 5 and 10 is: 15 |
| 32 | Write shell script to demonstrate Sort command with different sort command option sort –b,sort –r,sort –o,sort –n,sort –M,sort –u,sort -ksort -t SEP  # create a sample text file  echo "apple  ball  cat  dog  Elephant" > sample.txt  # sort the lines in alphabetical order  echo "Sorted in alphabetical order:"  sort sample.txt  # sort the lines in reverse order  echo "Sorted in reverse order:"  sort -r sample.txt  # sort the lines numerically  echo "Sorted numerically:"  echo "5 2 10 1 8" | sort -n  # sort the lines by month abbreviation  echo "Sorted by month abbreviation:"  echo "Feb Jan Mar Apr" | sort -M  # sort the lines and remove duplicates  echo "Sorted and duplicates removed:"  echo "apple ball cat dog dog" | sort -u  # sort the lines by the second field (separated by colon)  echo "Sorted by the second field:"  echo "apple:2 ball:1 cat:3 dog:5 elephant:4" | sort -t ':' -k 2  # sort the lines ignoring leading whitespace  echo "Sorted ignoring leading whitespace:"  echo " apple  ball  cat  dog  Elephant" | sort -b  # sort the lines and save the output to a new file  sort -o sorted.txt sample.txt  echo "Output saved to sorted.txt file." |
| 33 | Write shell script to display number of character, words and Lines in text file using wc Command  if [ -z "$1" ]  then  echo "Please specify a file to process."  exit 1  fi  if [ ! -f "$1" ]  then  echo "File not found: $1"  exit 1  fi  file=$1  # Count the number of characters, words, and lines in the file  char\_count=$(wc -c < $file)  word\_count=$(wc -w < $file)  line\_count=$(wc -l < $file)  echo "Character count: $char\_count"  echo "Word count: $word\_count"  echo "Line count: $line\_count" |