**SHELL SCRIPT**

**Example-1: Write a script to read two or multiple integer values and display them.**

#!/bin/bash

# Prompt the user for input

read -p "Enter two or multiple integer values, separated by spaces: " input\_str

# Split the input string into individual values

values=($input\_str)

# Convert the values to integers

int\_values=()

for val in "${values[@]}"; do

int\_values+=("$val")

done

# Print the integer values

echo "The integer values are: ${int\_values[@]}"

**Example-2: Write a script to store the command to a variable and execute it in a shell script.**

#!/bin/bash

ls

pwd

var1=`pwd`

echo " present working directory is : $var1"

**Example-3: Write a script to find the sum/multiply/division of two numbers.**

#!/bin/bash

# Prompt the user to enter the first number

read -p "Enter the first number: " num1

# Prompt the user to enter the second number

read -p "Enter the second number: " num2

# Calculate the sum

sum=$((num1 + num2))

# Calculate the product

product=$((num1 \* num2))

# Calculate the quotient (division)

quotient=$((num1 / num2))

# Display the results

echo "The sum of $num1 and $num2 is: $sum"

echo "The product of $num1 and $num2 is: $product"

echo "The division of $num1 by $num2 is: $quotient"

**Example-4: 3 types of conditional statements:**

1. if statement: The if statement is used to execute a block of code if a certain condition is true. The basic syntax of an if statement is as follows:

if [ condition ]

then

# Code to be executed if the condition is true

fi

1. if-else statement: The if-else statement is used to execute one block of code if a condition is true, and another block of code if it is false. The basic syntax of an if-else statement is as follows:

if [ condition ]

then

# Code to be executed if the condition is true

else

# Code to be executed if the condition is false

fi

1. if-elif-else statement: The if-elif-else statement is used to execute different blocks of code depending on multiple conditions. The basic syntax of an if-elif-else statement is as follows:

if [ condition1 ]

then

# Code to be executed if condition1 is true

elif [ condition2 ]

then

# Code to be executed if condition2 is true

else

# Code to be executed if both condition1 and condition2 are false

Fi

**Note:**

**Strings Numbers**

|  |  |  |
| --- | --- | --- |
| **String** | **Numbers** | **Meaning** |
| == | -eq | Equal |
| != | -ne | not equal |
| > | -gt | greater than |
| >= | -ge | greater than or equal |
| < | -lt | less than |
| <= | -le | less than or equal |

**Example-5: Write a script to check whether the given number is equal to 5 or not**

#!/bin/bash

# Prompt the user to enter a number

read -p "Enter a number: " num

# Check if the number is equal to 5

if [ $num -eq 5 ]

then

echo "The number is equal to 5"

else

echo "The number is not equal to 5"

fi

**Example-6: write a script to find the biggest of two numbers.**

#!/bin/bash

# Prompt the user to enter two numbers

read -p "Enter the first number: " num1

read -p "Enter the second number: " num2

# Check which number is bigger

if [ $num1 -gt $num2 ]

then

echo "The first number is bigger"

elif [ $num2 -gt $num1 ]

then

echo "The second number is bigger"

else

echo "The two numbers are equal"

fi

**Example-6: write a script to find the biggest of 3 numbers.**

#!/bin/bash

# Prompt the user to enter three numbers

read -p "Enter the first number: " num1

read -p "Enter the second number: " num2

read -p "Enter the third number: " num3

# Check which number is bigger

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]

then

echo "The first number is bigger"

elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]

then

echo "The second number is bigger"

elif [ $num3 -gt $num1 ] && [ $num3 -gt $num2 ]

then

echo "The third number is bigger"

else

echo "Two or more numbers are equal"

fi

**Example-7: write a script to check if the given string is a file or a directory or a link or if it doesn’t exist.**

#!/bin/bash

# Prompt the user to enter a file or directory path

read -p "Enter a file or directory path: " path

# Check if the path exists

if [ ! -e $path ]

then

echo "The path doesn't exist"

else

# Check if the path is a file

if [ -f $path ]

then

echo "The path is a file"

# Check if the path is a directory

elif [ -d $path ]

then

echo "The path is a directory"

# Check if the path is a symbolic link

elif [ -L $path ]

then

echo "The path is a symbolic link"

else

echo "The path is not a file, directory or symbolic link"

fi

fi

**Note: if the file/directory/link exists in sub-directories we need to give the path before the string name**

./search.sh

Enter the string name

Release/1.txt

**Factorial:**

4=4\*3\*2\*1 and 5=5\*4\*3\*2\*1

4 = `expr 4 \\* 3 \\* 2 \\*1`

5 = `expr 1 \\* 5 \\* 4 \\* 3 \\* 2 \\* 1`

looping statement:

type 1 :

while [ condition ]

do

echo "any statement"

done

**Example-7: Write a script to find the factorial of a given number**

#!/bin/bash

echo "enter the number to find the factorial"

read num

result=1

while [ $num -gt 0 ]

do

result=`expr $num \\* $result`

num= `expr $num - 1`

done

echo " the factorial of ga iven number is $result"

**Example-7: Write a script to find the sum of the first n numbers**

#!/bin/bash

# take input from the user

echo "Enter the value of n:"

read n

# initialize sum variable to 0

sum=0

# loop through the first n natural numbers and add them to the sum variable

for ((i=1;i<=n;i++))

do

sum=$((sum+i))

done

# display the sum

echo "The sum of the first $n natural numbers is: $sum"

**Example-8: Write a script to find the number of characters in each line of a file**

#!/bin/bash

# take input file name fthe rom user

echo "Enter file name: "

read file\_name

# checkthe if file exists

if [ ! -f "$file\_name" ]; then

echo "File not found!"

exit 1

fi

# loop through each line in the file and count the characters

while reading -r line; do

count=$(echo "$line" | wc -c)

echo "Line $((++i)): $count characters"

done < "$file\_name"

**Example-8: Write a script to display the person having age greater than 60.**

#!/bin/bas

# Prompt user for a filename

echo "Enter filename: "

read filename

# Read the file and loop through each line

while reading the line; do

name=$(echo $line | cut -d',' -f1) # Extract name from line

age=$(echo $line | cut -d',' -f2) # Extract age from line

# Check if age is greater than 60

if [ $age -gt 60 ]; then

echo $name

fi

done < $filename

**Example-9: Write a script to display the content of the file in reverse.**

#!/bin/bash

if [ $# -ne 1 ]; then

echo "Usage: $0 FILENAME"

exit 1

fi

filename=$1

if [ ! -f $filename ]; then

echo "$filename does not exist"

exit 1

fi

tac $filename

**Example-9:** **Write a script to display file names if they contain the pattern and display the respective message whether the file contains a pattern or not.**

#!/bin/bash

echo "Enter file pattern to search for:"

read pattern

for file in \*

do

if grep -q "$pattern" "$file"; then

echo "$file contains the pattern."

else

echo "$file does not contain the pattern."

fi

done

**Example-10: Write a shell script to rename all text files into html files.**

#!/bin/bash

for file in \*.txt

do

mv "$file" "${file%.txt}.html"

Done

**For loop**

**Syntax**

For i in value1 value2, ……

Do

Statement

Done

**Example-11: Write a script to display the values**

#!/bin/bash

# Define some variables

name="John"

age=30

city="New York"

# Print the values

echo "Name: $name"

echo "Age: $age"

echo "City: $city"

**Example-12:** **Write a script to find the sum of elements in an array**

#!/bin/bash

# Declare the array

arr=(1 2 3 4 5)

# Set a variable to store the sum

sum=0

# Loop through the array and add each element to the sum

for i in "${arr[@]}"

do

sum=$((sum + i))

done

# Print the sum

echo "The sum of the array elements is: $sum"

**Example-13: Write a script to find factorial for a given set of numbers.**

#!/bin/bash

# Prompt the user for input

echo "Enter a number to find its factorial:"

read num

# Set a variable to store the factorial

factorial=1

# Loop through the numbers from 1 to the input number and multiply each number with factorial

for (( i=1; i<=num; i++ ))

do

factorial=$((factorial \* i))

done

# Print the factorial

echo "The factorial of $num is: $factorial"

**Example-14:** **Write a script to monitor the usage of the server memory, if the server memory reaches its threshold value (70%). It will send an email to the concerned person.**

#!/bin/bash

# Set the threshold value for memory usage

threshold=70

# Get the current memory usage percentage

memory\_usage=$(free | awk '/Mem/{printf("%.2f"), $3/$2\*100}')

# Check if memory usage is above the threshold

if (( $(echo "$memory\_usage > $threshold" | bc -l) )); then

# Send an email to the concerned person

echo "Warning: Memory usage is above $threshold% on the server." | mail -s "Memory usage warning" concerned\_person@example.com

fi

**have you written any scripts?**

Yes, I have written many scripts and some of them are.

1. Written a script to monitor the usage of the server memory, if the server memory reaches its threshold value (70%). It will send an email to the concerned person.
2. Written a script to monitor the services, if the service stops automatically it has to send an email notification to the concerned person.
3. written a script to clean up the old builds (we need to retain new (latest 10) builds and delete old builds)

**Example-15: Write a script to monitor the services, if the services stop automatically it has to send a mail notification to the concerned team.**

#!/bin/bash

# Define the service to monitor

service\_name="my\_service"

# Define the email notification settings

to="user@example.com"

from="monitoring@example.com"

subject="Service $service\_name stopped"

body="The $service\_name service has stopped. Please take action."

# Monitor the service indefinitely

while true; do

# Check if the service is running

if systemctl is-active --quiet $service\_name; then

echo "$service\_name is running"

else

# Service has stopped, send email notification

echo "$service\_name has stopped, sending notification"

echo "$body" | mail -s "$subject" -r "$from" "$to"

fi

sleep 10 # Wait 10 seconds before checking again

done

**Example-15: Write a script to clean up the old builds (files), we want to retain 10 builds and delete all old builds**

#!/bin/bash

# Define the directory where the builds are stored

build\_dir="/path/to/builds"

# Define the maximum number of builds to retain

max\_builds=10

# Get a list of all the builds in the directory, sorted by modification time

builds=($(ls -t $build\_dir))

# Count the number of builds

num\_builds=${#builds[@]}

# Delete old builds

if [ $num\_builds -gt $max\_builds ]; then

num\_to\_delete=$((num\_builds - max\_builds))

for (( i=0; i<$num\_to\_delete; i++ )); do

build\_to\_delete="${builds[$i]}"

echo "Deleting old build: $build\_to\_delete"

rm -r "$build\_dir/$build\_to\_delete"

done

else

echo "No old builds to delete"

Fi

**Example-15: write a shell script Switch case.**

#!/bin/bash

# Prompt the user for input

echo "Enter a number between 1 and 3:"

read num

# Switch case statement to handle user input

case $num in

1)

echo "You entered 1."

;;

2)

echo "You entered 2."

;;

3)

echo "You entered 3."

;;

\*)

echo "Invalid input. Please enter a number between 1 and 3."

;;

esac

**Example-16: Write a script to do the following things.**

If the day is Mon, we need to create two files

If the day is tue add content to the above file and create one folder “temp” : echo >> tues | mkdir -p folder

Wed moves these files to the temp folder: mv

Thu takes the backup: cp

Fri removes the files which we have created on Monday

Sat and Sunday displayed as holiday

#!/bin/bash

# Get the current day of the week (1=Monday, 2=Tuesday, etc.)

day=$(date +%u)

# Perform different actions based on the day of the week

case $day in

1) # Monday - create two files

touch file1.txt

touch file2.txt

echo "Created files: file1.txt and file2.txt"

;;

2) # Tuesday - add content to files and create a folder

echo "Some content" >> file1.txt

echo "Some more content" >> file2.txt

mkdir -p temp

echo "Added content to files and created folder: temp"

;;

3) # Wednesday - move files to the temp folder

mv file1.txt file2.txt temp/

echo "Moved files to temp folder"

;;

4) # Thursday - take a backup of the files

cp -r temp backup/

echo "Backed up files to backup folder"

;;

5) # Friday - remove files created on Monday

rm file1.txt file2.txt

echo "Removed files created on Monday"

;;

6|7) # Saturday or Sunday - display holiday message

echo "Weekend - enjoy the holiday!"

;;

esac

**Script Automation:**

Crontab – it is a scheduler used to schedule scripts in Linux.

Crontab -e - it is used to edit the schedule of the scripts.

Crontab -l - it is used to list the scheduled scripts/jobs.

How to schedule the scripts:

\* \* \* \* \* path of the file

1st indicates minutes.

2nd indicates hours.

3rd indicates the date.

4th indicates the month.

5th indicates day of the week.

Ex: I want to run the scripts on 4th Sept 2021 at 9 am

9.05

05: it will execute exact 5min

\*/05: it will execute every 5min

\*: it will execute for every min

**Example-17: Write a script if we**

press 1 it will create 2 files.

Press 2 to change the permission of a file.

Press 3 to search for a pattern in a file.

Press 4 to find/list the files in the current directory recursively.

#!/bin/bash

# Display a menu and prompt for user input

echo "Select an option:"

echo "1. Create two files"

echo "2. Change the permission of a file"

echo "3. Search a pattern in a file"

echo "4. Find/list files in current directory recursively"

read choice

# Perform different actions based on user input

case $choice in

1) # Create two files

touch file1.txt file2.txt

echo "Created files: file1.txt and file2.txt"

;;

2) # Change the permission of a file

echo "Enter the name of the file:"

read filename

echo "Enter the new permission (e.g., 644):"

read permission

chmod $permission $filename

echo "Changed permission of file: $filename"

;;

3) # Search a pattern in a file

echo "Enter the name of the file:"

read filename

echo "Enter the search pattern:"

read pattern

grep $pattern $filename

;;

4) # Find/list files in current directory recursively

echo "Listing files in current directory recursively:"

find . -type f

;;

\*) # Invalid input

echo "Invalid input - please enter 1, 2, 3, or 4"

;;

esac