Linux Lab

## 3b.Create a shell script to find a file with particular name, (show separate outputs for both the conditions)

1. **if that file exists then rename the existing file and create an empty file with that name.**

## if that file does not exist then create a new empty file. (iii)If both the conditions done together.

#!/bin/bash

# Set the desired file name file\_name="example.txt"

# Check if the file exists

if [ -e "$file\_name" ] then

# If the file exists, rename it and create an empty file with the original name echo "File '$file\_name' exists."

# Rename the existing file

mv "$file\_name" "${file\_name}\_old"

echo "Existing file renamed to '${file\_name}\_old'."

# Create an empty file with the original name touch "$file\_name"

echo "New empty file created with the name '$file\_name'."

echo "Both conditions done together." else

# If the file does not exist, create a new empty file touch "$file\_name"

echo "File '$file\_name' does not exist. New empty file created."

fi

# Additional message to indicate script completion echo "Script execution completed."

## Explanation:

* #!/bin/bash: This is the shebang line, specifying that the script should be interpreted using the Bash shell.
* file\_name="example.txt": This line sets the variable file\_name to the value "example.txt". This is the name of the file that the script will be working with.
* if [ -e "$file\_name" ]; then: This line checks if the file specified by file\_name exists using the -e test in the if statement.
* echo "File '$file\_name' exists.": If the file exists, this line prints a message indicating that the file exists.
* mv "$file\_name" "${file\_name}\_old": This line renames the existing file by using the mv command. It appends "\_old" to the original file name.
* echo "Existing file renamed to '${file\_name}\_old'.": This line prints a message indicating that the existing file has been renamed.
* touch "$file\_name": This line creates a new empty file with the original name specified by file\_name using the touch command.
* echo "New empty file created with the name '$file\_name'.": This line prints a message indicating that a new empty file has been created with the original name.
* echo "Both conditions done together.": This line prints a message indicating that both conditions (existing file and non-existing file) have been handled together.
* else: This part of the script is executed if the file specified by file\_name does not exist.
* touch "$file\_name": This line creates a new empty file with the name specified by file\_name.
* echo "File '$file\_name' does not exist. New empty file created.": This line prints a message indicating that the file did not exist, and a new empty file has been created.
* fi: This marks the end of the if statement.
* echo "Script execution completed.": This line prints a message indicating that the entire script execution has been completed.

Output:

exam@ThinkCentre-M70t:~$ vi prog.sh exam@ThinkCentre-M70t:~$ sh prog.sh

File 'example.txt' does not exist. New empty file created. Script execution completed.

exam@ThinkCentre-M70t:~$ sh prog.sh

File 'example.txt' exists.

Existing file renamed to 'example.txt\_old'.

New empty file created with the name 'example.txt'. Both conditions done together.

Script execution completed.

# 3c. Set up a cron job for the above developed scripts, that will be execute after every 30 minutes

Give the command crontab –e Press enter

\*\* \* \* \* bin/bash sh 3b.sh

Check the output file created for 1 min gap by deleting already created file. Or

\* \* \* \* \* /path/to/3b.sh

# 3d). Illustrating shell variables in a shell script

#!/bin/bash

# Define variables name="John" age=25

city="ExampleCity"

# Print values of variables echo "Name: $name" echo "Age: $age"

echo "City: $city"

# Concatenate variables

greeting="Hello, my name is $name. I am $age years old, and I live in $city." echo "$greeting"

Explanation:

* #!/bin/bash: This line is called the shebang. It tells the system to execute the script using the Bash shell.
* name="John", age=25, city="ExampleCity": These lines declare three variables (name, age, and city) and assign values to them. In this example, the name is set to "John," age to 25, and city to "ExampleCity."
* echo "Name: $name", echo "Age: $age", echo "City: $city": These lines use the echo command to print the values of the variables. The $name, $age, and $city are variable interpolations, and they get replaced with the values of the corresponding variables when the script is executed.
* greeting="Hello, my name is $name. I am $age years old, and I live in $city.": This line creates a new variable called greeting by concatenating the values of the name, age, and city variables using string interpolation. The resulting string is a greeting message.
* echo "$greeting": This line prints the value of the greeting variable, which contains the concatenated message.

To give execution Permission to your script give:

chmod +x example.sh

Then You can Run The script

./example.sh

**4.a)** Build a shell script to display the system space used. If it is greater than 80%, display as Low system Space and list the files having size greater than 1GB. Set up a cron job for the above developed script to execute every Monday morning 10AM

#!/bin/bash

# Get system space usage percentage

space\_usage=$(df -h / | awk 'NR==2 {print $5}' | sed 's/%//') echo "System Space Usage: $space\_usage%"

# Check if system space usage is greater than 80%

if [ "$space\_usage" -gt 80 ]; then echo "Low System Space"

# List files greater than 1GB echo "Files larger than 1GB:"

find / -type f -size +1G -exec ls -lh {} \; 2>/dev/null fi

## Explanation:

* df -h / | awk 'NR==2 {print $5}' | sed 's/%//': This command retrieves the percentage of disk space used on the root filesystem.
* if [ "$space\_usage" -gt 80 ]; then: This conditional statement checks if the system space usage is greater than 80%.
* find / -type f -size +1G -exec ls -lh {} \; 2>/dev/null: This command finds files larger than 1GB and lists them using ls -lh. The 2>/dev/null part suppresses error messages due to permission issues.

4b).Write a shell program to count number of words, characters, white spaces, and special symbols in each text and display the output on standard output. Set a cronjob to execute above script every 3rd day of week morning 9 AM.

#!/bin/bash

# Function to count characters, words, white spaces, and special symbols count\_text\_stats() {

input\_text="$1"

char\_count=$(echo -n "$input\_text" | wc -m) word\_count=$(echo "$input\_text" | wc -w) space\_count=$(echo "$input\_text" | tr -cd ' ' | wc -c) special\_count=$(echo "$input\_text" | tr -cd '[:punct:]' | wc -c)

echo "Text Analysis Report:" echo " "

echo "Character count: $char\_count" echo "Word count: $word\_count"

echo "White space count: $space\_count" echo "Special symbol count: $special\_count" echo " "

}

# Example text (you can replace this with your own text or input)

text\_to\_analyze="This is an example text! It contains special symbols, such as @ and #."

# Call the function to analyze the text count\_text\_stats "$text\_to\_analyze"

# Additional message to indicate script completion echo "Script execution completed."

Explanation:

* count\_text\_stats function takes an input text, uses various commands (echo, wc, tr) to count characters, words, white spaces, and special symbols, and then prints the analysis report.
* text\_to\_analyze is an example text. You can replace it with your own text or modify the script to take input from a file or user.
* The cron job line is added to the end of the script:

Command for Execution-crontab: 0 9 \* \* 3 /path/to/text\_analysis.sh

Replace

**/path/to/**

with the actual path where your

script is located.

To set up the cron job, you can run:

**text\_analysis.sh**

## Command for Execution-crontab

>crontab –e

Add the cron job line, save, and exit the editor. This cron job will execute the script every 3rd day of the week (Wednesday) at 9 AM.