1. Create a script that, given a user name, finds the home directory of the user using the /etc/passwd file

#!/bin/sh

# Check the number of arguments and retrieve user info from /etc/passwd

If [ $# -eq 0 ]; then

Echo “Please pass a username as an argument.”

Elif [ $# -eq 1 ]; then

Grep “$1” /etc/passwd | cut -d “:” -f 1,6

Else

Echo “This script accepts only one argument.”

Fi

Vi script1.sh

Sh script1.sh # No arguments

Output: Please pass a username as an argument.

Sh script1.sh user1 user2 # More than one argument

Output: This script accepts only one argument.

Sh script1.sh labex # Valid argument

Output (example if labex exists): labex:/home/labex

Sh script1.sh non\_existing\_user # Invalid argument

Output: (No output, as the user does not exist.)

1. Write a script that creates a file out of the /etc/passwd file.

#!/bin/sh

# new.sh – Script to extract user information and format it

If [ $# -eq 1 ]; then

Echo “username uses Id group Id Home directory” > result2

Echo “-------------------------------------” >> result2

Cat /etc/passwd | cut -f1 -d: > a

Cat /etc/passwd | cut -f3 -d: > b

Cat /etc/passwd | cut -f4 -d: > c

Cat /etc/passwd | cut -f6 -d: > d

Paste b c d > result4

Cat result2 result4 > “$1”

Else

Echo “This script accepts only one argument”

Fi

Vi new.sh

Sh new.sh # No arguments

Output: This script accepts only one argument

Sh new.sh arg1 arg2 # Two arguments

Output: This script accepts only one argument

Sh new.sh output.txt # One argument

Expected Output: A file named output.txt will be created

Cat output.txt # Check the Output File

Output Example:

Username uses Id group Id Home directory

User1 1001 1001 /home/user1

User2 1002 1002 /home/user2

…

1. Write a script to change every single-line comment in a C++ source file that uses C program start and end comment tokens to a single-line comment starting with a C++ single-line token.

#!/bin/sh

# script.sh – Script to remove comments from a C file

If [ $# -eq 1 ]; then

Sed ‘s+//.\*++g’ “$1” | sed ‘s+/\\\*.\*\\\*/++g’ > output

Else

Echo “Script does not accept more than one argument”

Fi

Cat > cfile.c

/\* Comment section \*/

// Testing purpose

Int main() {

Int a; // declaring a variable

Int b; /\* closing \*/

Return 0;

}

Ctrl + D

Vi script.sh

Sh script.sh # No arguments

Output: Script does not accept more than one argument

Sh script.sh arg1 arg2 # Two arguments

Output: Script does not accept more than one argument

Sh script.sh cfile.c # One argument

Cat output

Output Example:

Int main() {

Int a;

Int b;

Return 0;

}

1. Create a script that finds each line in a file that contains a specified string.

#!/bin/sh

# script4.sh – Script to search for a string in a file

If [ $# -eq 3 ]; then

If [ -s “$2” ]; then

Grep -n “$3” “$2”

Else

Echo “The file does not exist or it is empty.”

Fi

Else

Echo “This program requires 3 arguments: 1. String to be matched 2. Filename”

Fi

Cat > file1.txt

Hi! Good Morning! My self Narasimha Veddy.

This is a sample program!

Let’s search for a string.

Ctrl + D

Vi script4.sh

Sh script4.sh # No arguments

Output: This program requires 3 arguments: 1. String to be matched 2. Filename

Sh script4.sh “Good” “file1.txt” # Incorrect number of arguments

Output: This program requires 3 arguments: 1. String to be matched 2. Filename

Sh script4.sh “Good” “file1.txt” “some\_extra\_argument” # Correct arguments

Output:

1: Hi! Good Morning! My self Narasimha Veddy.

Summary of the Output:

If the second argument is not a valid, non-empty file:

The file does not exist or it is empty.

If the file exists and contains the string:

Line Number: Line Contents

1. Write a script to backup and archive a list of files.

#!/bin/sh

# backup.sh – Script to back up a specified file to a directory

If [ $# -eq 2 ]; then

If [ -f “$1” ]; then

If [ -d “$2” ]; then

For filename in “$1”; do

Cp “$filename” “$2”

Done

Echo “Copy Done!”

Else

Echo “The directory does not exist.”

Fi

Else

Echo “The file does not exist or it is empty.”

Fi

Else

Echo “This script accepts only 2 arguments: 1. Filename 2. Backup directory name.”

Fi

Vi backup.sh

Echo “This is a test file for backup.” > testfile.txt

Mkdir /home/username/backup # Replace username

Sh backup.sh # Incorrect number of arguments

Output: This script accepts only 2 arguments: 1. Filename 2. Backup directory name.

Sh backup.sh “nonexistentfile.txt” “/home/username/backup” # Non-existing file

Output: The file does not exist or it is empty.

Sh backup.sh “testfile.txt” “/home/username/nonexistentbackup” # Non-existing directory

Output: The directory does not exist.

Sh backup.sh “testfile.txt” “/home/username/backup” # Correct arguments

Output:

Copy Done!

Summary of the Output:

If the number of arguments is not 2:

This script accepts only 2 arguments: 1. Filename 2. Backup directory name.

If the specified file does not exist or is empty:

The file does not exist or it is empty.

If the specified directory does not exist:

The directory does not exist.

If the file is successfully copied to the backup directory:

Copy Done!

1. Write a script that finds all soft links to a specific file.

#!/bin/sh

# check\_links.sh – Script to check for soft links pointing to a specified file

If [ $# -eq 1 ]; then

If [ -f “$1” ]; then

Echo “Soft links are:”

Ls -l | grep “->” | grep “$1” | cut -d “ “ -f9

Else

Echo “File does not exist.”

Fi

Else

Echo “This script accepts only one argument: the filename.”

Fi

Vi check\_links.sh

Echo “This is a test file.” > testfile.txt

Ln -s testfile.txt softlink\_to\_testfile

Sh check\_links.sh # Incorrect number of arguments

Output: This script accepts only one argument: the filename.

Sh check\_links.sh “nonexistentfile.txt” # Non-existing file

Output: File does not exist.

Sh check\_links.sh “testfile.txt” # Correct argument

Output:

Soft links are:

Softlink\_to\_testfile

Summary of the Output:

If the number of arguments is not 1:

This script accepts only one argument: the filename.

If the specified file does not exist:

File does not exist.

If there are soft links pointing to the specified file:

Soft links are:

Softlink\_to\_testfile

If no soft links point to the file:

No output after “Soft links are:”

1. Create a script that simulates the ls -l command but prints only three columns of our choice.

#!/bin/sh

# process\_columns.sh – Script to process columns based on user input

If [ $# -eq 3 ]; then

If [ “$1” -ge 1 ] && [ “$1” -le 14 ] && [ “$2” -ge 1 ] && [ “$2” -le 10 ] && [ “$3” -ge 1 ] && [ “$3” -le 10 ]; then

Awk “{print \$$1, \$$2, \$$3}” input\_file > 15out

Awk ‘{print $1, $2}’ input\_file > out2

Awk ‘{print $3}’ input\_file > out3

Paste out2 out3 | expand -t 50

Else

Echo “Enter numeric arguments and the argument values should be within the specified ranges.”

Fi

Else

Echo “Enter exactly three numeric arguments.”

Fi

Vi process\_columns.sh

Cat > input\_file <<EOL

1 2 3

4 5 6

7 8 9

10 11 12

EOL

Sh process\_columns.sh 1 2 # Incorrect number of arguments

Output: Enter exactly three numeric arguments.

Sh process\_columns.sh 15 2 3 # Out of range arguments

Output: Enter numeric arguments and the argument values should be within the specified ranges.

Sh process\_columns.sh 1 2 3 # Valid arguments

Output: (No output in the terminal; check the output files.)

Cat 15out

Output:

1 2 3

4 5 6

7 8 9

10 11 12

Cat out2

Output:

1 2

4 5

7 8

10 11

Cat out3

Output:

3

6

9

12

Summary of the Output:

If the number of arguments is not 3:

Enter exactly three numeric arguments.

If the arguments are outside the specified ranges:

Enter numeric arguments and the argument values should be within the specified ranges.

If valid arguments are provided, it processes the input file and generates output files (15out, out2, and out3).

1. Create a script that sends contents of a message file to everybody who logged in.

#!/bin/sh

# send\_message.sh – Script to send a message to all users from a specified file

If [ $# -eq 1 ]; then

If [ -r “$1” ]; then

Welcome=`who | awk ‘{print $1}’ > users.txt`

Cat “$1” | wall -n

Echo “Successfully sent the message to all users.”

Else

Echo “The file is not readable.”

Fi

Else

Echo “This script accepts only one argument: a readable file name.”

Fi

Vi send\_message.sh

Cat > message.txt <<EOL

Hi everyone! This is a test message.

Please check your notifications.

EOL

Sh send\_message.sh # Incorrect number of arguments

Output: This script accepts only one argument: a readable file name.

Sh send\_message.sh non\_readable\_file.txt # Non-readable file

Output: The file is not readable.

Sh send\_message.sh message.txt # Valid file

Output: Successfully sent the message to all users.

Explanation of the Script:

The script checks if exactly one argument is provided ($# -eq 1).

It checks if the provided file is readable (-r “$1”).

If the checks pass, it uses the who command to get the list of logged-in users and sends the message from the specified file using wall.

The message is sent to all users currently logged in to the system.

Summary of the Output:

If the number of arguments is incorrect:

This script accepts only one argument: a readable file name.

If the file is not readable:

The file is not readable.

If a valid readable file is provided, it sends the message successfully:

Successfully sent the message to all users.

1. Create a script that can be executed only from a specific terminal.This is done for security purposes.

#!/bin/sh

# secarity.sh – A script to check the terminal and run another script if conditions are met

If [ $# -eq 0 ]; then

Check=$(tty) # Get the terminal type

Original=’/dev/pts/0’ # Specify the original terminal to check against

If [ “$check” = “$original” ]; then # Check if the current terminal matches the original

Sh sample.scr # Run the sample script

Else

Exit 1 # Exit if the terminal does not match

Fi

Else

Echo “This script does not accept any arguments”

Fi

Vi secarity.sh

Vi sample.scr

Chmod +x sample.scr

Sh secarity.sh # Run with no arguments

Output: Sample script executed. # If in the correct terminal

Sh secarity.sh arg1 # Run with an argument

Output: This script does not accept any arguments.

Sh secarity.sh # Test in a different terminal

Output: (Depends on terminal matching)

Explanation of the Script:

- The script first checks if there are no arguments provided ($# -eq 0).

- It retrieves the current terminal using the tty command and compares it to a predefined terminal (/dev/pts/0).

- If the terminals match, it executes sample.scr; otherwise, it exits.

- If any arguments are provided, it outputs a message indicating that arguments are not accepted.

Summary of the Output:

1. If run without arguments and the terminal matches:

Sample script executed.

1. If run with any arguments:

This script does not accept any arguments.

1. If run in a terminal that does not match:

(No output, script simply exits.)

1. Create a script that compiles all C source files in your home directory and create executable files

#!/bin/sh

# my\_script.sh – A script to execute output files

If [ $# -eq 0 ]; then

# If no arguments are provided, process files listed in ‘output’

For filename in $(cat output); do

./”$filename”.e # Execute each file with ‘.e’ extension

Done

Else

Echo “The program doesn’t accept any arguments”

Fi

Step-by-Step Commands:

1. Create the Script:

Vi my\_script.sh

1. Create an Output File:

Vi output

Add the names of the executable files:

Script1

Script2

1. Create Sample Executable Files:

Vi script1.sh

#!/bin/sh

Echo “Running script1”

Vi script2.sh

#!/bin/sh

Echo “Running script2”

1. Make the Executable Files Executable:

Chmod +x script1.sh script2.sh

Mv script1.sh script1.e

Mv script2.sh script2.e

1. Run the Script with No Arguments:

Sh my\_script.sh

Expected Output:

Running script1

Running script2

1. Run the Script with One Argument:

Sh my\_script.sh arg1

Expected Output:

The program doesn’t accept any arguments

Explanation of the Script:

- The script checks if no arguments are provided ($# -eq 0).

- If no arguments are provided, it reads filenames from the output file and attempts to execute each one as an executable (appending .e to each filename).

- If any arguments are provided, it outputs a message indicating that the program does not accept arguments.

Summary of the Output:

1. If run without arguments:

Running script1

Running script2

1. If run with any arguments:

The program doesn’t accept any arguments

11)

#!/bin/sh

*# duplicateName.scr - A script to find files with same filename in different directories*

*# Validation Section*

if [ $# -ne 0 ]; then

echo "This program doesn't accept any arguments"

exit 1

fi

*# Body Section*

find . -type f | awk -F '/' '{

# Get the filename (last field)

f=$NF

# Store the full path

a[f] = f in a ? a[f] " " $0 : $0

# Count occurrences

b[f]++

}

END {

# Print files that appear more than once

for (x in b) {

if (b[x] > 1) {

printf "Duplicate filename: %s\n", x

printf "%s\n", a[x]

}

}

}'

Step-by-Step Commands:

1. Create directories for testing:

bash

Copy

mkdir -p ~/a/aa/aaa ~/a/bb ~/b/bb/bbb ~/b/cc

1. Create test files:

bash

Copy

*# Create file1 in multiple locations*

touch ~/a/file1 ~/a/aa/aaa/file1 ~/b/bb/bbb/file1

*# Create file2 in multiple locations*

touch ~/a/aa/file2 ~/b/file2 ~/b/bb/file2

*# Create file3 in multiple locations*

touch ~/a/bb/file3 ~/b/cc/file3

*# Create unique files*

touch ~/a/unique1 ~/b/unique2 ~/b/bb/unique3

1. Make the script executable:

bash

Copy

chmod +x duplicateName.scr

1. Testing scenarios:

Test with no arguments:

bash

Copy

./duplicateName.scr

Expected Output:

Copy

Duplicate filename: file1

./file1

./a/aa/aaa/file1

./b/bb/bbb/file1

Duplicate filename: file2

./a/aa/file2

./b/file2

./b/bb/file2

Duplicate filename: file3

./a/bb/file3

./b/cc/file3

Test with arguments:

bash

Copy

./duplicateName.scr arg1

Expected Output:

Copy

This program doesn't accept any arguments

1. Verify results:

bash

Copy

ls -R ~