Linux Programming: Assignment-7: 25-09-2025

NAME:SANDEEP
USN:ENG24CY0047
ROLL NO:40
SECTION:B

1. **What is a bash shell script? Give one example. (CO4)**

A: A Bash shell script is an ASCII text file that has a collection of commands authored for the Bash (Bourne Again SHell) command-line interpreter. It enables you to automate tasks, run multiple commands in sequence, and build programs that can be run in a Linux/Unix system. In essence, it's a means of putting together commands and logic (such as loops and conditionals) in one file that can be run as a program.

Basic Example:

Create a file example.sh containing the following:

```
#!/bin/bash

# This is a simple bash script


echo "Hello, World!"
```

Explanation:

#!/bin/bash → Shebang line instructing the system to use Bash to interpret the script.

echo "Hello, World!" → Outputs "Hello, World!" to the terminal.

Run the script:

```
chmod +x example.sh   # Makes the script executable

./example.sh         # Runs the script
```

```
Hello, World!
```

1. **Write a simple shell script to print "Hello World". (CO4)**

A: Script: hello.sh

```
#!/bin/bash
```

```
# This script prints "Hello World"

echo "Hello World"
```

Steps to execute:

Name the file as hello.sh.

Make it executable:

```
chmod +x hello.sh
```

Run the script:

```
./hello.sh
```

Output:

```
Hello World
```

This is the most basic example of a Bash shell script.

1. **What is the purpose of comments (#) in a shell script? (CO4)**

A: In a shell script, the # is utilized to add comments.

Reason Behind Comments:

Clarify the Code: Comments provide an explanation of what a specific line or block of the script does, and it is simpler for humans to comprehend.

Enhance Readability: They make scripts easier to read, particularly by others or for future maintenance.

Temporarily Disable Code: Shell ignores lines beginning with #, so you can "comment out" commands for debugging or testing purposes without removing them.

Example:

```
#!/bin/bash


# This script prints Hello World

echo "Hello World"  # Outputs the message to the terminal
```

Lines beginning with # are ignored by Bash.

Only echo "Hello World" is run.

Comments are essential for documentation and maintenance in shell scripts.

1. **How do you declare variables (int, float, double, string, Boolean, and char in a shell script? (CO4)**

A: Variable declaration in Bash shell scripting is not the same as in languages such as C, C++, or Java. Bash is loosely typed, and hence variables don't come with a specific type such as int, float, or char. Rather, all variables are assumed strings by default, and arithmetic operations can be done using special syntax.

This is how you can conceptually declare and work with various types in a shell script:

1. Integer

```
#!/bin/bash

num=10      # integer assignment

echo "Integer: $num"
```

To perform arithmetic:

```
((num = num + 5))

echo "After addition: $num"
```

2. Float / Double

Bash does not natively support floats. Use bc or awk for floating-point operations.

```
#!/bin/bash
```

```
float=3.14

echo "Float value: $float"

# Arithmetic using bc

result=$(echo "$float + 2.5" | bc)

echo "After addition: $result"
```

3. String

```
#!/bin/bash

name="Alice"

echo "String: $name"
```

4. Boolean

Bash does not support a Boolean type. Use 0 / 1 or true / false (keywords):

```
#!/bin/bash

flag=true

if [ "$flag" = true ]; then

   echo "Boolean is true"

else

   echo "Boolean is false"

fi
```

5. Character

Bash does not have an independent char type. One character is simply a string of size 1:

```
#!/bin/bash

char="A"

echo "Character: $char"
```

1. **Write a shell script to display the current date and time of the system. (CO4)**

A: Script: datetime.sh

```
#!/bin/bash

# This script prints the current date and time


echo "Current date and time: $(date)"
```

Steps to execute:

Save the file as datetime.sh.

Make it executable:

```
chmod +x datetime.sh
```

Run the script:

```
./datetime.sh
```

Sample Output:

```
Current date and time: Tue Oct 7 14:30:25 IST 2025
```

Explanation:

date is a command in Linux/Unix that prints the current date and time.

$(date) runs the command and places its output within the echo statement.

This is a simple but very handy script for system monitoring or logging.

1. **Explain the difference between a constant and a variable in bash script. (CO4)**

A: 1. Variable

A variable is a named storage for data which may change while the script is running.

In Bash, variables are declared without the need to indicate a type and are able to store strings, numbers, etc.

Example:

```
#!/bin/bash
name="Alice"     # variable assignment
echo "Name: $name"

name="Bob"       # updating the value
echo "Updated Name: $name"
```

Output:

```
Name: Alice
Updated Name: Bob
```

2. Constant

A constant is a value that may not be altered after it has been assigned.

In Bash, a constant may be declared using the readonly or declare -r command.

Example:

```
#!/bin/bash
readonly PI=3.14
echo "Value of PI: $PI"
```

```
# Attempting to change the value will result in an error

# PI=3.14159   # This will produce an error
```

1. **Write a shell script to read two integer number from the user and compute the sum of both the number. (CO4)**

A: Script: sum.sh

```
#!/bin/bash

# Read two integers and calculate their sum


# Read first integer

read -p "Enter the first number: " num1


# Read second integer

read -p "Enter the second number: " num2


# Calculate sum

sum=$((num1 + num2))


# Print the result

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

Steps to run:

Save the file named sum.sh.

Make it executable:

```
chmod +x sum.sh
```

Run the script:

./sum.sh

Sample Run:

```
Enter the first number: 10

Enter the second number: 25

The addition of 10 and 25 is: 35
```

Explanation:

read -p is employed to prompt for the user input.

$((num1 + num2)) does arithmetic addition in Bash.

echo writes the output to the console.

This is a basic demonstration of arithmetic operation and user input in shell scripting.

1. **What is the use of source command in shell scripting? (CO4)**

A: In shell scripting, the source command is employed to run a script within the existing shell environment as opposed to initiating a new subshell. Any variables, functions, or modifications made by the script thus remain accessible in the existing shell after completion of the script.

Use / Purpose of source:

Load the environment variables from a file into the existing shell.

Run functions or scripts without entering a new shell.

Make the changes instantly in the current session (such as modifying PATH, aliases, etc.).

Syntax:

```
source filename
```

Or its shorthand:

```
. filename
```

Example:

Let's say we have a script variables.sh:

```
#!/bin/bash

# Set some variables
NAME="Alice"
AGE=25
```

If you execute it normally:

```
./variables.sh
echo $NAME
```

$NAME will be blank because it executed in a subshell.

If you execute with source:

```
source variables.sh
echo $NAME
```

Output:

```
Alice
```

The variable NAME is now set in the current shell.

1. **How can you debug a shell script? Give two methods. (CO4)**

A: Debugging a Shell Script

Debugging assists you in locating and correcting errors in your script. Two standard ways are:

1. Using -x option (Trace Execution)

Displays each command prior to execution.

Assists you in viewing what the script is executing.

Example:

```
bash -x script.sh
```

Or within the script:

```
#!/bin/bash
set -x   # begin debugging
echo "Hello"
set +x   # end debugging
```

2. Using -n option (Check Syntax)

Verifies the script for errors without running it.

Example:

```
bash -n script.sh
```

1. **Write a bash script to create and delete a file. (CO4)**

A: Script: file_ops.sh

```
#!/bin/bash
# This program creates and deletes a file

# Prompt for file name
```

```
read -p "Enter the file name: " filename

# Create the file
touch "$filename"
echo "File '$filename' created."

# Delete the file
rm "$filename"
echo "File '$filename' deleted."
```

Steps to execute:

Save the file as file_ops.sh.

Make it executable:

```
chmod +x file_ops.sh
```

Run the script:

```
./file_ops.sh
```

Sample Run:

```
Enter the file name: test.txt
File 'test.txt' created.
File 'test.txt' deleted.
```

Explanation:

touch → creates an empty file.

rm → removes the file.

read -p → reads input from the user.

It is a simple and easy method of file creating and deleting through a Bash script.