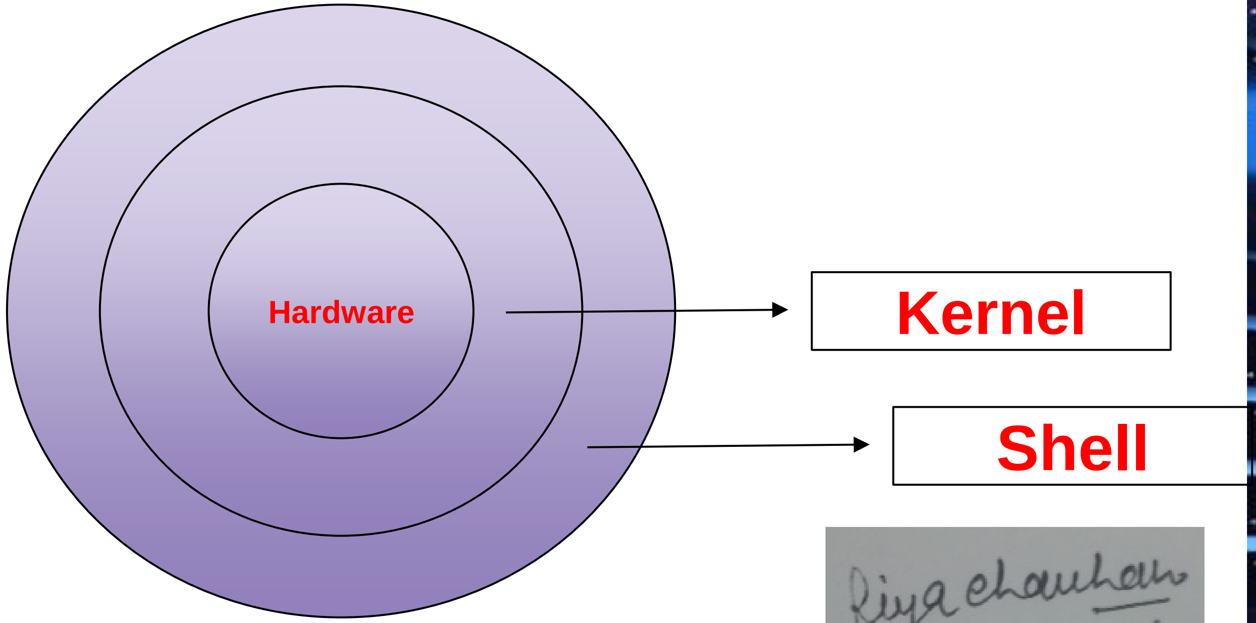
IMP Terminologies

- **Kernel**
  - The kernel is the heart of the operating system.
- **Shell**
  - The shell is the utility that processes your requests
- **Terminal**
  - Provides interface to a user so that he/she can access the shell
- **Shell Scripting**
  - A shell script is a text file containing a series of shell commands and constructs that are executed in sequence to avoid repetitive work.

Riya chauhan

What is Shell

- A Shell provides you with an interface to the Unix system
- It gathers input from user and executes programs based on that input.
- When a program finishes executing, it displays that program's output.
- There are different flavors of a shell
    - Bourne shell ($)
    - C shell (%)



Riya chauhan

Types of Shell

Bourne Shell

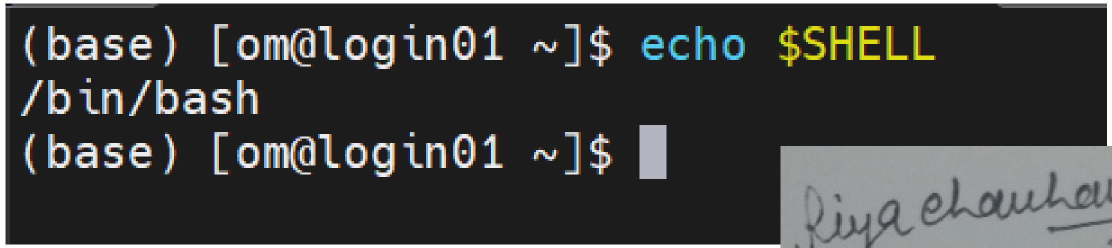
- Bourne shell (sh)
- Korn shell (ksh)
- Bourne Again shell (bash)
- POSIX shell (sh)

C Shell

- C Shell (csh)
- TENEX/TOPS C shell (tcsh)

❖ **How To Check Shell**

➢ **echo $SHELL**

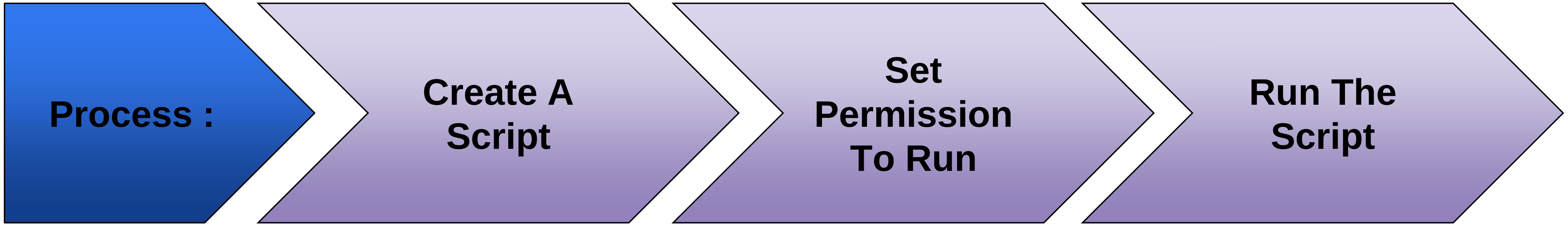


```
(base) [om@login01 ~]$ echo $SHELL
/bin/bash
(base) [om@login01 ~]$ 
```

Riya chauhan

Process of Bash Script

The basic concept of a shell script is a list of commands, which are listed in the order of execution.



- Create a Script using Text Editor
- Using chmod give executable permission to script

```
chmod +x test.sh
```

- Run the script

```
./test.sh
```

Riya chauhan

Shell Scripting

- The basic concept of a shell script is a list of commands, which are listed in the order of execution.
- Before you add anything else to your script, you need to alert the system that a shell script is being started using **shebang** construct

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

- Save the above content and make the script executable

```
chmod +x test.sh
```

- To execute program

```
./test.sh
```

Riya chauhan

Variables in Shell Scripting

- **Local Variables**
  - Variables present within current instance of shell
  - Not available to programs started by this shell
- **Environment Variables**
  - An environment variable is available to any child process of the shell
  - It can be set using the **export** command
- **Shell Variables**
  - Special variable that is set by the shell and is required by the shell in order to function properly.

Riya chauhan

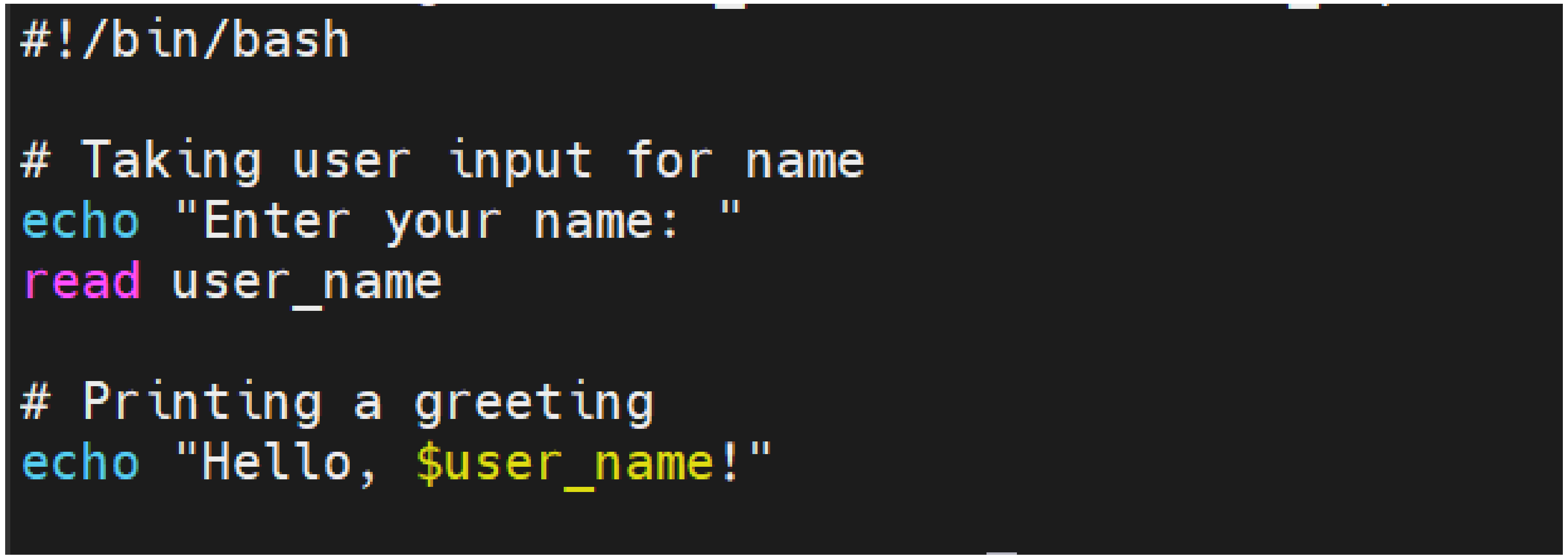
Rules for Variables

- **No white space either side of assignment operator (=)**
  - name = “ABC” ❌
- **Variable name can’t have special character**
  - @num=“ABC” ❌
- **The first character of the variable name cannot be a number**
  - 1_name=“ABC” ❌
- **Variable names cannot be reserved words**
  - If , else , case ❌

Riya chauhan

Standard Input and Output

- Standard Input
  - **read** command to take user input
- Standard Output
  - **echo** command to display output

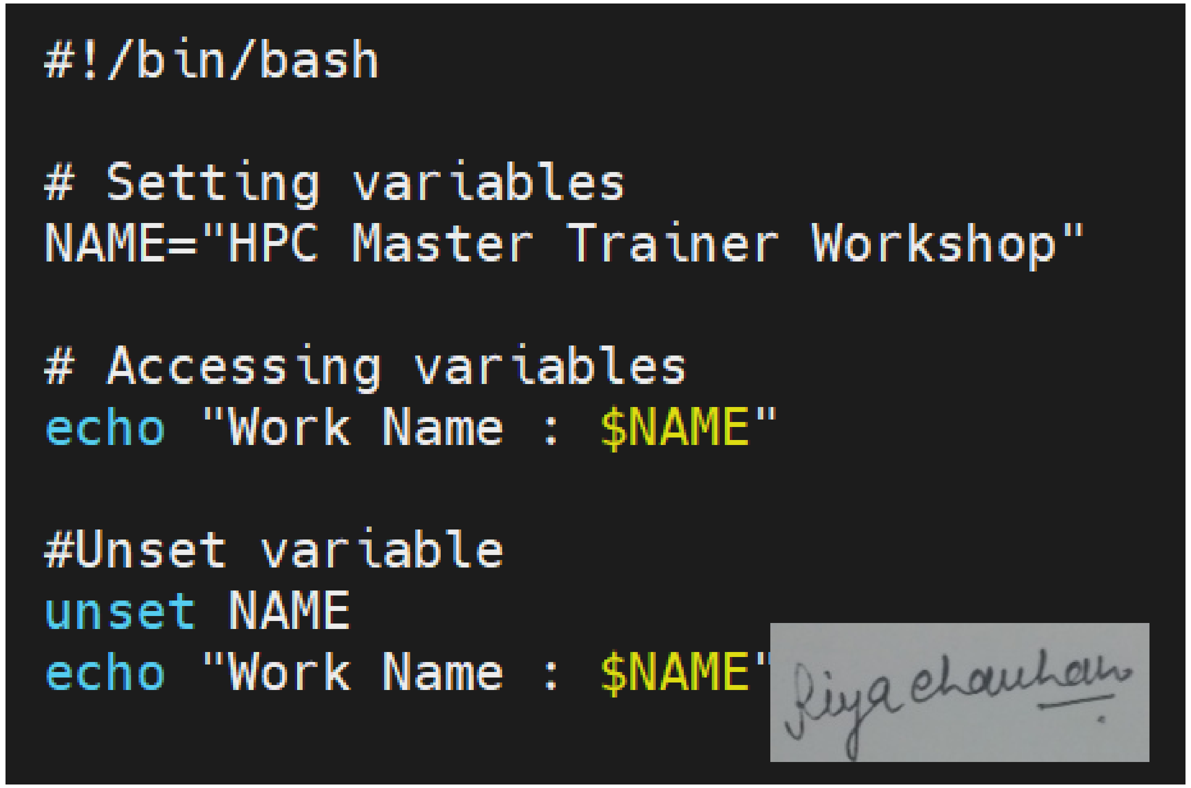


```
#!/bin/bash

# Taking user input for name
echo "Enter your name: "
read user_name

# Printing a greeting
echo "Hello, $user_name!"
```

[illegible]

Scalar Variables

- **Define Variable**
  - variable_name = <variable data>
  - Name=”UNIX WORLD 2024”
- **Accessing variables**
  - echo $NAME
- **Unsetting Variable**
  - unset $NAME



```bash
#!/bin/bash

# Setting variables
NAME="HPC Master Trainer Workshop"

# Accessing variables
echo "Work Name : $NAME"

#Unset variable
unset NAME
echo "Work Name : $NAME"
```

Riya chauhan

Array Variables

- A scalar variable is capable enough to hold a single value only.
- Array variable can hold multiple values at the same time.
- To set Array elements

```
array_name=(value1 value2 ... valueN)
```

- To access Array elements

```
${array_name[index]}
```

Riya chauhan

Shell Basic Operators

- Bash scripting supports various operators for performing different types of operations
    1) Arithmetic Operator
    2) Relational Operator
    3) Boolean Operator
    4) String Operator
    5) File Test Operator

Riya chauhan

Arithmetic Operators

$((value1 + value2))

Operator	Description
+	addition
-	subtraction
*	Multiplication
/	Division
%	Modulo
=	Assignment
==	Equality

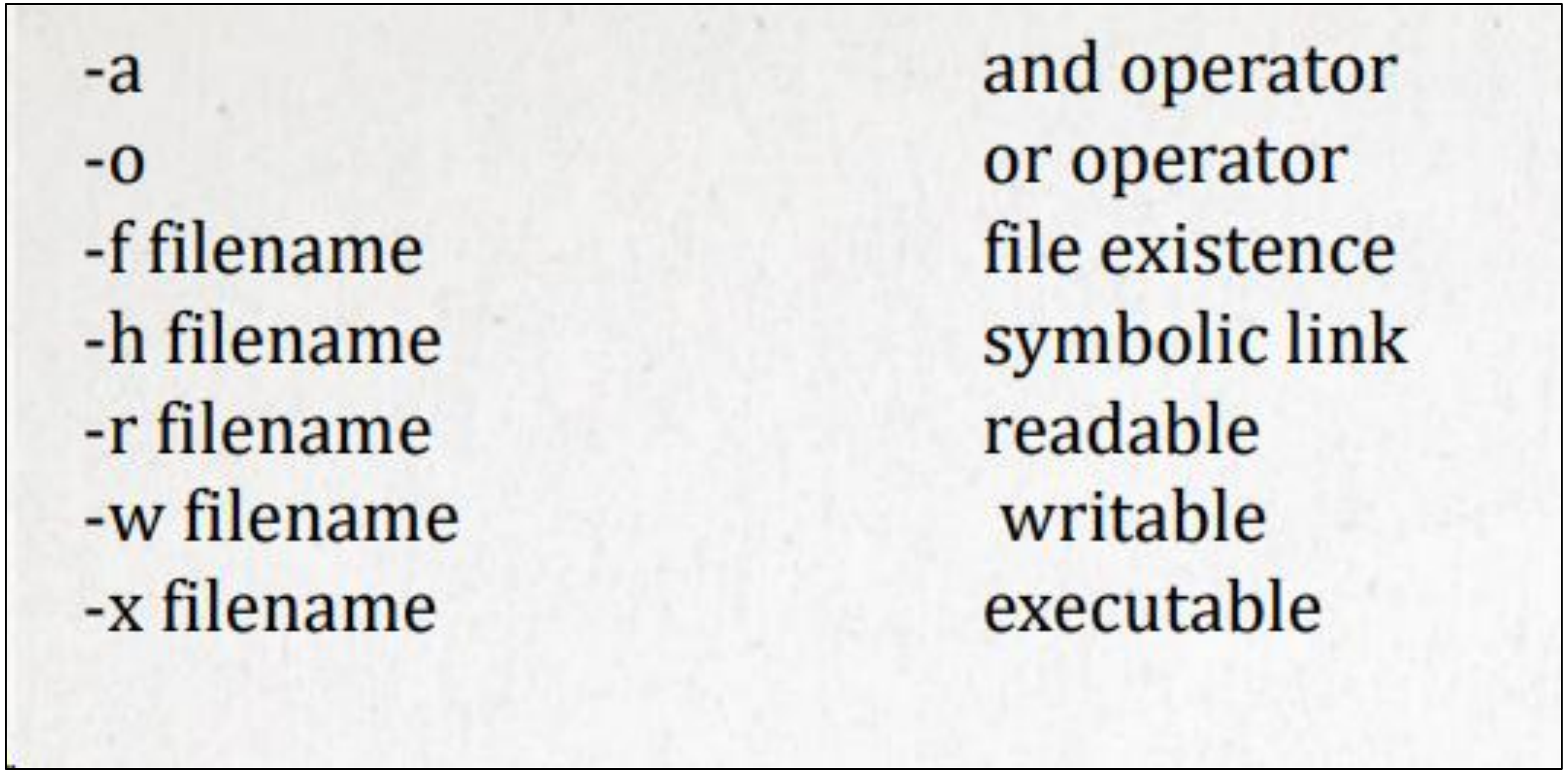
Riya chauhan

Relational Operators

Operator	Description
-eq	Checks if two values are equal
-ne	Checks if two values are not equal
-gt	checks if left operand greater than right operand
-lt	checks if left operand less than right operand
-ge	checks if left operand greater than equal to right operand
-le	checks if left operand less thanequal to right operand

Riya chauhan

File Test Operators



-a	and operator
-o	or operator
-f filename	file existence
-h filename	symbolic link
-r filename	readable
-w filename	writable
-x filename	executable

9/12/2024

Riya chauhan

Shell Decision Making

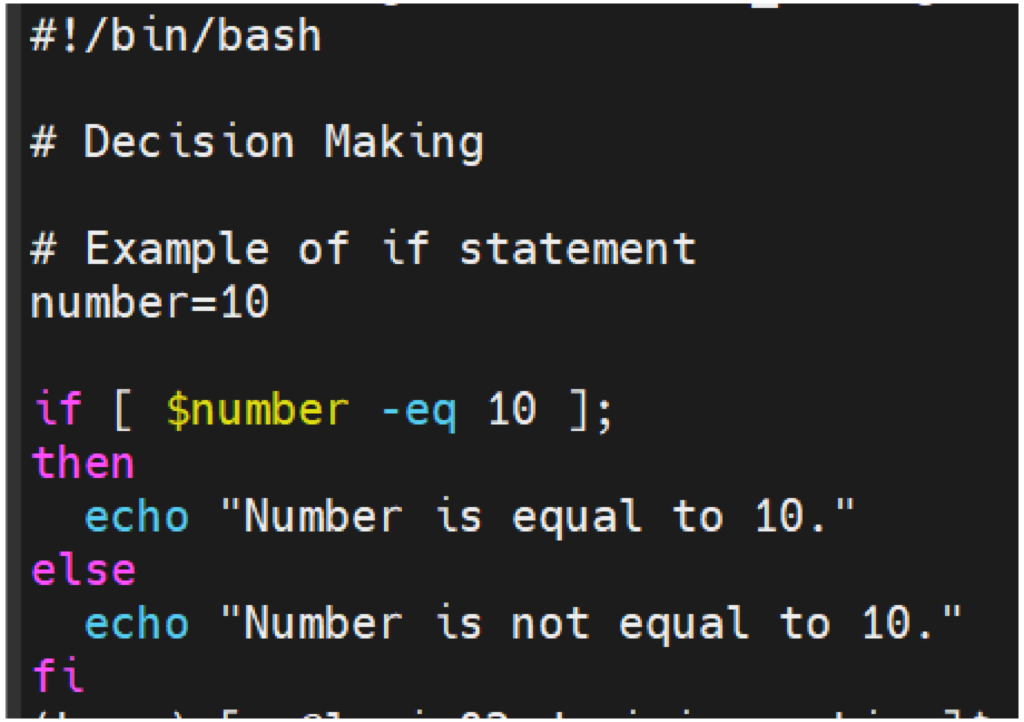
- Writing a shell script, there may be a situation when you need to adopt one path out of the given two paths.
- So you need to make use of conditional statements that allow your program to make correct decisions and perform the right actions

1. if statement
2. if …. else statement
3. if..elif..else..fi statement (Else If ladder)
4. if..then..else..if..then..fi..fi..(Nested if)
5. switch statement

Riya chauhan

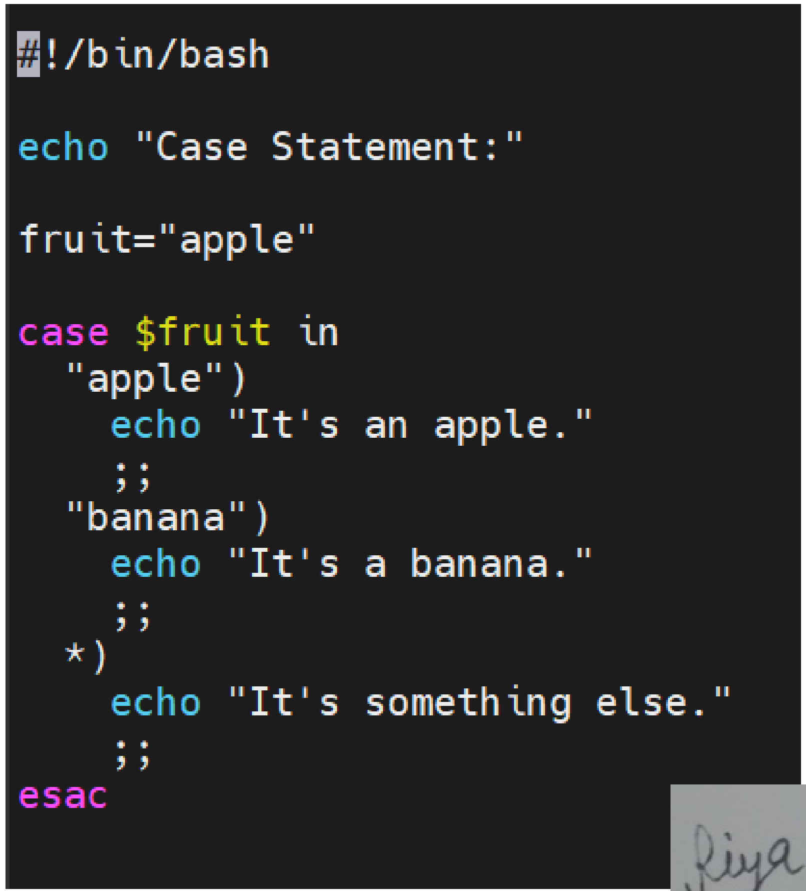
Decision Making Loops

- If … else Statement



```
#!/bin/bash

# Decision Making

# Example of if statement
number=10

if [ $number -eq 10 ];
then
  echo "Number is equal to 10."
else
  echo "Number is not equal to 10."
fi
```

- Switch Case statement



```
#!/bin/bash

echo "Case Statement:"

fruit="apple"

case $fruit in
  "apple")
    echo "It's an apple."
    ;;
  "banana")
    echo "It's a banana."
    ;;
  *)
    echo "It's something else."
    ;;
esac
```

Riya chauhan

Looping Statements

- There are total 3 looping statements that can be used in bash programming
  - while statement
  - For statement
  - until statement

- To alter the flow of loop statements
  - Break
  - Continue

Riya chauhan

While Loop Statements

- command is evaluated and based on the resulting loop will execute, if the command is raised to false then the loop will be terminated that.
- Syntax

```
#/bin/bash
while <condition>
do
        <command 1>
        <command 2>
done
```

[illegible]

For Loop Statements

- The for loop operates on lists of items.
- It repeats a set of commands for every item in a list
- Syntax

```
#/bin/bash
for <var> in <value1 value2 ... valuen>
do
                        <command 1>
                        <command 2>
done
```

Riya chauhan

Until Loop Statements

- The until loop is executed as many times as the condition/command evaluates to false.
- The loop terminates when the condition/command becomes true.
- Syntax

```
#/bin/bash
until <condition>
do
            <command 1>
            <command 2>
done
```

Riya chauhan

Redirection

❖ To redirect output

```
echo "Additional text" > output.txt
```

❖ To append output

```
echo "Additional text" >> output.txt
```

❖ To redirect standard error

```
command_that_might_fail 2 > error.log
```

❖ To redirect standard error and output

```
command_that_might_fail > output_and_error.log 2>&1
```

❖ To redirect to NULL device

```
command_to_silence > /dev/null
```

Riya chauhan

Functions

- functions are named blocks of code that perform a specific task.
- Its main goal is to break down a complicated procedure into simpler subroutines

Syntax :

```
function_name(){
                    // body of the
function
}
```

9/12/2024

Riya chauhan

Riya chauhan