- **Hello World Script**

```bash
#!/bin/bash
# This script prints "Hello, World!" to the terminal
echo "Hello, World!"
```

- **Variables and User Input**

```bash
#!/bin/bash
# This script takes user input and stores it in a variable
echo "Enter your name:"
read name
echo "Hello, $name!"
```

- **Conditionals (If-Else)**

```bash
#!/bin/bash
# This script checks if the number is positive or negative
echo "Enter a number:"
read number

if [ $number -gt 0 ]; then
 echo "$number is positive"
elif [ $number -lt 0 ]; then
 echo "$number is negative"
else
 echo "The number is zero"
fi
```

- **Loops (For Loop)**

  ```bash
  #!/bin/bash
  # This script prints numbers 1 to 5 using a for loop

  for i in {1..5}
  do
   echo "Number: $i"
  done
  ```

- **While Loop**

  ```bash
  #!/bin/bash
  # This script prints numbers 1 to 5 using a while loop

  counter=1
  while [ $counter -le 5 ]
  do
   echo "Number: $counter"
   ((counter++))
  done
  ```

- **Functions**

  ```bash
  #!/bin/bash
  # This script defines a function and calls it

  greet() {
   echo "Hello, $1!"
  }

  # Call the function with a parameter
  greet "Alice"
  ```

- **Array Usage**

  ```
  #!/bin/bash
  # This script demonstrates the use of arrays in bash

  # Declare an array
  fruits=("Apple" "Banana" "Cherry" "Date")

  # Print each fruit in the array
  for fruit in "${fruits[@]}"
  do
   echo "Fruit: $fruit"
  done
  ```

- **File Handling (Reading and Writing)**

  ```
  #!/bin/bash
  # This script reads from a file and writes to a new file

  input_file="input.txt"
  output_file="output.txt"

  # Read input file
  echo "Reading from $input_file:"
  cat $input_file

  # Write to output file
  echo "Writing to $output_file:"
  echo "This is a new line of text" > $output_file
  ```

- **Redirecting Output to a File**

  ```
  #!/bin/bash
  # This script demonstrates how to redirect output to a file

  echo "This message will be written to a file" > output.txt
  echo "Appending this message to the file" >> output.txt
  ```

- **Error Handling**

```
#!/bin/bash
# This script handles errors using exit status

echo "Starting the script"

# Example of a command that might fail
mkdir /root/new_folder

# Check if the command was successful
if [ $? -eq 0 ]; then
 echo "Folder created successfully!"
else
 echo "Error occurred: Folder creation failed"
 exit 1
fi
```

- **Using grep for Searching in Files**

```
#!/bin/bash
# This script demonstrates how to search for a pattern in a file
using grep

echo "Enter a search pattern:"
read pattern

echo "Searching for '$pattern' in file.txt:"
grep "$pattern" file.txt
```

- **Processing Command-Line Arguments**

```bash
#!/bin/bash
# This script processes command-line arguments

if [ $# -lt 2 ]; then
 echo "Usage: $0 <name> <age>"
 exit 1
fi

name=$1
age=$2

echo "Hello, $name! You are $age years old."
```

- **Cron Jobs (Scheduling Tasks)**

```bash
#!/bin/bash
# This script schedules a task using cron
# Open crontab for editing:
# crontab -e

# Example: Run this script every day at 7am
# 0 7 * * * /path/to/this/script.sh
```

- **Working with awk**

```bash
#!/bin/bash
# This script uses awk to process text from a file

echo "Displaying the second column from file.txt using awk:"
awk '{print $2}' file.txt
```

- **Process Management**
  ```
  #!/bin/bash
  # This script demonstrates how to manage processes

  # Display running processes
  echo "List of running processes:"
  ps aux

  # Kill a specific process (use with caution)
  echo "Enter PID of the process to kill:"
  read pid
  kill $pid
  ```

- **Network Commands (Ping & Port Check)**
  ```
  #!/bin/bash
  # This script pings a host and checks open ports

  echo "Enter the host to ping:"
  read host
  ping -c 4 $host

  echo "Enter port to check (e.g., 80):"
  read port
  nc -zv $host $port
  ```

- **Using sed for Text Manipulation**
  ```
  #!/bin/bash
  # This script uses sed to replace text in a file

  echo "Enter the text to search for:"
  read search
  echo "Enter the text to replace it with:"
  read replace

  sed -i "s/$search/$replace/g" file.txt
  echo "Text replaced successfully!"
  ```

- **Working with tar (Archiving Files)**

```
#!/bin/bash
# This script creates and extracts tar archives

echo "Creating a tar archive of the directory:"
tar -cvf archive.tar /path/to/directory

echo "Extracting the tar archive:"
tar -xvf archive.tar
```

- **Advanced Regular Expressions with grep**

```
#!/bin/bash
# This script demonstrates advanced regex usage with grep

echo "Searching for lines containing a number followed by a word:"
grep -P '\d+\w+' file.txt
```

- **Using trap to Catch Signals**

```
#!/bin/bash
# This script demonstrates the use of trap to catch signals

trap 'echo "You pressed CTRL+C"; exit' INT

while true
do
 echo "Running... Press CTRL+C to stop."
 sleep 2
done
```

How to Execute a Basic Shell Script

- **Create a Shell Script File:**
  - Open a text editor (e.g., `nano`, `vim`, or any GUI-based editor).
  - Write your script. For example:
    - **`echo "Hello, World!"`**
  - Save the file with a `.sh` extension, for example, `hello_world.sh`.
- **Make the Script Executable:**
  - Open a terminal and navigate to the directory where your shell script is saved.
  - Run the following command to make the script executable:
    - **`chmod +x hello_world.sh`**
  - This command gives the script permission to be executed.
- **Run the Script:**
  - To execute the script, run the following command in the terminal:
    - **`./hello_world.sh`**
  - This tells the terminal to execute the script from the current directory.

Created By **Prof. Abhishek Chauhan**