**Bash Scripting: A Comprehensive Overview**

**Introduction to Bash**

**What is Bash?**

* **Bash** is a command-line interpreter or Unix Shell, widely used in GNU/Linux Operating Systems.
* **Created By**: Brian Jhan Fox.
* **Default Role**: Acts as the default login shell for most Linux distributions.

**Scripting with Bash**

* **Purpose**: Automates the execution of tasks, reducing manual effort and minimizing human error.
* **Usage**:
  + Automate repetitive tasks.
  + Manage system operations efficiently.

**Applications of Bash Scripts**

1. **File Manipulation**
2. **Executing Routine Tasks**: E.g., Backup operations.
3. **Automation**: Simplifies complex workflows.

**Advantages of Bash Scripts**

* Easy to use and simple.
* Eliminates repetitive tasks.
* Allows the execution of a sequence of commands as a single command.
* Automates frequently performed tasks.

**Disadvantages of Bash Scripts**

* Errors can be costly.
* Each shell command launches a new process, slowing execution.
* Potential compatibility issues across platforms.

**Steps to Write a Bash Script**

1. **Create a File**: Use the .sh extension.
2. **Write Scripts**: Add commands within the file.
3. **Provide Execution Permission**: Use chmod +x scriptname.sh.

**Creating and Running a Shell Script**

* **Command**: Open a text editor and create a file:

gedit scriptname.sh

* **Start with Shebang**:

#!/bin/bash

* **Example**:

#!/bin/bash

echo "Hello, GeeksforGeeks"

* **Save & Provide Execution Permission**:

chmod +x scriptname.sh

* **Execute**:

./scriptname.sh

* **Output**:

Hello, GeeksforGeeks

**Variables in Bash Scripting**

**Types of Variables**

1. **Local Variables**:
   * Accessible only within the function or script where declared.
   * Example:

local my\_var="Hello"

1. **Global Variables**:
   * Accessible throughout the shell session.
   * Example:

export my\_var="Global"

1. **Environment Variables**:
   * Predefined by the system (e.g., $PATH, $USER).
2. **Special Variables**:
   * Predefined for specific purposes.
   * Examples:
     + $0: Script name.
     + $1, $2, ...: Command-line arguments.
3. **Positional Parameters**:
   * Hold command-line arguments passed to a script.
   * Example:

echo "First argument: $1"

1. **Readonly Variables**:
   * Cannot be modified after declaration.
   * Example:

readonly my\_var="Fixed"

1. **Array Variables**:
   * Store indexed or associative lists.
   * Example:

my\_array=(1 2 3)

echo ${my\_array[0]}

1. **Temporary Variables**:
   * Used within loops or temporary contexts.

**Input and Output in Bash**

**Basics**

* Scripts can take inputs and produce outputs or errors.
* Example:

echo "Enter filename"

read filename

if [ -e $filename ]; then

echo "$filename exists"

cat $filename

else

cat > $filename

echo "File created"

fi

**Explanation**

* **First Run**:
  + Creates the file if it doesn’t exist.
* **Second Run**:
  + Displays the content of the existing file.
* **Redirection**:
  + >: Redirects stdout to a file.
  + 2>: Redirects stderr.
  + &>: Redirects both stdout and stderr.

**Functions in Bash**

**Definition**

* A block of code that performs tasks and can be reused.
* **Syntax**:

function\_name() {

commands

}

* **Example**:

myFunction() {

echo "Hello World"

}

myFunction

* **Output**:

Hello World

**Decision Making in Bash**

**If-Else Statement**

* **Purpose**: Executes code based on conditions.
* **Syntax**:

if [ condition ]; then

statements

fi

* **Example**:

if [ "$Name" = "Satyajit" ]; then

echo "His name is Satyajit."

fi

**Case-Sac Statement**

* **Purpose**: Simplifies multiple condition checks.
* **Syntax**:

case $var in

Pattern1) Statement1 ;;

Pattern2) Statement2 ;;

esac

* **Example**:

case "$Name" in

"Satyajit") echo "Content Writer" ;;

esac

**Comparisons in Bash**

**String Comparison**

* Operators:
  + ==: Equal.
  + !=: Not equal.
  + -n: Non-null string.
  + -z: Null string.
* Example:

if [ "A" == "A" ]; then echo "Equal"; fi

**Numeric Comparison**

* Operators:
  + -eq: Equal.
  + -ne: Not equal.
  + -gt: Greater than.
* Example:

if [ 5 -gt 3 ]; then echo "Greater"; fi

**System-Defined Variables**

* Examples:
  + $USER: Logged-in user.
  + $HOME: Home directory.
  + $PATH: Executable search path.

**Usage**:

echo "User: $USER, Home: $HOME"

**Bash Built-in Special Variables**

These predefined variables are crucial for managing scripts, debugging, and argument handling.

**Common Special Variables**

| **Variable** | **Description** |
| --- | --- |
| $0 | Script/command name. |
| $1, $2... | Positional arguments. |
| $@ | All arguments (as a list). |
| $\* | All arguments (as a single string). |
| $# | Argument count. |
| $? | Exit status of the last command. |
| $$ | PID of the current shell/script. |
| $! | PID of the last background command. |
| $\_ | Last argument of the last command. |

**Examples**

1. **Script Information**

echo "Script Name: $0"

1. **Handling Arguments**

echo "First argument: $1"

echo "Number of arguments: $#"

1. **Command Status**

ls /nonexistent

echo "Exit status: $?"

1. **Process Information**

echo "PID: $$"

1. **Last Argument**

echo "Previous argument: $\_"

**Difference Between $@ and $\***

| **Feature** | **$@** | **$\*** |
| --- | --- | --- |
| Format | Separate strings | Single string |
| Usage | Iterating arguments | Single entity |

Example:

for arg in "$@"; do echo "$arg"; done

echo "$\*"

**Arrays in Bash**

Arrays store multiple values in a single variable. Types:

1. **Indexed Arrays** (numeric keys):

fruits=("apple" "banana" "cherry")

echo "${fruits[1]}" # Outputs: banana

1. **Associative Arrays** (string keys, Bash 4.0+):

declare -A capitals

capitals[India]="New Delhi"

echo "${capitals[India]}" # Outputs: New Delhi

**Common Operations**

1. **Adding/Modifying Elements**

fruits[3]="orange"

capitals[France]="Paris"

1. **Reading All Elements**

echo "${fruits[@]}"

echo "${!capitals[@]}" # Keys

echo "${capitals[@]}" # Values

1. **Length**

echo "${#fruits[@]}" # Number of elements

1. **Iterating**

for fruit in "${fruits[@]}"; do echo "$fruit"; done

for key in "${!capitals[@]}"; do echo "$key: ${capitals[$key]}"; done

1. **Deleting**

unset fruits[2]

unset capitals[India]

**Example Script: Indexed and Associative Arrays**

#!/bin/bash

# Indexed Array

fruits=("apple" "banana" "cherry")

echo "Fruits: ${fruits[@]}"

# Associative Array

declare -A capitals

capitals[USA]="Washington, D.C."

capitals[Japan]="Tokyo"

for country in "${!capitals[@]}"; do

echo "$country: ${capitals[$country]}"

done

**Bash Variable Types**

Variables are typeless by default, but declare and local provide control.

1. **Integer Variables**

declare -i num=5

num+=5

echo "$num" # Outputs: 10

1. **Readonly Variables**

declare -r pi=3.14

1. **Exported Variables**

declare -x PATH="/custom/path:$PATH"

1. **Local Variables**

my\_function() {

local msg="Hello"

echo "$msg"

}

### Code 1: Voting Eligibility Check

#!/bin/bash # Specify Bash as the script's interpreter

Age=17 # Global variable for age

# Check if age is 18 or older

if [ "$Age" -ge 18 ]; then

echo "You can vote" # Output if eligible to vote

else

echo "You cannot vote" # Output if not eligible to vote

fi # End of if-else block

# Update variables

Name="SATYAJIT GHOSH" # Global variable for name

Age=20 # Update age

# Print the updated name and age

echo "The name is $Name and Age is $Age"

### Code 2: Global and Local Variables in Bash

#!/bin/bash # Specify Bash as the script's interpreter

var1="Apple" # Global variable declared outside the function

myfun() {

local var2="Banana" # Local variable, accessible only inside this function

var3="Cherry" # Global variable, declared without 'local'

# Access and print variables inside the function

echo "The name of first fruit is $var1" # Access global variable

echo "The name of second fruit is $var2" # Access local variable

}

myfun # Call the function

# Access variables outside the function

echo "The name of first fruit is $var1" # Access global variable

echo "The name of second fruit is $var2" # Local variable not accessible, prints nothing

echo "The name of third fruit is $var3" # Access global variable defined in the function

4o