SHELL SCRIPTING

Basics of Shell Scripting:

1. What is Shell Scripting:

Shell scripting involves writing a series of commands for the shell (command-line interpreter) to execute.

Provides a way to automate repetitive tasks, execute commands in sequence, and create custom functions.

2. Components of a Shell Script:

Shebang (#!) Line:

Specifies the path to the shell that should be used to interpret the script.

Example: #!/bin/bash for a Bash script.

Comments:

Lines starting with # are comments and are ignored by the shell.

Used for documentation and explaining code.

Variables:

Containers for storing data.

Example: name="John"

Commands:

Executable statements that perform specific actions.

Example: echo "Hello, $name"

Control Structures:

Statements that control the flow of execution.

Example: if, else, for, while loops.

Exit Status:

Every command returns an exit status (0 for success, non-zero for failure).

Types of Shells:

1. Bash (Bourne Again Shell):

Most widely used on Unix-like systems.

Combines features from the original Bourne Shell (sh) and enhancements from other shells.

2. Sh (Bourne Shell):

Original Unix shell.

Limited features compared to modern shells like Bash.

3. Csh (C Shell):

Syntax similar to the C programming language.

Popular among some Unix users.

4. Ksh (Korn Shell):

Developed by David Korn at Bell Labs.

Incorporates features from both the Bourne Shell and C Shell.

5. Zsh (Z Shell):

An extended version of Bash with additional features.

Offers advanced auto-completion and theming.

How to Create Your First Script:

1. Choose a Text Editor:

Use a text editor like nano, vim, or gedit to create the script.

2. Write the Script:

Start with the shebang line to specify the shell.

Add comments for clarity.

Example:

```
#!/bin/bash
# This is a simple Bash script
echo "Hello, world!"
```

3. Save the Script:

Save the file with a .sh extension, e.g., myscript.sh.

4. Make the Script Executable:

Run chmod +x myscript.sh to make the script executable.

5. Execute the Script:

Run ./myscript.sh to execute the script.

6. Debugging:

Use echo statements for debugging.

Check the exit status of commands ($?).

Example:

#!/bin/bash

This script greets the user

echo "Enter your name:"

read name

echo "Hello, $name! Welcome to Shell Scripting."

Conditional Statements:

1. if Statement:

Syntax:

if [condition]; then # Commands to execute if condition is true fi

Example:

if ["$name" == "John"]; then

echo "Hello, John!"

else

echo "Hello, stranger!"

fi

2. elif Statement:

Syntax:

if [condition]; then

Commands to execute if condition is true elif [another_condition]; then # Commands to execute if another_condition is true else # Commands to execute if none of the conditions are true fi

Example:

if ["$grade" -ge 90]; then

echo "Excellent!"

elif ["$grade" -ge 80]; then

echo "Good!"

else

echo "Need improvement." Fi

Case Statement:

case Statement:

Syntax:

case expression in pattern1) # Commands to execute for pattern1 ;; pattern2) # Commands to execute for pattern2 ;; *) # Default commands if no pattern matches ;; esac

Example:

case "$fruit" in

"apple") echo "It's an apple." ;;

"banana" | "orange") echo "It's a banana or an orange." ;;

*) echo "Not sure what it is." ;;

Esac

Loops:

1. for Loop:

Syntax:

for variable in list; do # Commands to execute for each item in the list done

Example:

for i in {1..5};

do echo "Iteration $i"

done

2. while Loop:

Syntax:

while [condition]; do # Commands to execute as long as the condition is true done

Example:

count=1

while [$count -le 5];

do echo "Count is $count"

((count++))

Done

Exit Status:

1. Exit Status of Commands:

Every command returns an exit status.

$? variable holds the exit status of the last executed command.

0 indicates success, non-zero values indicate failure.

2. Using Exit Status in Scripts:

Example:

```
grep "pattern" file.txt
if [ $? -eq 0 ]; then
echo "Pattern found!"
else
echo "Pattern not found."
fi
```

3. exit Command:

Exits the script with a specified status.

Example:

```
if [ condition ]; then
echo "Condition is true."
exit 0 # Exit with success status
else
echo "Condition is false."
exit 1 # Exit with failure status
fi
```